Machine learning as an alternative to thresholding for space radiation high current event detection

Adrien Dorise, Corinne Alonso, Audine Subias, Louise Travé-Massuyès, Leny Baczkowski, François Vacher

Machine learning as an alternative to thresholding for space radiation high current event detection

Adrien Dorise, Corinne Alonso, Audine Subias, Louise Travé-Massuyès, Leny Baczkowski and François Vacher

***Abstract*—The space environment is known to be the seat of radiation of different kinds to which satellites in orbit are subjected. These include cosmic rays that come from stars and radiation belts that come from the Earth magnetic field. The impact of radiation on electronic components results in anomalies called "Single Event Effects" which can lead to the destruction of equipment. Various protection methods exist, like hardening of components or satellite shielding, but they are often costly and/or difficult to implement. This is why space designers try to circumvent these processes by an efficient software protection method. This paper reports a set of experiments based on machine learning tools that will provide the basis to design and develop an anomaly detection method for Single Event Effects. The data sets that were used are issued from emulated radiations obtained by laser tests on a SAM3X microcontroller, complemented by data obtained by simulation.**

***Index Terms*—Fault detection, Machine learning, Electronic applications, Aerospace engineering, Satellites, Space Radiations**

I. Introduction

Since the launch of the first satellite Sputnik-1 on the 4th October 1957, efficiency of electronic components have increased significantly. In consequence, electronic devices have become smaller. However, coming with these technological breakthroughs, a new problem has emerged in space applications. Space electronic became sensitive to space radiation and thus, new types of faults appeared [1]. These faults are called *Single Event Effects* (SEEs) [2]. Single Event Effects in a micro-electronic device are caused by a single energetic particle and can take many forms. It can go from a logical change of state also called *single event upset*, to the modification of the component's supply current that constitutes the so-called *Single Event Latch-up* (SEL). This work focus on SEL.

SEL occurs when an energetic particle triggers a low-impedance path leading to a parasitic structure in the substrate of a micro-component [3] [4]. This leads to a *High Current Effect* (HCE) that can be destructive. However, when detected in time, a power cycling is able to protect the component from any damage. Therefore, it is a real advantage to detect correctly single event latch-up. In addition to the destructive high current event caused by SEL, μlatch-up phenomena, causing small HCE, have to be considered. These faults are not destructive but can paralyse a function of the system or induce long-term damages. Based on Chandola [5], SEL can be described as collective anomalies because multiple points are diverging far from the nominal behaviour. As for μlatch-up, they can be described as collective and contextual anomalies, as they can remain hidden in the nominal behaviour of components.

The most common SEL detection method relies on an *anti latch-up system*: a threshold is set on the supply current [6]. When the current is higher than the limit, an alarm is raised and a power cycling is triggered. The main drawback of this system is that it cannot detect μlatch-up hidden in the supply current, as the threshold is chosen within a margin to the supply current maximum peak. It is why extensive researches are done to improve this method. Specific components are emerging with the purpose of detecting HCE [7]. One patent about a self adjusting threshold is filled [6], proposing less margin between the threshold and the supply current. Another patent [8] uses a detection method based on two detectors. The first one monitors the absolute load current and the second one analyses the current rate of change. Also, Airbus recently funded Zero-Error Systems to work on radiation-proof space systems [9]. Researches on space radiation detection are a challenging issue of prime importance, as they can significantly improve satellites durability and reduce the costs due to radiation hardened technology. This paper presents a preliminary work on a new anti latch-up system based on machine learning algorithms. We focus on classification and clustering, in an attempt to see how machine learning techniques can improve HCE, and more specifically, small HCE detection.

The paper is divided as in the following. Section 2 reviews the implementation of a test bench for the creation of data sets. Section 3 reviews the results of HCE detection, first using supervised learning, then using a combination of both unsupervised and supervised learning. Finally, section 4 draws conclusion and open on the future work.

II. Data sets creation

In this paper, we are trying to detect the effects of space radiations on electronic devices. Therefore, The first step is to select a component that we could use to create our case study.

