SSC24-IV-01

Ensemble Learning for Autonomous Onboard Satellite Fault Diagnosis with Validation Tool

Madeline Anderson
Massachusetts Institute of Technology
Cambridge, MA; 585-483-9984
mloui@mit.edu

Jeremy Muesing
Auria Space
Boulder, CO; 303-916-3263
jeremy.muesing@auria.space

Kerri Cahoy
Massachusetts Institute of Technology
Cambridge, MA; 650-814-8148
kcahoy@mit.edu

Kenneth Center
Auria Space
Boulder, CO; 240-391-3310
ken.center@auria.space

ABSTRACT

In-orbit events and onboard malfunctions, often manifesting as telemetry faults, can compromise satellite missions. Monitoring such faults through ground operations is time intensive and tedious, motivating the need for advanced onboard and autonomous satellite resilience capabilities, which will increase the likelihood of mission success. We present a novel approach for onboard autonomous satellite fault diagnosis that leverages ensemble learning to jointly detect faults and attribute them to a probable cause. The framework consists of an ensemble of representation learners, including an Autoencoder (AE), Kalman Filter (KF), Gaussian Mixture Model (GMM), Long Short Term Memory network (LSTM), and the PCMCI causal discovery algorithm, which extract informative data representations from satellite telemetry data. Then, using a decision tree classifier called XGBoost, we detect and classify faults based on the fused representations. The implementation is modular and can easily integrate into a larger fault response and decision-making system.

In the development of our prototype, we recognized that the scarcity of historical faulting telemetry data and detailed ground truth labels present major limitations to the improvement of autonomous fault diagnosis approaches, both in terms of the ability to train machine learning algorithms and to provide quantitative and comparative validation of methods. To address this limitation, we develop a novel statistical telemetry simulation tool, called SatFaultSim, that generates non-faulting and faulting data for the training and validation of fault detection and attribution algorithms. SatFaultSim can model 11 common fault cases, such as ionizing radiation faults, system resets, thermal and current faults. Each case statistically emulates fault scenarios observed during past satellite missions. The tool generates faults based on user input through configuration files and is extendable to accommodate additional fault cases.

We use SatFaultSim to train and validate our ensemble learning approach to jointly demonstrate the capability of the simulation tool and the effectiveness of our algorithmic techniques for satellite fault diagnosis. We simulate a 10-orbit dataset of faulting and non-faulting telemetry samples, split into training and validation subsets. After training the machine learning models, we perform an in-depth evaluation resulting in highly promising results, with an overall combined fault detection and attribution accuracy of 99.89%,

detection accuracy of 99.94%, and average fault attribution accuracy of 99.97%. The false positive and negative rates are very low for each fault type, all falling under 0.013%. These metrics show that our approach is highly capable of identifying and attributing known faults and can serve as a baseline for autonomous fault diagnosis methods. The algorithmic framework and simulation tool are implemented in Python and available on Github[†]. Moving forward, our work will facilitate further improvements to autonomous fault resilience, such as the integration of non-parametric and un-supervised learning techniques to accommodate unseen and rare fault types. Our approach can also enhance an autonomous decision-making framework by informing an onboard fault response or mitigation system.

INTRODUCTION

Satellites are launched in increasing numbers each year, offering critical services in communications, navigation, and Earth observation. While satellite engineers have significantly reduced the costs of these missions, developing and deploying satellites — particularly large constellations comprising thousands of units[1] — still requires considerable time and investment. Once in orbit, stakeholders depend on satellites for various tasks, such as providing internet service and collecting data for scientific research or defense purposes. Unfortunately, in-orbit events and internal malfunctions can compromise a satellite's functionality, leading to anomalous behaviors known as *faults*. These faults, often detectable in the satellite telemetry data, signify deviations from normal operations and can require advanced mitigation strategies.

Unfortunately, manually monitoring telemetry faults through ground operations is both time-consuming and labor-intensive for mission operators. It also relies heavily on consistent and reliable communication between the satellite and the ground station. Furthermore, communication latencies can result in delayed responses, which can be inadequate for addressing time-sensitive faults. These obstacles highlight the need for *onboard* and *autonomous* strategies to enhance satellite resilience against anomalous events, thereby increasing satellite reliability and the likelihood of mission success.

For this study, we focus on advancing two aspects of autonomous fault response: fault *detection* and fault *attribution*. Fault detection involves recognizing abnormal behavior, while fault attribution involves associating the anomaly to a specific event or cause. To this end, we develop a joint fault detection and attribution framework that employs ensemble representation learning techniques and decision trees applied to satellite telemetry data. Specifically, we utilize Autoencoders (AE), Kalman Filters (KF), Gaussian Mixture Models (GMM), Long Short Term Memory networks (LSTM), and PCMCI causal discovery to learn representations of satellite telemetry data. We then use the XGBoost decision tree classifier model to detect and attribute faults based on the learned representations. We develop these capabilities for integration with Auria Space's flight-certified Autonomous Planning System Framework (APS).

