POLITECNICO DI TORINO
Repository ISTITUZIONALE

Self-supervised Real NVP for on-board satellite fault detection

Original
Self-supervised Real NVP for on-board satellite fault detection / Cena, C., Albertin, U., Bucci, S., Martini, M., Balossino, A., Chiaberge, M.. - In: ACTA ASTRONAUTICA. - ISSN 0094-5765. - ELETTRONICO. - 239:(2026), pp. 49-60. [10.1016/j.actaastro.2025.10.068]

Availability:
This version is available at: 11583/3004763 since: 2025-11-03T21:44:02Z

Publisher:
Elsevier

Published
DOI:10.1016/j.actaastro.2025.10.068

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Elsevier preprint/submitted version

Preprint (submitted version) of an article published in ACTA ASTRONAUTICA © 2026, http://doi.org/10.1016/j.actaastro.2025.10.068

(Article begins on next page)

11 September 2026

Self-supervised Real NVP for on-board satellite fault detection ⋆

Carlo Cena[a,*], Umberto Albertin[a], Silvia Bucci[b], Mauro Martini[a], Alessandro Balossino[b], Marcello Chiaberge[a]

[a] *Department of Electronics and Telecommunications, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129, Torino, Italy*
[b] *Argotec, Via Luigi Burgo, 8, 10099, San Mauro Torinese, Italy*

Abstract

In the space sector, the harsh environmental conditions and limited accessibility make robust fault detection systems crucial for ensuring mission success. Many existing approaches are computationally heavy, making them unsuitable for real-world applications where satellites have limited computational capabilities. We introduce a very light and powerful normalizing flow model, Real NVP based, to accomplish the task of high robustness in fault detection with low computational power. A self-supervision technique is implemented in the framework to improve its robustness. Extensive experimentation was conducted on three publicly available datasets, benchmarking different configurations, including pre-training with self-supervision, multi-task learning, and standalone self-supervised training. We show good generalization capabilities with different data, including the recently released dataset of the European Space Agency. An ablation study is performed to analyze performance changes when introducing faults during both self-supervision and main loss stages of training. Finally, systems-on-chip and artificial intelligence accelerators are used to perform latency tests, verifying our frame-

⋆Funding: This work was supported by the project PNRR-NGEU which has received funding from the MUR – DM 351/2022 and MUR – DM 117/2023.

*Corresponding author

Email addresses: carlo.cena@polito.it (Carlo Cena), umberto.albertin@polito.it (Umberto Albertin), silvia.bucci@argotecgroup.com (Silvia Bucci), mauro.martini@polito.it (Mauro Martini), alessandro.balossino@argotecgroup.com (Alessandro Balossino), marcello.chiaberge@polito.it (Marcello Chiaberge)

work's feasibility and adaptability in real satellite scenarios. The proposed self-supervised model based on Real NVP achieves better results with respect to the state-of-the-art in terms of detection performance and inference time over multiple benchmark datasets of the field, also confirming its consistency on low-power edge devices.

Keywords: Signal processing, Space vehicle telemetry, Anomaly Detection, Normalising Flow, Machine Learning

---

1. Introduction

Satellite missions play a vital role in various fields, such as communication, navigation, and remote sensing, and are essential for monitoring climate change, predicting natural disasters, like floods, and analyzing vegetation trends for agriculture [1, 2, 3]. Additionally, the fast-growing space exploration sector is fueling the advancement of new satellite technologies. For instance, small CubeSats are now frequently used for tasks like observing asteroid impacts [4]. However, they face a unique set of challenges that make their operation and maintenance particularly difficult. These challenges include harsh environmental conditions, such as extreme temperatures, high radiation levels, and the constant threat of collisions with space debris, all of which can cause significant damage to spacecraft [5]. Additionally, communication delays due to the vast distances between satellites and ground control stations hinder real-time monitoring and troubleshooting, often requiring satellites to operate autonomously for extended periods. The limited power, memory, and processing capabilities on-board also constrain the ability to implement complex monitoring and fault-detection systems. Moreover, the highly integrated nature of satellite subsystems -such as propulsion, power, and communication- means that a fault in one area can easily propagate to others, amplifying the risk of mission failure.

Conventional fault detection methods, whether rule-based or model-based ([6, 7]), often fall short in handling complex and uncertain scenarios. This is especially true when dealing with high-dimensional and nonlinear data, where the limitations of these methods become more apparent. These techniques are typically designed for more straightforward, predictable conditions, and struggle to adapt to the unpredictable nature of intricate systems and the vast array of variables involved. This creates challenges in accurately identifying and diagnosing faults in such environments. Recent advances in Artificial

Intelligence (AI) have prompted an increasing number of studies aimed at improving fault detection in space systems.

In this work we performed an in-depth study of the advantages of the Real NVP model and its integration with a self-supervision task, verifying that an enhancement with respect to the state-of-art is obtained.

[8] proposes applying a self-supervised learning task to fault detection within satellite Electrical Power Systems (EPS), using a dataset created in laboratory by NASA [9]. The approach involves permuting sensor channel measurements and tasking the network with identifying the correct permutation order. This encourages the model to learn meaningful representations that are more suited to detecting faults in EPS data.

Thus, our key contribution sees the unification of the Real NVP using the self-supervision training approach to perform fault identification in a satellite space system. We performed a deep ablation study testing our fault detection framework using three different datasets, all publicly available non-simulated space datasets to the best of our knowledge. One of these three datasets is the very recently released European Space Agency Anomaly Detection Benchmark (ESA-ADB) [10], an aerospace dataset composed of several months of data coming from three different missions. So far, this dataset is rarely used and tested in the literature due to its youth. The other two datasets used are the the Soil Moisture Active Passive (SMAP) satellite and the Mars Science Laboratory (MSL) rover datasets [11] and the ADAPT dataset [9] that are very used in this application field. The ablation study involved a detailed study of how training is influenced by introducing faulty data in the self-supervision and main losses, and testing different framework configurations to understand their advantages and disadvantages. Finally, we tested the inference latency of the models on edge devices with computational resources comparable to those of On-Board Computers (OBCs), and AI accelerators, used in small satellites, to test the feasibility of implementing them into embedded systems, obtaining very good results.

To summarize, the key contributions of our paper with respect to the state-of-art are:

1. we propose a novel framework that combines the strengths of the Real NVP with the self-supervised training approach, processing the input features to enhance the understanding of hidden data patterns. The training is performed by combining the self-supervision loss with the log-likelihood loss after the Real NVP network.

2. we deeply tested our framework in all real-world and laboratory space datasets available online: the very recent ESA-ADB dataset [10], and the frequently used ADAPT [9] and SMAP-MSL [11] in order to have the proper comparison with respect to the literature in this field and about this fault detection topic;
3. we carefully analyzed the effect of different training configurations in order to highlight potential advantages and drawbacks on using our framework;
4. we assess the latency of the models on various relevant edge devices to simulate and demonstrate their practical application in real space environments.

With these contributions, this paper aims to show how a custom framework based on Normalizing Flow - Real NVP model performs with respect to the state-of-art, obtaining very good results, with an improvement of up to 31% on the recent ESA-ADB dataset. Moreover, the results obtained would also be evaluated from the computational point of view, demonstrating not only new best results in the literature, but also a concrete feasibility in deploying these models in satellite embedded devices with an inference latency of 2.5ms.