A. A space case study

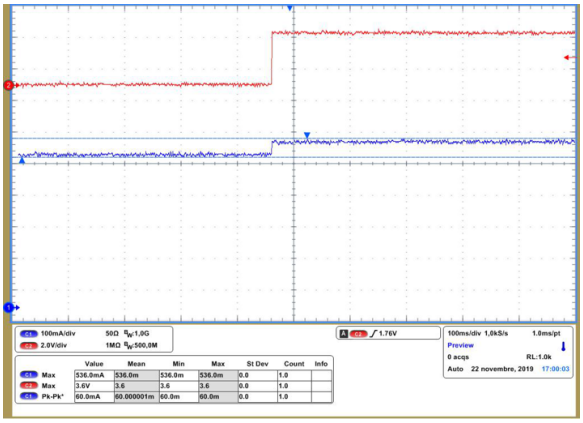
The Atmel SAM3X microcontroller has been chosen for our case study, as this electronic component is used in real-case space application. For instance, the radiation tolerant version

Submited for review the 09/04/2021.

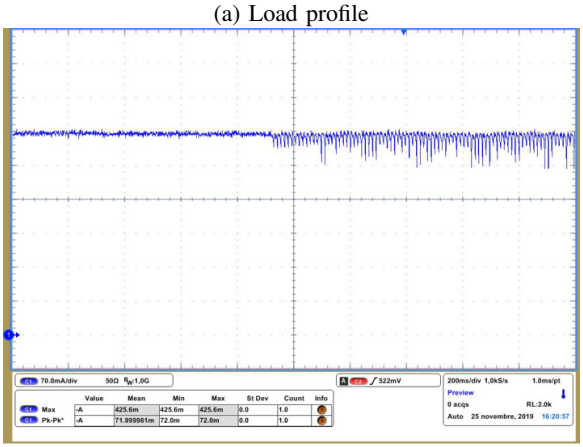
This work was granted by the CNES and Région Occitanie.

Adrien Dorise, Corinne Alonso, Audine Subias and Louise Travé-Massuyès are working in LAAS-CNRS, France (e-mail: name.surname@laas.fr)

Leny Baczkowski and François Vacher are working in CNES, France (e-mail: name.surname@cnes.fr)



(a) Load profile



(b) Software profile

Fig. 1: SAM3X supply current profiles depending on the activity

was implemented on the ANGEL satellite launched in December 2019. Moreover, its non radiation-hardened version is widely commercialized on development boards. The SAM3X is a 32 bits flash microcontroller based on the Cortex M3 processor. It operates at 84MHz, features 512 Kb of flash memory and possesses 103 I/O lines.

Experiments are made to understand the behaviour of the chip supply current. Two different supply current profiles have been highlighted. The first one is directly linked to the I/Os of the microcontroller. Indeed, a change in an output induces a modified electrical load, which in turn triggers a modification in the supply current amplitude. The second profile depends on the state of the chip: when the SAM3X is performing a calculation or a memory state modification, the behaviour of the current is impacted. In this case, the amplitude is unchanged, but the variance is. An example of these two profiles is shown in figure 1. These profiles allow a better understanding of the supply current normal behaviour. They can be considered as a guide to build and implement real case scenarios of a satellite operating. With these scenarios it is possible to generate samples associated to the normal behaviour.

In supervised learning, both normal and abnormal labels are required. We now have the first half with normal labels.

B. A laser experiment to generate abnormal data

For anomaly detection purpose, abnormal samples must also be considered. In our case, it means injecting single event latch-up faults in the SAM3X. To do so, a laser test bench was done. Indeed, by pointing a laser on sensitive nodes of a microcontroller, it is possible to create HCE. Note that it is not possible to say with certainty that the faults created are indeed SEL, but they are close enough to be valid in our study.

First, the upper face of the SAM3X is removed using atmospheric plasma to access the internal components of the chip. Then, the test bench is designed as shown in Figure 2. Below is a description of the different equipment used.

- *U1:* Tenma 72-8690A DC Power supply featured with two 32V, 3A outputs
- *U2:* MAX17612CEVKIT evaluation board circuit protection. This equipment is used as a protection device to avoid destruction of the device under test. It also send an alarm when the supply current is higher than a set threshold.
- *U3:* Device Under Test (DUT). An Arduino Due development board equipped with an ATSAM3X8E microcontroller.
- *A1/V1:* Keithley DAQ6510 to monitor supply current and voltage.
- *PC_Laser:* Computer linking *U2* to the laser software. This software uses this information to pinpoint sensitive nodes in the DUT.
- *PC_Manip:* Computer recording the data.

The test campaign was done at CNES in Toulouse, France. The laser characteristics are: wavelength $\gamma = 1064nm$; average power $Pavg = 1250mW$ and impulsion time $t = 125\mu s$.

Two types of faults occurred during the tests: punctual current variations and high current events. As the HCEs are pretty similar to fault created by space radiation, it is sufficient for our study, making possible to create a complete data set including both normal and abnormal data.