In addition to developing a machine learning framework for fault detection and attribution, we also create a novel telemetry simulation tool named SatFaultSim. This tool is designed for the training and validation of fault detection and attribution methods. It represents a significant improvement over previous efforts in autonomous fault diagnosis, which have often been constrained by a lack of historical faulting telemetry data and the absence of detailed ground truth labels. SatFaultSim addresses these challenges by providing a flexible simulated environment where diverse fault scenarios can be systematically introduced and evaluated.

We utilize our newly developed telemetry simulation tool, SatFaultSim, to generate telemetry data across 10 orbits, including normal operations and 11 randomly occurring faults. This simulated dataset serves as the foundation for training and validating our ensemble learning approach for fault detection and attribution. After conducting a thorough evaluation of our model's predictions, the results are highly encouraging: our approach achieves an overall combined fault detection and attribution accuracy of 99.89%, with a detection accuracy of 99.94% and an average fault attribution accuracy of 99.97%. Additionally, the false positive and negative rates for each fault type are exceptionally low, all below 0.013%.

These metrics demonstrate that our framework is not only proficient at identifying known faults but also consistently recognizes a diverse array of fault types. Furthermore, we showcase the functionality and potential of SatFaultSim for supporting ongoing research efforts by offering a controlled environment for algorithmic testing and validation. Lastly, our ensemble learning framework sets a promising foundation that can be further enhanced with self-supervised and unsupervised learning techniques to address rare or previously unseen fault types, broadening the applicability and robustness of our ap-

[†]`https://github.com/madelineloui/SOFAR`

proach for satellite operations.

Previous Work

We review several significant contributions to the field of autonomous fault resilience strategies, beginning with advancements in fault *detection*. The survey by Goldstein et al. provides a comprehensive overview of unsupervised statistical methods for multivariate fault detection, including clustering and outlier detection techniques.[2] Boumghar et al. explore the application of deep learning through autoencoder neural networks for identifying anomalous spacecraft behavior.[3] Hundman et al. apply semi-supervised learning techniques, utilizing recurrent neural networks (RNNs) coupled with non-parametric thresholding, to detect spacecraft anomalies.[4] Carlton investigates the efficacy of statistical change-point event detection methods for monitoring satellite telemetry streams.[5]

We also consider key developments in autonomous fault *diagnosis* and *attribution*. Tipaldi et al. review various fault detection and recovery strategies for space projects from a systems engineering perspective.[6] Ibrahim et al. integrate support vector machines (SVM) with fault tree analysis to detect satellite failures and attribute them to likely causes.[7] Tudoroiu et al. develop a fault diagnosis system for spacecraft reaction wheel errors using interacting multiple model (IMM) filters.[8] Li et al. employ dynamic neural networks for similar diagnostic purposes.[9] Suo et al. focus on satellite power system fault diagnosis, employing fuzzy Bayes risk-based feature selection and SVMs.[10] Xie et al. introduce a fault diagnosis framework for power systems that utilizes Bayesian learning to deduce causal relationships.[11] Lastly, Gizzi et al. present a novel approach to fault detection and attribution for both known and rare fault cases, leveraging a combination of Kalman Filters, Autoencoders, and causality networks.[12]

These works collectively speak to the diverse and innovative approaches being explored and applied within the realm of satellite fault management. They also highlight the transition towards more sophisticated, autonomous systems for ensuring greater reliability in space operations.

Contributions

While many existing fault diagnosis approaches offer significant insights, we observe that they often depend on a single learning model. In contrast, ensemble learning, which aggregates predictions from multiple models to enhance overall performance for a given task,[13] remains underutilized in this domain. Moreover, the acquisition of appropriate training and evaluation data presents substantial challenges: datasets are not always accessible, faults occur sparsely, and satellite mission data frequently lacks detailed ground truth labeling. The variability in data used for evaluating different approaches also complicates the comparison of algorithmic strategies.

To address these challenges, we provide the following contributions:

1. We develop an algorithmic approach for joint fault detection and attribution that leverages *ensemble learning* and *decision trees* to enhance the robustness and accuracy of fault diagnosis.
2. We develop a *statistical satellite telemetry simulation tool*, named SatFaultSim, that automatically generates normal and faulting telemetry data for training and evaluation purposes.
3. We demonstrate the effectiveness of our algorithmic framework using data generated by SatFaultSim and include a comprehensive evaluation of the model performance.

APPROACH

In this section, we provide a brief overview of our algorithmic approach for autonomous fault diagnosis. Figure 1 illustrates the inference pipeline of our fault detection and attribution module. Figure 2 depicts the module, highlighted by the yellow block, as part of a broader satellite fault resilience system. Data sources feed telemetry samples into the system for learning and inference, while the onboard response system and ground mission operators interface with autonomous fault detection and attribution predictions for enhanced resilience.