The rest of the paper is organized as follows: Chapter 2 gives an overview of the related works. Chapter 3 provides a detailed explanation of the proposed model with the loss functions used. Chapter 4 provides an overview of the ablation study performed describing all the configurations tested. Chapter 5 describes the datasets used to evaluate the framework. In Chapter 6, we present a comprehensive discussion of the obtained results. Finally, Chapter 7 offers concluding remarks on the outcomes and implications of applying our framework.

2. Related Works

In the literature the fault detection topic is principally related to the industrial [12, 13], automotive [14], energy [15], and space environment [16, 10, 17, 18]. As it is possible to imagine, labeled data in the space environment is extremely costly and rarely publicly available due to intellectual property and security concerns.

Unsupervised learning offers promising solutions taking into account this problem, such as [12] or [13] that have been used in the past for fault detection

tasks on unlabeled datasets to identify deviations from nominal conditions. While unsupervised techniques offer advantages such as scalability and independence from labeled data, they often exhibit lower accuracy in fault detection, potentially resulting in more false positives during inference.

A more recent approach is the Physics-Informed Real NVP model, introduced in [18], which integrates domain-specific knowledge with the capability to model complex data distributions to solve the highlighted problems of unsupervised approaches. This model incorporates physical principles relevant to the ADAPT dataset [9] into its loss function, allowing it to capture fundamental physical interactions within the data. By leveraging normalizing flows [19, 20] and affine coupling layers [21], it efficiently manages high-dimensional data distributions, making it particularly effective for analyzing complex, multivariate satellite datasets.

Self-supervision has gained popularity due to its ability to learn meaningful representations that improve fault detection [22]. Self-supervised techniques typically rely on pretext tasks, which generate supervisory signals through artificial objectives. These tasks generally fall into two categories: contrastive and generative approaches [23]. For instance, contrastive learning methods, such as SimCLR [24], teach models to distinguish between similar and dissimilar data points, while generative approaches, like Masked Autoencoders [25], focus on reconstructing data from corrupted inputs, enabling models to learn key underlying features.

The success of self-supervised techniques depends on designing tasks that align with the structure of the data. A preliminary application of the self-supervision with Real NVP is introduced in [8]. In other fields, for example, self-supervised tasks have been applied through puzzle-solving for object recognition [26], or augmentation-based tasks in GANs to improve representation learning [27].

However, time series data poses unique challenges for self-supervised methods due to its sequential nature and the need to model temporal dependencies. Traditional techniques often emphasize inter-sample relationships but may neglect intra-sequence patterns within individual time series. Recent innovations, such as SelfTime [28], aim to overcome this limitation by simultaneously capturing both inter-sample and intra-sequence patterns. This approach uses specialized modules to analyze relationships between different time series as well as time steps within a single series. [22] described several self-supervision tasks used for both univariate and multivariate time series analysis. For instance, [29] applies transformations such as permuta-

tion and scaling to sensor data, creating a training strategy to detect human activities, which can be adapted to similar tasks in satellite fault detection.

3. Methodology

In this section, we provide a detailed explanation of the neural network model configuration and the loss functions used. As mentioned earlier, the framework is composed of two different parts: a Real NVP model trained with also a self-supervision loss.

Real NVP: it is a normalizing flows model ([18], [30]) designed to capture complex data distributions by transforming their probability density through a series of invertible mappings. Real NVP effectively learns high-dimensional, intricate distributions by mapping data to a simpler latent space. Unlike other normalizing flows models, it employs affine coupling layers to capture local relationships and diverse data modes, enabling both fast inference and efficient sample generation. In each coupling layer, a subset of variables is transformed using neural networks as invertible functions, while the remaining variables remain unchanged. Each coupling layer consists of a scale and a translation neural network, that use tanh and linear activation functions respectively. These networks take half of the Real NVP signals as input and apply transformations to the other half (Fig. 1).

Self-supervision model: self-supervision [31] is a recent AI technique able to capture important data features of an unlabeled dataset using the unlabeled data themselves as label instead of having a real labeled dataset. In this work, we use the paradigm of self-supervision to train our model permuting

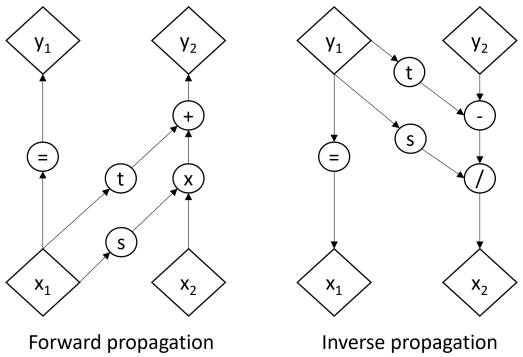

Figure 1: Representation of a coupling layer [30] and of its two directions of inference.

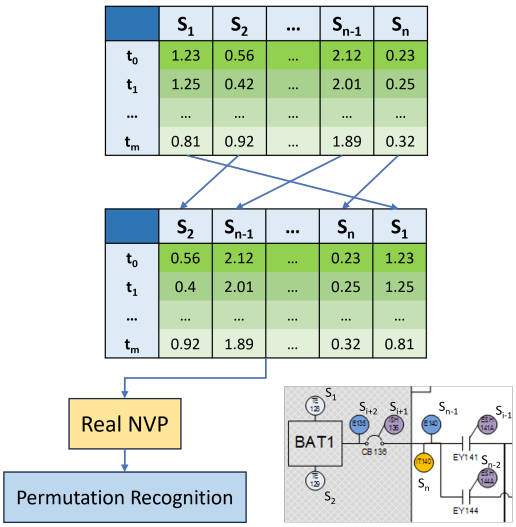

Figure 2: Self-supervised loss: the dataset's columns are permuted and the Real NVP model is trained to predict the correct permutation. As shown in the image patch extracted from the ADAPT circuit, each column is associated with a given sensor of the testbed. [8]

the original features of the dataset itself.

The aim of this work involves the integration of the Real NVP model with the self-supervision task taking into account both losses during training to outperform the baseline in terms of characteristics and performance. This additional self-supervised loss is designed to enhance the training of the Real NVP model by guiding it to learn features that improve fault detection capabilities. A column permutation technique is applied to implement the self-supervised loss, where some columns in the dataset are swapped. In particular, the self-supervision loss is computed by permuting the order of the dataset's columns $(S_1, S_2, ..., S_n)$ (see Fig. 2), and asking the model to predict which permutation has been applied. Indeed, the Fully-Connected (FC) layer (F) has a soft-max activation function and as many neurons as the number of permutations. This classifier allows the computation of the permutation index among a given set. The method used to select the permutations is well explained in Chapter 5.

During inference, the framework identifies system faults using the trained Real NVP model (R) and a Gaussian Multivariate layer (G). This last layer assigns a nominality score that quantifies the similarity of data processed by the Real NVP model to the training data, which is expected to be composed

only of nominal samples, or of a minimal quantity of faulty data. If this assumption holds, lower scores suggest a strong similarity to nominal data, whereas higher scores indicate potential faults in the tested samples.