However, even though these data are similar to a real case scenario, it is impossible to control the abnormal behaviour of an electronic component. No example of μlatch-up was found, and only few examples of abnormal data were generated during the test campaign. Thus, it appears that these laser experiments are not sufficient to provide enough faulty behaviour samples needed to perform machine learning detection. As a consequence, a simulation framework is created in order to access easily a large amount of data sets.

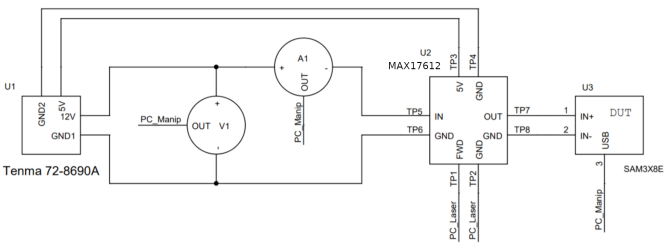


Fig. 2: Laser bench diagram

C. A simulation framework to generate data sets

Based on the data generated by laser tests, multiple data sets are simulated in MATLAB. The data gathered from the experimental testing are used to mimic the behaviour of a real microcontroller. A simulated data set can be defined as $D(t) = \{I(t), y(t)\}$ where $I(t)$ and $y(t)$ are the supply current and the class labels at time $t \in \Re_+$. The variable $I(t)$ is the result of multiple functions and can be described as:

$$\begin{aligned} I(t) = (f_{Nominal}(t, \bar{I}) + \sum_i f_{Load_i}(t, l_i, t_i, d_i) \\ + \sum_j f_{Soft_j}(t, s_j, t_j, d_j) + \sum_k f_{Temp_k}(t, a_k, t_k, d_k) \\ + \sum_l f_{HCE_l}(t, f_l, t_l, r_l)) * \sum_m f_{Reset_m}(t, t_m, d_m) \end{aligned} \quad (1)$$

with $i, j, k, l, m \in \mathbb{N}^+$ indexing the multiple occurences of each function. The $f_{Nominal}$ function simulates the base current of a microcontroller. It is done by taking the mean current of the component $\bar{I}$ and adding normally distributed noise [10]:

$$f_{Nominal}(t, \bar{I}) = \bar{I} + Noise(t) \quad (2)$$

The f_{Load_i} function simulates the electrical load created by a component on the microcontroller. It is defined as a rectangular function:

$$f_{Load_i}(t, l_i, t_i, d_i) = \begin{cases} 0 \text{ if } t < t_i \text{ and } t > t_i + d_i \\ \frac{l_i}{2} \text{ if } t = t_i \text{ or } t = t_i + d_i \\ l_i \text{ if } t > t_i + d_i \text{ and } t < t_i + d_i \end{cases} \quad (3)$$

with l_i the added load current, t_i the time when the load begins, d_i the duration. f_{Soft_j} corresponds to the current modifications induced by the internal processing of the microcontroller (calculations or memory modification for example). When active, this function amplifies the noise already present in the $f_{Nominal}$ function:

$$f_{Soft_j}(t, s_j, t_j, d_j) = \begin{cases} 0 \text{ if } t < t_j \text{ and } t > t_j + d_j \\ -1^{r(t)} * s_j \text{ if } t \geq t_j \text{ and } t \leq t_j + d_j \end{cases} \quad (4)$$

with s_j the added noise, $r(t)$ a function alternating between 0 and 1, the random function, t_j the time when the function begins, d_j the duration. The f_{Temp_k} function corresponds to a modification of the component environment temperature. A linear function is then applied to the data set:

$$f_{Temp_k}(t, a_k, t_k, d_k) = \begin{cases} 0 \text{ if } t < t_k \\ a_k(t - t_k) \text{ if } t \geq t_k \text{ and } t \leq t_k + d_k \\ a_k(t_k + d_k) \text{ if } t > t_k + d_k \end{cases} \quad (5)$$

with a_k the linear coefficient of the variation, t_k the time when the variation begins, d_k the duration. The $Reset_m$ function simulates a power cycling:

$$f_{Reset_m}(t, t_m, d_m) = \begin{cases} 1 \text{ if } t < t_m \text{ and } t > t_m + d_m \\ 0 \text{ if } t \geq t_m \text{ and } t \leq t_m + d_m \end{cases} \quad (6)$$

with t_k the time when the reset begins, d_k the duration. Finally, the f_{HCE_l} function simulates persistent high current events. It is similar to f_{Load_i}, except that the modification stays active permanently or until f_{Reset_m} is performed:

$$f_{HCE}(t, f_l, t_l, r_l) = \begin{cases} 0 \text{ if } t < t_l \text{ and } t \geq r_l \\ f_l \text{ if } t \geq t_l \text{and } t < r_l \end{cases} \quad (7)$$

with f_l the fault magnitude, t_l the time when the fault begins, r_l the time of the next reset (the end of the data set if no reset occurs afterwards).