The module performs joint fault detection and attribution by leveraging an ensemble of five representation learning algorithms and a decision tree classifier. The data representation ensemble consists of the following: an Autoencoder[14] (AE), Kalman Filter[15] (KF), Gaussian Mixture Model[16] (GMM), Long Short Term Memory network[17] (LSTM), and the PCMCI causal discovery algorithm[18] (PCMCI), each capturing different latent aspects of the telemetry data. After concatenating the latent representations, we apply Extreme Gradient Boosting[19] (XGBoost), a form of Gradient Boosted Decision Trees, as a supervised classification approach for both detecting and attributing fault causes using the learned

representations of the data. We expand on each model in the following subsections.

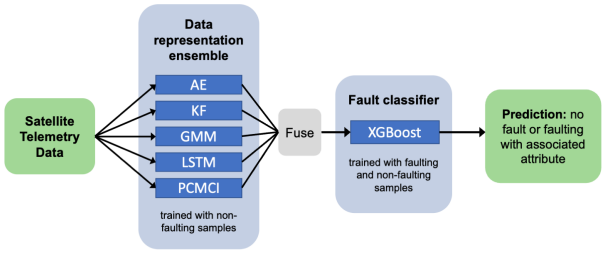

Figure 1: Fault Detection and Attribution Module

Autoencoder

Autoencoder neural networks[14] (AE) learn lower dimensional representations of data by utilizing unsupervised data reconstruction. An AE consists of two main components: an encoder and a decoder. First, the encoder f compresses the input data into a lower dimensional latent space. Then, the decoder g projects the latent representation back into the original input data space. The network learns optimal parameters for the encoder and decoder by performing data reconstruction, where the objective is to minimize the *reconstruction error*. This error compares the original input, x, with the the decoder's output, $\hat{x} = g(f(x))$ aiming to make the reconstructed output as close as possible to the initial data. While different loss functions can be used, mean squared error (MSE) is a popular method for quantifying the reconstruction error for AE neural networks:

$$L(x, \hat{x}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{1}$$

Several works employ the AE for fault-related applications: for example, An et al. utilize the reconstruction error of a Variational Autoencoder (VAE) to detect intrusions in communication networks[20] and Fang et al. use Denoising Autoencoders (DAE) to detect faults in satellite electrical power system streams.[21] Gizzi et al. similarly leverage the reconstruction loss of an AE to detect satellite faults in telemetry data.

We leverage a neural-network based AE to detect anomalous data by first training the model with sliding windows of nonfaulting telemetry data. During inference, we compute the reconstruction error for each telemetry signal at each time step. Higher reconstruction errors can indicate the presence of anomalies.

Kalman Filter

The Kalman filter[15] (KF) estimates states of a linear dynamic system with Gaussian noise. The filtering process consists of two main steps: prediction and update.

In the prediction step, the KF predicts the next system state and error covariance based on the system's previous states and the known behavior of the system:

$$\hat{x}_{n,n-1} = A\hat{x}_{n-1,n-1} + Bu_n \tag{2}$$

$$\Sigma_{n,n-1} = A\Sigma_{n-1,n-1}A^T + Q \tag{3}$$

where in Eq. 2, $\hat{x}_{n,n-1}$ is the predicted state estimate, A is the state transition matrix, B is the control-input model, and u_n is the control vector. In Eq. 3, $\Sigma_{n,n-1}$ is the predicted covariance estimate and Q is the process noise covariance.

In the update step, the filter updates its prediction based on the new measurement:

$$K_n = \Sigma_{n,n-1}H^T(H\Sigma_{n,n-1}H^T + R)^{-1} \tag{4}$$

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_n(z_n - H\hat{x}_{n,n-1}) \tag{5}$$

$$\Sigma_{n,n} = (I - K_nH)\Sigma_{n|n-1} \tag{6}$$

where in Eq. 4, K_n is the Kalman Gain, H is the observation model, and R is the measurement noise covariance. In Eq. 5, $\hat{x}_{n,n}$ is the updated state estimate and z_n is the actual measurement. In Eq. 6, $\Sigma_{n,n}$ is the updated covariance estimate and I is the identity matrix.