The two loss functions used are the following:

- a self-supervision loss, already introduced in [8], with the following shape:

$$L_{\text{self_sup}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{p=1}^{P} y_{i,p}\log(s)$$
$$s = \frac{\exp(F(R(x_{i,p})))}{\sum_{j=1}^{P}\exp(F(R(x_{i,j})))} \tag{1}$$

  where N is the number of samples, P is the number of permutations, $y_{i,p}$ is the label (1 if p is the correct permutation, 0 otherwise), $x_{i,p}$ is a vector composed of concatenating a number of rows of the permuted dataset that corresponds to the number of time steps considered, R is the Real NVP model and F the Fully Connected layer (see Fig. 3).

- a log-likelihood loss, often used in Real NVP models, with the following shape:

$$L_{\text{like}} = -\sum_{i=1}^{N} log(p_X(R(x_i))) \tag{2}$$

  where X is the data distribution, x_i is a sample, and p_X the probabilistic data distribution. The p_X distribution could be very complex to find, but the Real NVP and the gaussian multivariate layer transform it into a simpler distribution.

The baseline approach (shown in Fig.3) employs only a log-likelihood loss to train the Real NVP neural network for fault detection.

4. Ablation study

Several training configurations (see Fig.3) are designed and studied in order to understand which one behaves better. For each configuration we highlight the best result found during the experiments for the usage of faulty data.

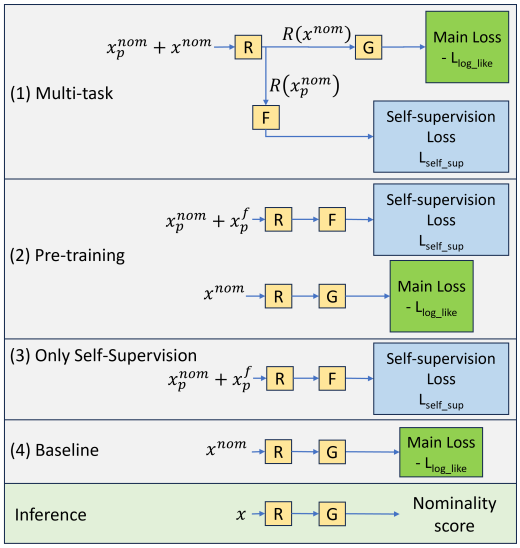

Figure 3: R, F, and G represent the model, the Fully Connected, and the Gaussian distribution layer respectively. p is the permutation applied to the input and used in the self-supervised loss, ($L_{\text{self_sup}}$). The main loss is equivalent to the default loss for Real NVP, ($-L_{\text{log_like}}$). In settings (2) and (3) both nominal (x_p^{nom}) and fault data (x_p^f) were used. For the main loss only nominal data (x^{nom}) was given as input. The baseline is represented by the architecture explained in [18].

- *Multi-task:* In this configuration, the Real NVP model is trained using a combination of self-supervised loss and log-likelihood loss. Only nominal data (fault-free data) are used during training (Chapter 6). The nominal input is first processed by the Real NVP model. The transformed nominal output is then passed through the Gaussian layer, while the permuted version of the data is fed into the FC layer (F). Backpropagation is performed using both loss functions simultaneously to optimize the model. This is expected to improve training efficiency;

- *Pre-training:* In this configuration, the Real NVP model is first pre-trained using only the self-supervised loss. Unlike the previous setup, the training can use either just nominal data or a mix of nominal and faulty data (Chapter 6). After this pre-training, the model is trained again separately using only nominal data with the log-likelihood loss. The pre-training phase helps the model capture essential features, while the subsequent training with log-likelihood loss refines the model's ability to estimate data probabilities;

- *Only Self-Supervision:* In this configuration, the Real NVP model is trained exclusively with the self-supervised loss. As in the previous setup, the input can consist of either nominal data or a mix of nominal and faulty data (Chapter 6). The log-likelihood loss is not used in this configuration. The Gaussian Multivariate layer is only used during inference to calculate the nominality score. This approach does not require labeled data, making it more flexible in scenarios where labeled data is scarce or unavailable, but it might not achieve the same level of accuracy as configurations that include log-likelihood loss;

- *Baseline:* The baseline configuration, based on [18], does not include the self-supervised loss function. In this setup, the Real NVP model is trained exclusively with the main log-likelihood loss function using nominal data only (Chapter 6).

Differently from what was done in [18] and [8], we decided to avoid training with a physics-informed loss, so as to perform a set of ablation studies on an approach that could be more easily adopted and generalizable, given the scarcity of publicly available non-anonymized data (e.g. both [11] and [10] provide anonymized data).

5. Experimental setups

In this section we deeply describe all the experimental setups tested (i.e. including the dataset used, the employed architecture and the metrics analyzed), and the hardware used for obtaining the findings of the Results section.

5.1. Datasets

We aim to evaluate our framework on several publicly available datasets related to the space domain, avoiding simulated datasets. To the best of our knowledge, the space datasets available in the literature include ADAPT, ESA-ADB, and SMAP & MSL. For the first two datasets, we were able to test the complete framework across all configurations. However, for the SMAP & MSL dataset, only the log-likelihood loss could be applied due to anonymized and unsynchronized telemetry data. In Table 1 we present a summary of the relevant information for each dataset, while in Table 2 are summarized the best hyper-parameters found for each model.

Table 1: Dataset Information: Relevant information for each dataset. The "x1" added in the "#Features" column serves the purpose of showing that the features of SMAP & MSL should be treated as univariate datasets. The values presented for ADAPT, SMAP and MSL should be considered as means of the individual splits, in ADAPT, and univariate features, in SMAP and MSL. The training percentage shows how many samples we used for the training over the total number of samples available in the dataset.

Dataset	Missions # Splits	#Samples	%Train	#Features	%Faults Train/Test
ADAPT [9]	7	1,505	46.27	69	47.47/40.66
ESA-ADB [10]	-	90,483,379	1.61	-	2.49/1.92
	1	42,365,287	3.40	7	4.88/1.81
	2	48,118,092	0.04	12	0.74/2.02
SMAP & MSL [11]	-	694,846	27.85	-	3.90/12.42
	SMAP	562,800	24.02	55 x1	0.71/12.48
	MSL	132,046	44.16	27 x1	11.30/12.08

5.1.1. ADAPT

Description: a dataset created by NASA Ames Research Center [9] to evaluate EPS fault detection algorithms. We divided the original dataset into 7 splits, each of these is further divided into 70-30 training-test sets. Only 1/3 of each training set is taken into account for the evaluation to demonstrate the efficiency of our framework even in situations with a limited amount of data. We trained the self-supervised task using training data coming from all splits, excluding the test set of the current split.

Architecture: the implemented Real NVP model consists of four coupling layers, each containing two neural networks, with each network comprising three FC layers. The first two FC layers have 32 neurons, while the final layer contains as many units as the dimensionality of the input, which consists of 50 timestamps in our case. We used 8000 permutations in configurations "Multi-task" and "Pre-training", and 10000 in "Only self-supervision". Therefore, the number of units in the FC layer used for the self-supervision task, (F), is either 8000 or 10000.