Using equation (1), a complex normal behaviour scenario can be created. Finally, small HCE are added to the dataset to get a complete representation of our problem. Figure 3 gives an example of a data set generated with the simulation principle presented above. For this study, the faults injected are small so they can be easily misjudge with nominal behaviour. Indeed, classic threshold detection wouldn't be able to detect such faults. This study is done by using ten simulated training sets of 10,000 samples and twenty simulated test sets of 1,000 samples. The sampling frequency is set to 1 second per sample.

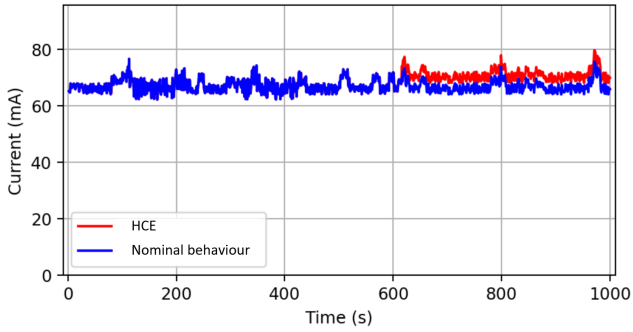


Fig. 3: Simulation of the supply current

The next section presents the different machine learning algorithms tested on these data sets for anomaly detection purposes.

III. Machine learning anomaly detection

In this section, machine learning algorithms are considered to improve the existing latch-up detection system. The algorithms are expected to learn the normal behaviour of the chip in order to distinguish a fault when a HCE occurs. For this purpose, supervised classification algorithms are considered first as a proof of concept. Then, unsupervised algorithms are tested. But before, we need to gather more information from the signal to feed our algorithms.

A. feature extraction

The data used in this study are composed of time series. The only features currently available are supply current value and time. However, HCEs caused by SEL are created by energetic particles colliding with a satellite at random times. In that case, we consider that time is an irrelevant feature in HCE detection. The first objective is to extract additional relevant information from the time series to enable characterizing HCE

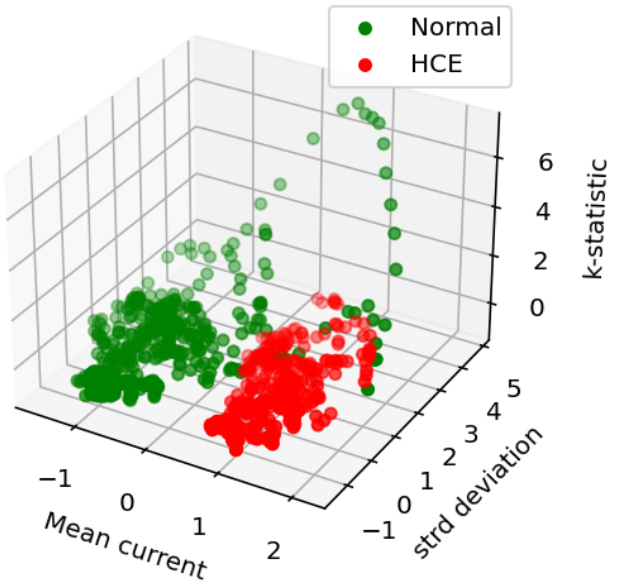


Fig. 4: Statistical features calculated using sliding time window. The values are normalized

signature. Statistical features are calculated using the Python package *stats* from scikit-learn [11]. The minimum, maximum, geometrical mean, variance, standard deviation of the mean, median absolute deviation and k-statistic are calculated from the data set. Still, we cannot just perform statistical calculation on all samples of the set at once, as we want a statistical value for each sample of the set. Thus, time windows have to be defined.

Two methods are tested to define the time windows. The first method is based on a sliding time-window. The k neighbours of a particular point are taken to calculate the statistical features. In consequence, a statistical value for each point of the data set is collected. The second method uses the Python package Ruptures [12]. This method calculates the ruptures occurring in a time series. Indeed, HCE can be defined as a rupture in the supply current. So we can consider that each rupture in our data set can hide a HCE. To calculate the features, the data points between two current ruptures are used as time windows.

The result of the first time window transformation using three statistical features is visible on Fig. 4. The two classes are also visible and a separation is appearing. We now have to classify these samples using multiple data sets.

B. Supervised anomaly detection

Supervised anomaly detection assume the availability of both normal and abnormal labels. It builds a model based on classes to determine whether or not a new sample is an anomaly [5]. In our case, we distinguish normal behaviour from HCE classes. The most popular algorithms were tested to have a good overview of the possibilities given by machine learning. Each of them is briefly described below.

1) Algorithms used:

- *K-Nearest Neighbors:* K-nn is a non-parametric method that consists in finding the class of a point based on its k nearest neighbours in the feature space (Altman [13])
- *Naïve Bayes:* This method is based on applying Bayes's theorem (see equation (8)) with a strong independence assumption between the features.

$$P(A\backslash B) = \frac{P(B\backslash A) + P(A)}{P(B)} \tag{8}$$

- *Decision Trees:* This method uses tree-like models (Quinlan [14]) where the target variable can take a value in a discrete set. In these structures, the leaves represent class labels and branches represent conjunction of feature values. This method is very popular because of its easy to understand principle.
- *Random Forests*: This method is an *ensemble learning* method, meaning that it is based on multiple machine learning algorithms working together to get a better result. In this case, a multitude of decision trees are used to build the prediction.
- *Support Vector Machines*: SVM models maps training examples to points in space so as to maximize the width of the gap between the two classes. To do so, the algorithm uses support vectors to calculate the maximum margin between classes and find a linear correlation between data. For non-linear separation, it is possible to use the *kernel trick*, searching for the boundary in a higher dimension space (Cortes [15]).

2) Results:

To get the global accuracy of the algorithm, the mean accuracy of a total of 200 combinations of training set and test set is calculated. All seven statistical features are used. When the training is completed, boundaries are created, splitting the different classes. Using these boundaries, it is possible to predict the class of future points (in our case, the points of test sets) and therefore evaluate the model.

The accuracy of the different algorithms is obtained using equation (9).

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{9}$$

The results are available in Fig. 5 for both sliding time window and rupture window. Detailed results are available in Table. V. In our case, we consider that a supervised learning algorithm is satisfactory when its accuracy is between 90% and 98% (higher would be a sign of overfitting). From the results, we can conclude that most of the algorithms performed well. Moreover, SVM has the best performance. The accuracy of the rupture method is lower compared to sliding window. However, as this method does not require to calculate the features for each point in the data set, it enables to optimize computation time. Thus, we design a protection for a time-critical system and advantages of both methods have to be considered before choosing the most suitable.

In the light of the good results given by supervised classification methods regarding the detection of μlatch-ups, we

can validate learning algorithm as a possible improvement to the current anti-latch-up system. The next step is to conclude about unsupervised learning algorithms.

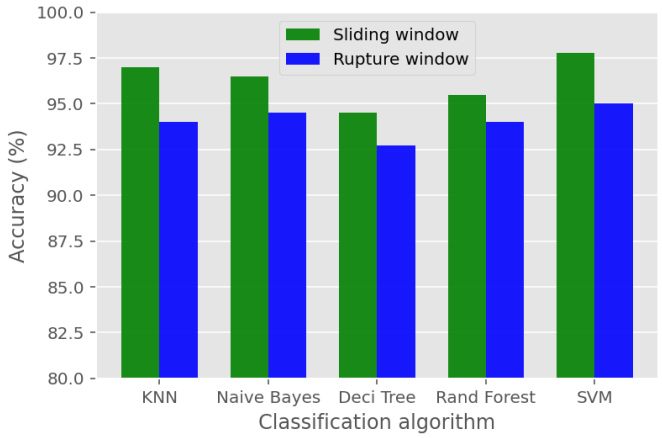


Fig. 5: Classification algorithms accuracy

C. Unsupervised anomaly detection

We demonstrated that supervised learning prediction is acceptable enough for the detection of small HCE. However, supervised learning is difficult to apply in real space applications as labelling the date is time-consuming and hence expensive. In that case, unsupervised clustering algorithms are considered instead of supervised classification, because class labels are not needed by the algorithm. Unsupervised learning algorithms determine classes based on similarity criterion. As before, normal and abnormal data are present in the training set.