Metrics: This metric allows for measuring the model's ability to distinguish between classes without requiring the selection of a specific threshold. Moreover, additional metrics used include the False Positive Rate at a 95%

Table 2: Architectures Summary: Best hyper-parameters found for each dataset and configuration.

Dataset	Config.	Coup. Layers	FC Layers	Units	Time Window
ADAPT	all	4	3	32	50
ESA-ADB 1	1,4	4	1	256	10
ESA-ADB 1	2,3	4	3	256	10
ESA-ADB 2	1,4	4	2	256	10
ESA-ADB 2	2,3	8	1	64	10
SMAP	-	4	6	64	10
MSL	-	6	3	64	10

true positive rate (FPR95), the F1-score, and the average precision. The threshold used to compute the F1-score and the average precision is the threshold at 95% true positive rate.

5.1.2. ESA-ADB

Description: [10] recently published a new dataset consisting of data collected from three missions. Two of these missions were used to conduct a set of experiments with several machine learning methods, employing different subsets of data for training. In particular, they:

1. used increasingly bigger datasets for training, referring to each stage as a "phase", to simulate the gradual generation of data over the course of a mission;
2. experimented with a few subsets of sensors, to reduce the amount of data and faults analyzed by the algorithms. This led them to use features 41-46 for mission 1 and features 18-28 for mission 2 when training with the smallest datasets.

We decided to focus our experiments using the "phase 1" data for training, which consists of 3 months of data for mission 1 and 1 month for mission 2. This is to simulate missions in which it would not be possible to update the weights of the Real NVP using new data. Given that the samples of the dataset are collected using varying sampling frequencies, we concatenated the time difference between two sequential observations to the selected features, to give time information to the neural network. Therefore, we have 6+1

Table 3: Parameters used for the threshold selection in ESA-ADB dataset. The batch size, the smoothing window size and the smoothing percentage are used to define the period of the moving average, the error buffer groups nearby errors, the window size is the time window used as input for the models, and p is the threshold for pruning anomalous sequences.

Parameter	Value
batch size	70
smoothing window size	30
window size	10
error buffer	100
smoothing percentage	0.05
min error value	0.007
p	0.005

features for Mission 1 and 11+1 features for Mission 2, as presented in Table 1.

Architecture: the Real NVP used with the ESA-ADB dataset was designed through a grid search, that led to different model's architecture depending on the mission and on the configuration. As shown in Table 2, where the configurations follow the numeration of Fig. 3: "Multi-task" (1), "Pre-training" (2), "Only self-supervision" (3), and "Baseline" (4). The number of units in the FC layer (F) used for the self-supervision task depends on the number of permutations. We used 2500/8000 permutations in configuration "Multi-task" and 5000/10000 permutations in both "Pre-training" and "Only self-supervision" configurations.

Metrics: we evaluated the models using both the AUROC, to disentangle the model evaluation from the threshold selection step, and F05 to facilitate the comparison with [10] and future works. The threshold used to evaluate the F05 score is computed using [11] method. It employs a dynamic thresholding approach to detect faults. Initially, the model predicts the next value and calculates the error by comparing it to the actual value, then smooths these errors to reduce noise. A dynamic threshold is selected to highlight significant deviations. Although our framework generates a nominality score instead of a prediction error, this dynamic thresholding principle can still be applied effectively to identify significant deviations in nominality scores. As a consequence, we used the same values of [10] for all parameters used to

compute the threshold, but "p", which is the minimum percentage decrease between maximum errors in faulty sequences, used to reduce closely spaced anomalous signal predictions. Its value was changed to 0.005 by looking at the differences between the nominality scores generated by our framework for nominal and faulty data, to reduce the effect of pruning. See Table 3.

5.1.3. SMAP & MSL

Description: these datasets are composed of data coming from the Soil Moisture Active Passive (SMAP) satellite and the Mars Science Laboratory (MSL) rover [11]. Unfortunately, both datasets consist of several anonymized telemetry columns that are not synchronized, making it unfeasible to apply multivariate approaches. This issue prevents the use of the self-supervision loss, since the meaning of each column and the correlation between them is unknown. However, we aim to evaluate our algorithm to demonstrate the improvements offered by our Real NVP framework, even without self-supervision, in comparison to the state-of-the-art methods. A model is created and trained for each column/sensor to accomplish the fault detection task. Each column represents the whole available dataset for each model. The dataset is already provided with the training/test split.

Architecture: The neural network architecture changes according to the mission, while, in both cases, the final FC layer is composed of 10 units, as this is the time window used as input.

Metrics: we aligned ourselves with recent works that tested AI methods in SMAP & MSL, using the F05 metrics to place a higher weight on precision, prioritizing the reduction of false positives. The final F05 score for each dataset is calculated by averaging the results across all of its sensors. The threshold used to classify data between nominal and faulty is computed using the same algorithm and parameters of [11].

5.2. Implementation Details

All models have been trained and evaluated, with 5 different random seeds, on a shared cluster of Intel Xeon Scalable Processors Gold 6130 and nVidia Tesla V100 or on a server with two GeForce RTX 2080 and a Intel Xeon Silver 4110 processor. Each training was performed keeping 30% of the training data as validation set and using it for early stopping and hyper-parameter selection. All datasets were scaled between 0 and 1, if needed. For each dataset, given its target size P, we randomly permuted the columns for a number of times greater than P and kept the P permutations that

lead to the greatest positional deviation from the features' true positions and distinctiveness from other permutations within the set. Specifically, we aimed to maximize 3.

$$D = \sum_{i,j=1,i\neq j}^{P} \sum_{k=1}^{n} |i_k - j_k| + \sum_{i=1}^{P} \sum_{k=1}^{n} |k - i_k| \tag{3}$$

where n is the number of sensors present in the dataset (Fig. 2), i and j indicate distinct permutations, i_k (j_k) represents the sensor's index moved in position k by permutation i (j).

5.3. Inference Latency

We tested the inference time of the Real NVP model on a AMD GX-412HC, which shares the characteristics of some of the SoCs used in the space sector as OBCs, and on a Jetson Nano, which has already been considered for use in satellites [32], both in the low power (5W) and in the nominal (10W) mode. Additionally, we tested the use of a Myriad X and a Coral as AI accelerator, both in their USB format. The former has been used in combination with the AMD GX-412HC, while the latter with the Jetson Nano. The two AI accelerators were used to perform inference with the neural network, while the log-likelihood computation was performed either on the AMD GX-412HC or on the Jetson Nano.

6. Results

Here we show and discuss the results obtained using the different experimental setups explained in the previous Section 5. The results shown here are obtained by setting the trained framework in inference mode, removing the FC layer (F) used for the self-supervision task from the networks, allowing for the direct use of the log-likelihood values outputted by the Real NVP model's distribution layer (G). We first present the results on ADAPT, then on the ESA Anomaly Detection Benchmark dataset, SMAP & MSL, and finally the inference latencies on edge devices.

6.1. ADAPT

In Table 4 and 5 we show the results obtained during the ablation study performed to understand the effect of using faulty data during training. In particular, the experiments shown in Table 4 are performed training the

Table 4: ADAPT Ablation Study: Results obtained training the models using both nominal and faulty samples for the self-supervision task and only nominal samples for the main loss.