1) Algorithms used:

- *K-means clustering:* This vector quantization method aims to partition N observations into K clusters. Each point is assigned to the cluster with the nearest mean. K must be chosen by the user. Some methods exist to help deciding about the K parameter. The one used in this paper is the *Elbow method*. It consists in calculating the Within Cluster Sum of Squares (WCSS) (see equation (10)) for a multitude of K value and then finding the *elbow* on the curve. This break is the optimal number of clusters K

$$WCSS = \sum_{clusters_j} \sum_{points_i} distance(P_i, C_j)^2 \qquad (10)$$

  where C_i is the centroid for observation P_i.
- *Hierarchical Clustering:* This method aims to build a hierarchy of clusters [16]. There are generally two strategies: *agglomerative* when starting with one cluster for each observation and merging them until all data form one unique cluster, and *divisive* when starting with one single cluster, and spliting it recursively as one moves down the hierarchy. As in K-means clustering, the user has to specify the number of desired clusters. In this paper, we use the *dendrogram* method to select K.
- *Density-Based Spatial Clustering of Applications with Noise (DBSCAN):* DBSCAN is a density-based clustering non-parametric algorithm [17]. It aims to group data points that are closely packed together. It also marks as outliers points that lie in low-density regions. The main parameter to set is the radius of a neighbourhood ϵ. Note that this algorithm does not need to specify the number of desired clusters.
- *Dynamic clustering for tracking evolving environments (DyClee):* DyClee is a two-stages distance-based and density-based clustering algorithm [18]. Data samples are fed as input to the distance based clustering stage in an incremental, online fashion, and they are then clustered to form μclusters. The density-based algorithm analyses the micro-clusters to provide the final clusters. Thanks to a forgetting process, clusters may emerge, drift, merge, split or disappear, hence following the evolution of the environment. Like DBSCAN, DyClee does not require the number of desired clusters, instead the main parameter to be set is *g_size* that defines the size of the μclusters.

2) Parameter selection:

As clustering does not use class labels, it is important to fine-tune the algorithms. Results are heavily sensitive to parameters selection, it is hence important to find the best combination.

Unlike classification, results now show multiple clusters without any label. To cope with this, the *expert opinion* method is used on the data set. Every cluster created by the algorithms is assigned to a ”normal” or ”abnormal” label depending on the true class (assuming that it is known) of the majority of samples in the cluster. This method allows the creation of labelled data based on clusters found by the algorithm.

Even though Kmeans and Hierarchical clustering generate an a priori specified number of clusters, it is not the case for DBSCAN and DyClee. In our case study, the optimal number of classes would be two classes, so the best possible accuracy should be achieved with the least number of clusters. To evaluate this, a *Score function* is implemented (see equation. 11) that gives a penalty when the number of clusters is too high.

$$Score = \begin{cases} Accuracy & \text{if } \frac{C}{N} <= R \\ \frac{Accuracy}{\sqrt[P]{e^{(\frac{C}{N}-R)}}} & \text{if } \frac{C}{N} > R \end{cases} \qquad (11)$$

with *P* the penalty, *C* the clusters created, *N* the true classes and *R* the ratio. The values to be selected by the user are the Penalty P and the ratio R. P influences the weight of the penalty when there is a big number of clusters. R establishes the clusters ratio threshold when the penalty is applied. Using this score, we are able to select more efficiently the parameters of DBSCAN and DyClee to limit the amount of clusters.

During these tests, the size of the simulated training sets was too important, and results were inconclusive. This is why we chose to perform our study using only the simulated test sets of 1,000 samples. Ten data sets are used. The mean accuracy and score is then calculated to find out the optimal combination

(see tables I, II, III). $R = 10$ and $P = 25$ are used as parameters for the score function.

TABLE I: Hierarchical clustering parameters testing

Hierarchical clustering		
Param	value	Mean accuracy
Metric	Euclidean	75.93%
	Cosine	**94.25%**
	Manhattan	74.84%

TABLE II: DBSCAN parameters testing

DBSCAN			
Param	value	Average accuracy	Average score
Epsilon ϵ	0.0005	67.89%	67.04%
	0.001	68.72%	68.65%
	0.005	**77.19%**	71.03%
	0.008	71.85%	**71.59%**
	0.01	69.73%	69.73%
	0.03	59.90%	59.90%
Metric	Euclidean	**66.56%**	**66.56%**
	Cosine	66.34%	66.34%
	Manhattan	66.39%%	66.39%

TABLE III: DyClee parameters testing

DyClee			
Param	value	Average accuracy	Average score
g_size μclusters	0.07	93.14%	28.13%
	0.08	**95.10%**	42.56%
	0.10	94.31%	69.79%
	0.12	92.12%	**89.46%**
	0.14	74.30%	74.30%
	0.16	77.66%	77.66%

Considering hierarchical clustering, *Cosine metric* largely outscore the other metrics. However, the results are not as obvious for the metric of DBSCAN. Looking at the epsilon ε parameter of DBSCAN, ε=0.005 had the best accuracy, but the score of ε=0.008 is greater. Therefore, even if ε=0.005 is better in predicting classes, the number of clusters created is significantly higher, meaning that it is better to use ε=0.005 as our epsilon parameter. The same case occurs for DyClee, where the final choice for g_size is 0.12. As a matter of fact, the defined score worked pretty well to indicate when too many clusters are created and it helped to select the best parameters. It is important to note that the score indicator should only be used for parameter tuning. Indeed, parameter values determine how many clusters will be created, so using the score function afterwards would be superfluous.

Considering the number of clusters K for Kmeans and hierarchical clustering, both WCSS and dendogram methods explained above showed an optimal number of clusters of K=3.

3) Combination of unsupervised and supervised learning:

After parameters tuning, let us consider prediction using clustering algorithms. The methods consist in training the algorithm on a training set. Then the expert opinion is used to assign the different clusters to either normal or abnormal labels. After that, the newly created classes are used to train a classification algorithm on the same train set. The result is the creation of boundaries between classes, providing the model to be used on new data sets and giving us an accuracy score as in the classification section. The algorithm is summarized in table IV.

TABLE IV: Clustering with classification methodology

1) Get train set $train_i$
2) Apply clustering on $train_i$, giving C clusters
3) Expert opinion reducing C clusters in N classes
The number of cluster is now Cn
4) Apply classification with Cn labels on $train_i$
5) Test classification on multiple test sets $test_j$
6) Calculate Accuracy

The main drawback of this method is that it multiplies prediction errors of both classification and clustering algorithms. However, the benefits are that the accuracy results are a relevant estimation of the performance of unsupervised learning techniques in the detection of HCE. In addition, as we have seen in the supervised detection section, the performance of classification are good enough to have only a small impact on clustering predictions. For our tests, SVM classification algorithm is used, as it had the best results among the other algorithms.

4) Results:

The results are shown in Fig. 6 and detailed results are available in Table. V. They show great results for Kmeans, Hierarchical clustering and DyClee. The under performance of DBSCAN is probably due to its density based calculation. Indeed, our case study implies that we work with unbalanced data (normal data are prominent compared to abnormal data) and weights are not applied to counterbalance it. Even though DyClee also uses a density method, it employs both distance and density, which can explain its good performance. Once again, the overall results are encouraging as most of the algorithms shows an accuracy around 90% for the sliding time window method. Seeing this, it is clear that machine learning is able to approve overall performance of threshold detection method. As a reminder, threshold detection wouldn't be able to detect any HCE in the data sets presented to the machine learning algorithms.

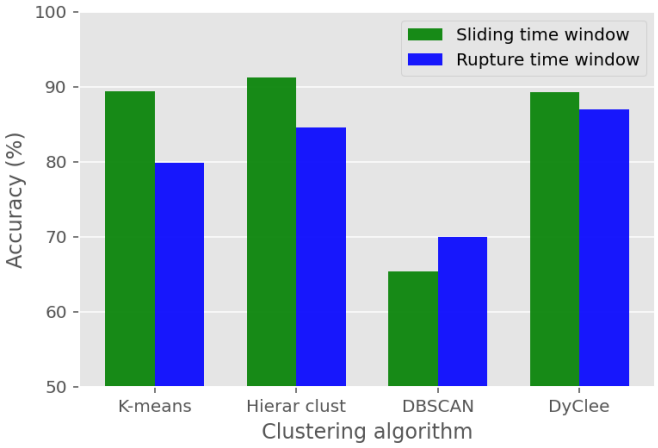


Fig. 6: Clustering algorithms accuracy

TABLE V: Detailed results of machine learning algorithms for HCE detection (TP: True Positive, FP: False Positive, FN: False Negative, TN: True Negative)