Configuration	AUROC	FPR95	F1	Avg. Prec.
Baseline	82.50	36.09	86.52	78.64
Multi-task	79.06	63.51	82.02	70.59
Pre-Training	90.39	30.16	89.08	82.25
Only self-sup.	91.55	25.79	89.91	83.94

Table 5: ADAPT Ablation Study: Results obtained using faulty data in the "Baseline" configuration and not in the other configurations.

Configuration	AUROC	FPR95	F1	Avg. Prec.
Baseline	75.20	74.37	78.27	66.03
Multi-task	80.99	36.42	86.56	78.70
Pre-Training	82.21	52.99	82.49	72.09
Only self-sup.	82.93	51.96	83.30	72.86

self-supervision task, in all configurations, using a dataset composed of both nominal and faulty samples, while the main loss was computed considering only nominal data. Those showed in Table 5 are obtained using faulty and nominal data in the "Baseline" configuration, and only nominal data in all other configurations.

The results show that the presence of faulty samples in the training dataset hinders the model performances when used in combination with the main loss, as can be seen by looking at the "Baseline" and "Multi-task" experiments. Notating that, while AUROC, F1, and Average Precision should be as high as possible, FPR95 has the opposite behavior. However, it seems that introducing faulty data only during the training phase of the self-supervision task, i.e. configurations "Pre-Training" and "Only self-supervision", leads to improvements in all metrics. To the left of Fig. 4 are shown the self-supervision losses logged during training of the "Only self-supervision" configuration. It shows that the model is able to learn how to identify different columns even when some of them are affected by faults and thus contain samples that behave differently from their nominal case. Additionally, we see that by introducing faulty samples the loss needs fewer epochs to con-

Table 6: ADAPT Ablation Study: Wilcoxon test for results shown in table 4 and 5. For each configuration and metric we show the p-value. Values below 0.05 are considered statistically significant. We show in bold the results with p-values greater than 0.05.

Configuration	AUROC	FPR95	F1	Avg. Prec.
Baseline	1e-4	6e-9	5e-8	1e-7
Multi-task	**0.339**	6e-5	9e-6	8e-7
Pre-Training	7e-9	9e-9	9e-10	1e-8
Only self-sup.	1e-9	9e-11	3e-9	3e-9

Table 7: ADAPT Ablation Study: Results obtained using or not faulty data during training while keeping constant the size of the training dataset.

Configuration	AUROC	FPR95	F1	Avg. Prec.
Baseline	**81.52**	**37.97**	**85.92**	**77.74**
Baseline Faults	73.54	61.72	81.46	70.51
Multi-task	**87.46**	**29.25**	**88.56**	**82.08**
Multi-task Faults	70.63	96.10	75.09	60.42
Pre-Training	**93.36**	**22.82**	**91.45**	**86.04**
Pre-Training Faults	89.38	33.23	87.89	80.54
Only self-sup.	**92.45**	**25.90**	**90.15**	**84.06**
Only self-sup. Faults	89.46	33.22	87.96	80.64

verge.

To check whether these results are statistically significant we performed a Wilcoxon test, which shows how much two distributions are different from each other, highlighting a significant difference when the p-value is higher than 0.05. We did this for all metrics of each configuration in Table 4 and Table 5. An example are "Baseline" in Table 4 and "Baseline" in Table 5. We obtained p-values close to 0 for all configurations and metrics but "Multi-task / AUROC", that led to a p-value higher than 0.05 and should therefore be considered not significant (see Table 6).

To test whether the above-mentioned findings are due to specific information brought by the faults or rather by the increase in the training dataset size, we ran a second set of experiments in which we removed part of the nominal data when introducing faulty samples. These tests are shown in Table 7, where the configurations with "Faults" refer to the experiments in

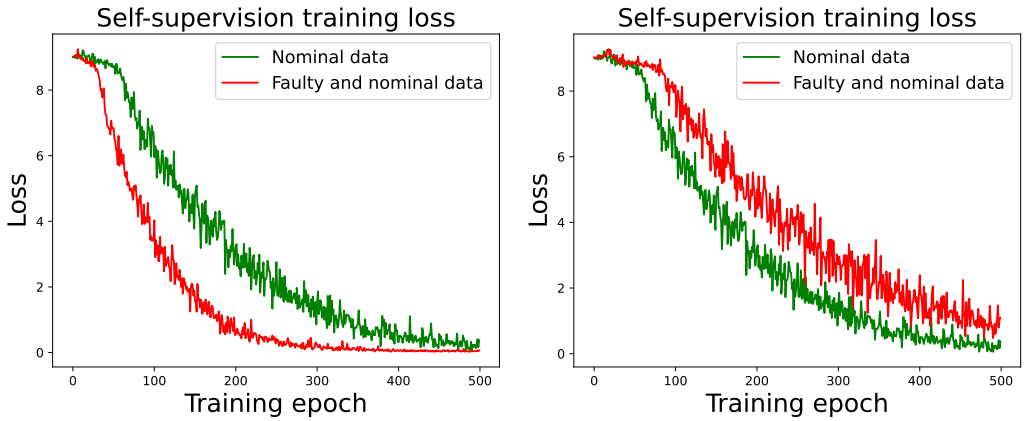

Figure 4: Losses over 500 training steps epochs when training using only nominal data and when introducing faulty samples while keeping a constant training set size (right) or not (left).

which we removed from the training data as many nominal samples as the number of faulty samples introduced. The remaining configurations use only nominal data.

Additionally, in Fig. 4 to the right are shown the losses of two different trainings performed by keeping the dataset size constant when introducing faulty samples. Differently from the case in which the dataset increases in size, shown to the left of the figure, when keeping the number of training samples constant the models trained with faulty data require more time to converge with respect to those trained using only nominal data.

Moreover, given that the experiments with faulty data led to worse results with respect to those that used only nominal samples, we can conclude that the performance improvements shown in Table 4 are due to the use of a bigger dataset. However, we found interesting that the self-supervision task suffers less from the presence of faults (configurations "Pre-training" and "Only self-supervision"), as this could be considered a favorable factor toward its usage in transfer learning phases, either on-board or on the ground on bigger unannotated datasets.

The small effect that the introduction of faults has on these two configurations is demonstrated also by the p-values obtained with the Wilcoxon test, shown in Table 8, which returned not significant results for the "Only self-supervision" configuration.

In light of this, we analyzed the effect that faults in the test set have on

Table 8: ADAPT Ablation Study: Wilcoxon test for results shown in table 7. For each configuration and metric we show the p-value. Values below 0.05 are considered statistically significant. We show in bold the results with p-values greater than 0.05.