		TP	FP	FN	TN
KNN	Sliding window	33.42%	1.46%	1.18%	63.95%
	Rupture window	24.72%	1.53%	4.44%	69.31%
Naive Bayes	Sliding window	34.17%	2.49%	0.42%	62.92%
	Rupture window	26.53%	3.10%	2.64%	67.73%
Decision tree	Sliding window	30.46%	1.11%	4.13%	64.30%
	Rupture window	23.76%	2.16%	5.51%	69.18%
Random forest	Sliding window	31.28%	0.95%	3.31%	64.46%
	Rupture window	24.31%	1.11%	4.86%	69.72%
SVM	Sliding window	33.26%	1.36%	1.33%	64.04%
	Rupture window	24.78%	1.57%	4.40%	69.26%
K-means	Sliding window	27.85%	3.87%	6.74%	61.53%
	Rupture window	10.23%	1.20%	18.93%	69.63%
HC	Sliding window	29.64%	3.73%	4.95%	61.67%
	Rupture window	15.83%	2.08%	13.33%	68.75%
DBSCAN	Sliding window	0%	0%	34.60%	65.40%
	Rupture window	0%	0%	29.17%	70.83%
DyClee	Sliding window	25.03%	1.15%	9.56%	64.25%
	Rupture window	17.41%	1.16%	11.76%	69.68%

IV. Conclusion

In this paper machine learning algorithms were investigated as an alternative to classical methods for space radiation faults detection. First, a case study of a radiation sensible device was developed and laser tests were performed for a better understanding of high current event. From this case study, a database was generated on which several machine learning algorithms were used for anomaly detection purposes. Second, multiple supervised classification algorithms were tested in order to detect high current event. Results showed good performance for most algorithms. In particular, μlatch-up can now be detected with acceptable accuracy, which anti latch-up system don't achieve. Third, unsupervised clustering was tested in combination of the previously tested supervised technics. Again, results were promising in the use of this method to enhance high current event detection.

Several points need to be investigated in the future. One would be to propose a machine learning method only based on normal data, as it can be difficult to get an exhaustive data set of all possible latch-up. The result of unsupervised learning being satisfactory, we intend to extend this work by testing semi-supervised methods on data sets excluding high current events. Furthermore, satellites are subject to ageing through time: the supply current of the equipment might vary, leading to an outdated and ineffective learning detection. We also plan to address this problem using the online feature of DyClee. Indeed, its ability to use a loss function, make possible to take in consideration the ageing of the components.

ACKNOWLEDGEMENTS

We would like to thank the CNES for providing the facilities to perform laser tests.

References

[1] E. Petersen, R. Koga, M. Shoga, J. Pickel, and W. Price, "The single event revolution," *IEEE Transactions on Nuclear Science*, vol. 60, pp. 1824–1835, 06 2013.

[2] R. Gaillard, *Single Event Effects: Mechanisms and Classification*, vol. 41, pp. 27–54. 09 2010.

[3] A. Al Youssef, L. Artola, S. Ducret, and G. Hubert, "Compact modeling of single-event latchup of integrated cmos circuit," *IEEE Transactions on Nuclear Science*, vol. 66, no. 7, pp. 1510–1515, 2019.

[4] F. Sexton, "Destructive single-event effects in semiconductor devices and ics," *Nuclear Science, IEEE Transactions on*, vol. 50, pp. 603 – 621, 07 2003.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, July 2009.

[6] R. Cibils, "Método para actualizar el umbral de referencia de al menos un parámetro operativo, unidad de protección para la mitigación de un evento simple de latchup (sel) en un dispositivo electrónico usando el umbral de referencia y disposición para la mitigación de un evento simple de latchup (sel) en un conjunto.," 11 2019.

[7] J. B. Carlsen, *Design and Validation of Two Single Event Latch-up Protection Solutions. Comparing a New Single Event Latch-up Test Circuit with the IDEAS IDE3466 Single Event Latch-up Detection Module*. PhD thesis, University of Oslo, 2018.

[8] J. J. Chang; Joseph Sylvester, Shu; Wei, "Electronic circuit for single-event latch-up detection and protection," 02 2020.

[9] D. Werner, "Airbus ventures adds zero-error systems to space portfolio," *SpaceNews*, 2020.

[10] *9. Random Numbers*, pp. 255–267.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[12] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[13] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[14] J. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[16] F. Murtagh and P. Contreras, "Methods of hierarchical clustering," *Computing Research Repository - CORR*, 04 2011.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, p. 226–231, AAAI Press, 1996.

[18] N. Barbosa Roa, L. Travé-Massuyès, and V. H. Grisales, "DyClee: Dynamic clustering for tracking evolving environments," *Pattern Recognition*, vol. 94, pp. 162–186, Oct. 2019.