Configuration	AUROC	FPR95	F1	Avg. Prec.
Baseline	2e-6	8e-5	4e-4	2e-4
Multi-task	1e-9	3e-11	6e-11	3e-11
Pre-Training	8e-4	7e-4	1e-4	4e-4
Only self-sup.	0.013	0.041	0.034	**0.054**

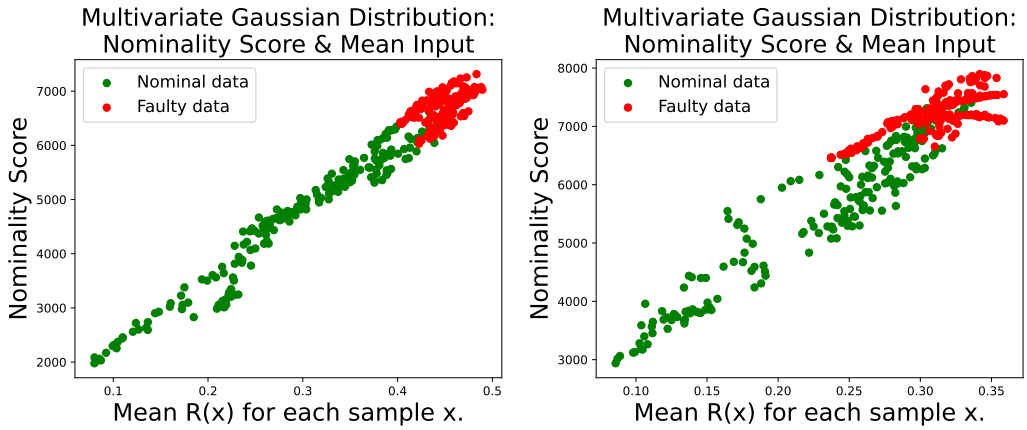

Figure 5: Relationship between the mean of the data transformed by the Real NVP and the nominality score outputted by the Gaussian distribution when training using both faulty and nominal data (left) or only nominal data (right).

the output distribution when training with or without faulty data, without removing any nominal samples, in the "Only self-supervision" configuration. In Fig. 5 for each sample we show the relationship between the mean of the transformed data R(x), on the x-axis, i.e. the output of the Real NVP neural network used as input for the Multivariate Gaussian distribution appended to the NN, and the nominality score predicted by distribution, on the y-axis. To the left of the figure we show the results obtained when training a model using both nominal and faulty data, while to the right are shown the results obtained training with only nominal data. When introducing faulty data without removing any samples from the original dataset the self-supervision task is able to separate nominal and faulty data in distinct, though not perfectly separated, clusters without being explicitly designed with that goal

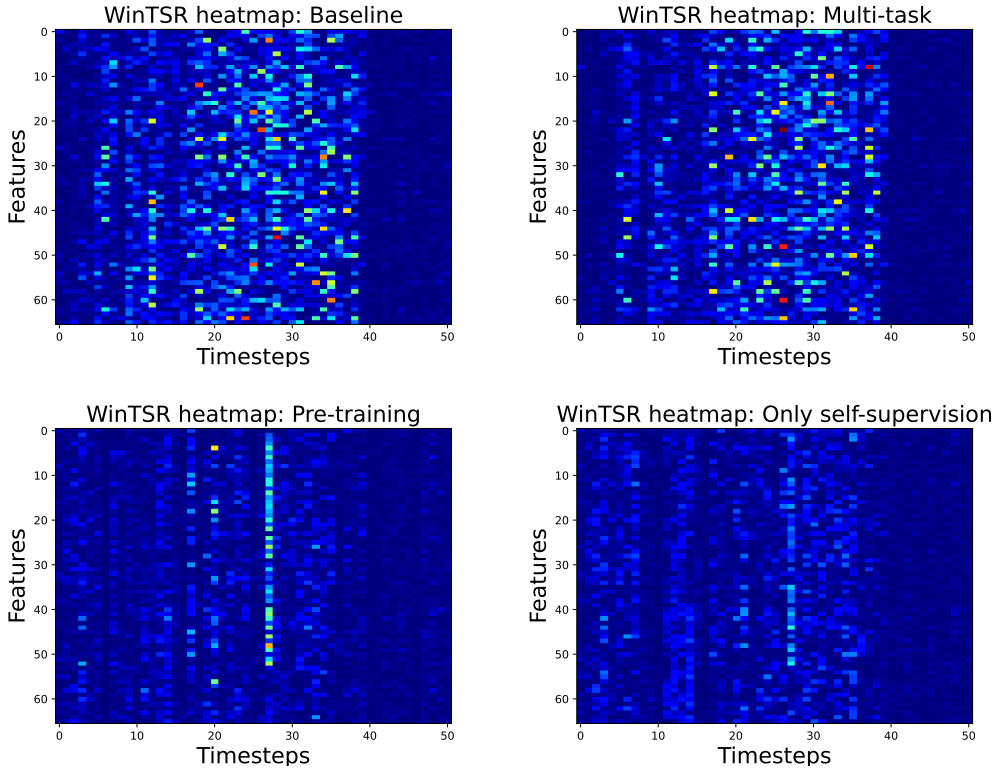

Figure 6: WinTSR averaged over nominal data for each configuration.

in mind. When using only nominal data during training the separation is less pronounced, as shown to the right of Fig. 5.

The analysis of all the previous findings led us to propose the framework shown in Fig. 3, where for each configuration we display how faulty data could be used for training, and in which configurations it would be best to have only nominal data. To better understand the impact of the self-supervision task, we used a recently proposed method for time series interpretability, Window Temporal Saliency Rescaling (WinTSR) [33], which aims to improve over previous approaches by scaling the features importance by the importance of the respective timesteps.

The results, shown in Fig. 6, are obtained by computing the average of the WinTSR in each configuration for the nominal samples. The "Baseline" and "Multi-task" configuration tend to give more importance to certain features and timesteps. This may hinder their performance when the anomalies are

Table 9: ESA-ADB Results for Mission 1.

Configuration	AUROC	F05
Baseline [18]	59.88	28.62
Multi-task	65.73	**29.82**
Pre-Training	**71.44**	20.34
Only self-sup	70.47	20.62
Mean ESA-ADB [10]	-	3.10
Best ESA-ADB [10]	-	22.70

Table 10: ESA-ADB Results for Mission 2.

Configuration	AUROC	F05
Baseline [18]	66.93	**63.80**
Multi-task	**67.91**	63.37
Pre-Training	54.37	41.47
Only self-sup	54.38	41.57
Mean ESA-ADB [10]	-	8.08
Best ESA-ADB [10]	-	62.20

linked to other features, while the remaining configurations are able to attend to all features equally.

6.2. ESA-ADB

Here are shown the results obtained when using a Real NVP and the self-supervision loss on two missions of the ESA Anomaly Detection Benchmark dataset [10]. We decided to present them in two separate tables to aid the visual inspection. Following the results obtained during the ablation study on ADAPT, we used faulty and nominal data as shown in Fig. 3: faulty data was introduced only in configurations "Pre-training" (2) and "Only self-supervision" (3).

In Table 9 are laid out the results obtained for Mission 1, while we show those of Mission 2 in Table 10. Given that [10] published a benchmark using several algorithms, we decided to show in our tables the mean and the best results obtained by them, to prevent the inclusion of extensive tables primarily consisting of previously published data. Our results show improved performances over all methods tested by [10].

Table 11: SMAP & MSL Results.

Model	F05 SMAP	F05 MSL	F05
Real NVP	**95.09**	87.17	**92.25**
GNN [17]	-	-	89.00
FeatExt + Class [16]	88.50	**87.30**	88.00

When considering the relative performances of our configurations, in Mission 1 the AUROC patterns are similar to those already seen in ADAPT, with the self-supervision task aiding the model and leading to increased performances, achieving the best results in the "Pre-Training" task. However, when looking at the F05, the results improve only in the "multi-task" configuration. Because of this finding, we believe that more work should be done toward the development of methods to estimate the threshold, which could also vary based on the considered configuration.

Finally, the results shown in Table 10 present a different behavior, here the use of the self-supervision task in configurations "Pre-training" and "Only self-supervision" hinders the performances of the Real NVP model, while in configuration "Multi-task" it leads to small improvements in the AUROC score and a limited decrease of the F05 metric. We believe that this is due to the low number of possible permutations and the low percentage of faults in the training dataset, as shown in Table 1, which limits the advantages of a grid search over the hyper-parameters. Furthermore, the presence of different sampling frequencies increases the complexity of this dataset.

6.3. SMAP & MSL

In Table 11 are presented the results obtained on the SMAP & MSL datasets by the Real NVP model, trained using only the main loss, and recent approaches that reached state-of-the-art results and have been specifically created for spacecraft applications. One, which we're going to call "GNN", [17] uses a Graph Neural Network (GNN) to sequentially predict the future sample, computes an estimation error as the difference between the predicted and the seen sample, and finally analysis the error signal to separate nominal and faulty data. The threshold is estimated through the method introduced by [11], as it happens for our results. Whereas [16] extracts 3 features and then uses a classifier trained in a supervised fashion to predict whether a sample contains a fault. We decided to refer to it as "FeatExt + Class".

Table 12: Inference latency tests on different edge devices. For each device, we show the average latency, in milliseconds, over 1000 inferences.

Device	Mean Latency [ms]
AMD GX-412HC	7.75
Myriad + AMD GX-412HC	2.50
Jetson Nano 5W	9.18
Jetson Nano 10W	5.01
Coral + Jetson Nano 5W	9.33
Coral + Jetson Nano 10W	5.88

The literature often aggregates the results obtained in the two datasets, but sometimes shows also the performances achieved on the individual datasets. We decided to show both the aggregated and the individual results when available, as already done by [16], to facilitate future comparisons. The aggregated score is computed as a weighted sum of the individual results (see 4), as already done in [16].

$$F05 = 0.653 \cdot F05_{SMAP} + 0.346 \cdot F05_{MSL} \tag{4}$$

The Real NVP led to improved results on the SMAP dataset while achieving comparable performances on MSL. This led to an increase of approximately 3% over the state-of-the-art on the aggregated score and of 6.6% on SMAP, while on MSL we achieved performances comparable to those of [16].

6.4. Inference Latency

In Table 12 we present the mean latency, over 1000 tests, of a Real NVP model when performing inference on two edge devices used in various settings. We obtained the fastest inferences by running the neural network on a Myriad USB stick and the log-likelihood computation on an AMD GX-412HC. It should also be noted that the tests performed using the Jetson Nano, both in high and low power mode, showed that the introduction of a Coral USB stick leads to slightly worse performances in terms of latency. We believe that this is due to the fact that the AI accelerator does not lead to notable inference speed-ups with respect to the Jetson Nano GPU with this model, and that consequently the results are affected by the USB data I/O time. This behavior is not observed in the tests conducted on the AMD GX-412HC, as the neural network inference is executed on the CPU due to the absence of a GPU.

7. CONCLUSION

We present a comprehensive analysis of the Real NVP neural network used for fault detection in the space domain, and of its use in combination with a self-supervised task based on feature permutations, when dealing with multivariate time series. The experiments are performed on several randomly generated custom splits of the ADAPT dataset, on the SMAP & MSL telemetry channels, and on the recently published ESA Anomaly Detection dataset. Finally, inference latency tests are performed on different combinations of SoCs and AI accelerators compatible with the resources available on small satellites.

We showed that the considered self-supervision task can be used even with unlabeled datasets with minor performance drawbacks with respect to semi-supervised approaches. We believe this to be an important result to enable re-training either on-board or on the ground with large unlabeled datasets. These features could be extremely useful to facilitate the use of AI on long missions. Additionally, we presented new state-of-the-art results on the univariate datasets SMAP & MSL and on the recently published ESA-ADB dataset using a semi-supervised approach. We believe that the results obtained justify further experiments. In particular, we think that an interesting direction for new experiments should be the search for better threshold estimates based on the configuration and algorithm considered, as our results suggest that the capabilities of the proposed solutions are not completely exploited.

Moreover, we showed that the proposed self-supervision task leads to improved performances with respect to the "Baseline" configuration in Mission 1 of the ESA-ADB dataset, thus extending its applicability to real-world scenarios. We showed that the Real NVP inference latency on several combinations of SoCs and AI accelerators is compatible with real-time scenarios, a necessary condition for it to be taken into consideration for on-board implementation on power-constrained spacecraft.

Finally, currently Transformers achieve state-of-the-art approaches for time-series analysis in several fields, but their efficient implementation in low-power edge devices is still under research [34]. Variational Autoencoders (VAEs) performance varies across dataset, these models have previously been used for all considered datasets: on ADAPT [18] and ESA-ADB [10] they achieve lower performances with respect to our approach; on SMAP&MSL they achieve comparable results when used in combination with GNNs, as

showed in [35]. Additionally, we believe that the use of a Real NVP model in combination with a physics-informed loss, when suitable dataset's information is available, can lead to improved performance and explainability.

We believe that future studies could work on the threshold selection step, as the AUROC results achieved with the ESA-ADB dataset reveal that the F05 score may be improved. Moreover, we believe it could be interesting to place a stronger emphasis on the fault isolation step and toward predicting faults in advance, which would require a different set of metrics.

ACKNOWLEDGMENT

This work has been developed with the contribution of the Politecnico di Torino Interdepartmental Centre for Service Robotics (PIC4SeR) (https://pic4ser.polito.it). Computational resources were provided by HPC@POLITO (http://hpc.polito.it).

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT and Copilot in order to avoid syntactical errors and improve readability. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

CRediT authorship contribution statement

Carlo Cena: Conceptualization, Investigation, Software, Writing. **Umberto Albertin**: Validation, Writing, Visualization. **Silvia Bucci**: Supervision. **Mauro Martini**: Writing, Supervision. **Alessandro Balossino**: Funding acquisition, Project administration, Resources. **Marcello Chiaberge**: Funding acquisition, Project administration, Resources.

References

[1] G. W. Hein, Status, perspectives and trends of satellite navigation, Satellite Navigation 1 (1) (2020) 22.

[2] M. Martini, V. Mazzia, A. Khaliq, M. Chiaberge, Domain-adversarial training of self-attention-based networks for land cover classification using multi-temporal sentinel-2 satellite imagery, Remote Sensing 13 (13) (2021) 2564.

[3] S. Campanella, J. Harrington, Satellite communications networks, Proceedings of the IEEE 72 (11) (1984) 1506–1519. doi:10.1109/PROC.1984.13046.

[4] E. Dotto, A. Zinzi, Impact observations of asteroid Dimorphos via Light Italian CubeSat for imaging of asteroids LICIACube, Nature Communications 14 (1) (2023) 3055.

[5] Y. Xu, X. Kong, W. Ma, Z. Zhu, J. Zhang, Integrated System Health Management Research and Application on Satellite, in: 2021 Global Reliability and Prognostics and Health Management (PHM-Nanjing), 2021, pp. 1–5.

[6] J. Schein, S. T. Bushby, N. S. Castro, J. M. House, A rule-based fault detection method for air handling units, Energy and buildings 38 (12) (2006) 1485–1492.

[7] R. Isermann, Model-based fault-detection and diagnosis–status and applications, Annual Reviews in control 29 (1) (2005) 71–85.

[8] C. Cena, S. Bucci, A. Balossino, M. Chiaberge, A Self-Supervised Task for Fault Detection in Satellite Multivariate Time Series, in: Proceedings of the 1st SPAICE Conference on AI in and for Space, Zenodo, 2024, pp. 292–296. doi:10.5281/zenodo.13885595.

[9] O. J. Mengshoel, A. Darwiche, K. Cascio, M. Chavira, S. Poll, N. S. Uckun, Diagnosing Faults in Electrical Power Systems of Spacecraft and Aircraft, in: D. Fox, C. P. Gomes (Eds.), Proceedings of the Twenty-Third Conference on Artificial Intelligence, AAAI Press, 2008, pp. 1699–1705.

[10] K. Kotowski, C. Haskamp, J. Andrzejewski, B. Ruszczak, J. Nalepa, D. Lakey, P. Collins, A. Kolmas, M. Bartesaghi, J. Martinez-Heras, G. De Canio, European Space Agency Benchmark for Anomaly Detection in Satellite Telemetry, arXiv preprint (2024). doi:10.48550/arXiv.2406.17826.

[11] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, T. Soderstrom, Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, arXiv preprint arXiv:1802.04431 (2018).

[12] U. Albertin, G. Pedone, M. Brossa, G. Squillero, M. Chiaberge, A real-time novelty recognition framework based on machine learning for fault detection, Algorithms 16 (2) (2023). doi:10.3390/a16020061.

[13] A. Priarone, U. Albertin, C. Cena, M. Martini, M. Chiaberge, Unsupervised novelty detection methods benchmarking with wavelet decomposition (2024). arXiv:2409.07135.

[14] K. Jeong, S. B. Choi, H. Choi, Sensor fault detection and isolation using a support vector machine for vehicle suspension systems, IEEE Transactions on Vehicular Technology 69 (4) (2020) 3852–3863. doi:10.1109/TVT.2020.2977353.

[15] G. Helbing, M. Ritter, Deep learning for fault detection in wind turbines, Renewable and Sustainable Energy Reviews 98 (2018) 189–198. doi:https://doi.org/10.1016/j.rser.2018.09.012.
URL https://www.sciencedirect.com/science/article/pii/S1364032118306610

[16] S. Cuéllar, M. Santos, F. Alonso, E. Fabregas, G. Farias, Explainable anomaly detection in spacecraft telemetry, Engineering Applications of Artificial Intelligence 133 (2024) 108083. doi:https://doi.org/10.1016/j.engappai.2024.108083.

[17] Graph Neural Networks for Anomaly Detection in Spacecraft, Zenodo, 2024. doi:10.5281/zenodo.13885527.

[18] C. Cena, U. Albertin, M. Martini, S. Bucci, M. Chiaberge, Physics-informed real nvp for satellite power system fault detection, in: 2024 IEEE International Conference on Advanced Intelligent Mechatronics (AIM), 2024, pp. 679–684. doi:10.1109/AIM55361.2024.10636990.

[19] E. G. Tabak, E. Vanden-Eijnden, Density estimation by dual ascent of the log-likelihood, Communications in Mathematical Sciences 8 (1) (2010) 217–233.

[20] I. Kobyzev, S. J. Prince, M. A. Brubaker, Normalizing Flows: An Introduction and Review of Current Methods, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (11) (2021) 3964–3979.

[21] L. Dinh, D. Krueger, Y. Bengio, Nice: Non-linear independent components estimation, arXiv preprint arXiv:1410.8516 (2014).

[22] K. Zhang, Q. Wen, C. Zhang, R. Cai, M. Jin, Y. Liu, J. Y. Zhang, Y. Liang, G. Pang, D. Song, et al., Self-supervised learning for time series analysis: Taxonomy, progress, and prospects, IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

[23] Z. Liu, A. Alavi, M. Li, X. Zhang, Self-Supervised Learning for Time Series: Contrastive or Generative?, arXiv preprint (2024). doi:10.48550/arXiv.2403.09809.

[24] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A Simple Framework for Contrastive Learning of Visual Representations, in: Proceedings of the 37th International Conference on Machine Learning, Vol. 119 of Proceedings of Machine Learning Research, PMLR, 2020, pp. 1597–1607.

[25] C. Zhang, C. Zhang, J. Song, J. S. K. Yi, I. S. Kweon, A Survey on Masked Autoencoder for Visual Self-supervised Learning, in: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI), International Joint Conferences on Artificial Intelligence Organization, 2023, pp. 6805–6813, survey Track.

[26] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, T. Tommasi, Domain generalization by solving jigsaw puzzles, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2229–2238.

[27] L. Hou, Q. Cao, Y. Yuan, S. Zhao, C. Ma, S. Pan, P. Wan, Z. Wang, H. Shen, X. Cheng, Augmentation-Aware Self-Supervision for Data-Efficient GAN Training, in: Advances in Neural Information Processing Systems (NeurIPS), Vol. 36, Curran Associates, Inc., 2023, pp. 31601–31620.

[28] H. Fan, F. Zhang, Y. Gao, Self-supervised time series representation learning by inter-intra relational reasoning, arXiv preprint arXiv:2011.13548 (2020).

[29] A. Saeed, T. Ozcelebi, J. Lukkien, Multi-task self-supervised learning for human activity detection, in: Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Vol. 3, ACM New York, NY, USA, 2019, pp. 1–30.

[30] L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using Real NVP, in: 5th International Conference on Learning Representations (ICLR), OpenReview.net, 2017.

[31] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, J. Tang, Self-supervised learning: Generative or contrastive, IEEE Transactions on Knowledge and Data Engineering 35 (1) (2023) 857–876. doi:10.1109/TKDE.2021.3090866.

[32] L. Diana, P. Dini, Review on hardware devices and software techniques enabling neural network inference onboard satellites, Remote Sensing 16 (21) (2024). doi:10.3390/rs16213957.

[33] M. K. Islam, J. Fox, Wintsr: A windowed temporal saliency rescaling method for interpreting time series deep learning models, arXiv preprint arXiv:2412.04532 (2024).

[34] J. Lee, I. Bahk, H. Kim, S. Jeong, S. Lee, D. Min, An autonomous parallelization of transformer model inference on heterogeneous edge devices, in: Proceedings of the 38th ACM International Conference on Supercomputing, ICS '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 50–61. doi:10.1145/3650200.3656628.

[35] W. Li, W. Hu, T. Chen, N. Chen, C. Feng, Stackvae-g: An efficient and interpretable model for time series anomaly detection, AI Open 3 (2022) 101–110. doi:https://doi.org/10.1016/j.aiopen.2022.07.001.
URL https://www.sciencedirect.com/science/article/pii/S2666651022000110