The residual error in a Kalman Filter is the difference between the actual and predicted measurement at each time step. It indicates how well the filter's predictions match the real observations:

$$y_n = z_n - H_n\hat{x}_{n,n-1} \tag{7}$$

The residual error can capture potential anomalies in the data while taking into account Gaussian noise. Several approaches utilize the Kalman filter for fault diagnosis, such as implementing a multiple model based approach to diagnose power system faults in NASA missions,[22] using multiple interacting Kalman filters to monitor faults,[8] and employing the z-score of the residual error to determine the significance of potential fault attributes.[12]

In our prototype, we learn the control-input model, B, by fitting a linear system for each signal over time using an auto-regressive model. During inference, we extract the residual error, y_n, of the model after each update, which measures the distance between observed and expected values. The

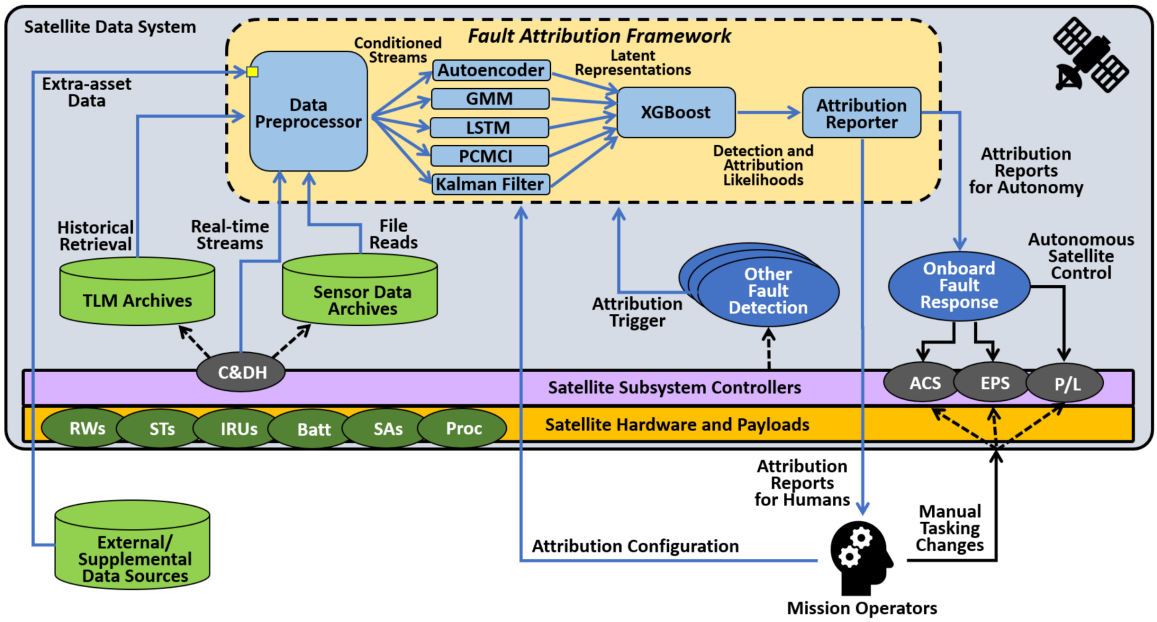

Figure 2: Fault Diagnosis and Response System

error measurement can indicate potential anomalies in the satellite system.

Gaussian Mixture Model

The Gaussian Mixture Model[16] (GMM) is a probabilistic method used to represent data as a mixture of Gaussian distributions. Each Gaussian component has a corresponding mean, covariance and weight, which indicates the amount it contributes to the overall model. The probability density of a GMM is given by

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{8}$$

where K is the number of components, and π_k and $N(x|\mu_k, \Sigma_k)$ are the weight and the Gaussian distribution of the k-th component, respectively.

The GMM fits the parameters (means, covariances, and mixture weights) to the observed data using the Expectation-Maximization (EM) algorithm. In the expectation, or E-step, it calculates the probability that each data point belongs to each Gaussian component given current parameter estimates:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{9}$$

where $\gamma(z_{nk})$ is the probability that observation x_n belongs to component k, also known as the "responsibility" of the component for that data point.

In the maximization, or M-step, it updates the weights, means, and covariances of each component by maximizing the likelihood of the data given the responsibilities calculated in the E-step:

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \gamma(z_{nk}) \tag{10}$$

$$\mu_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{11}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_n)(x_n - \mu_n)^T}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{12}$$

where N is the total number of data points. The parameters can be randomly initialized at the start. The process iteratively performs the EM steps until convergence.

GMMs can effectively identify anomalies in data, such as detecting potential safety issues on commercial airlines by assessing flight parameters like airspeed, altitude, pitch, and roll.[23] To apply the GMM to satellite telemetry faults, we first fit the GMM to non-faulting data. We can then utilize the model to assess deviations in new data observations by evaluating their likelihood under the learned model, as defined in Eq. 8. Observations that yield low likelihoods, indicating deviations from the expected model, may signal anomalies.

Long Short Term Memory

The Long Short Term Memory network[17] (LSTM) is a type of recurrent neural network (RNN) that captures long-range dependencies in sequential data using gating mechanisms. The network architecture consists of many LSTM "cells", including input gates, forget gates, and output gates for processing and retaining relevant data. The forget gate discards irrelevant information from the cell:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{13}$$

where W_f is the weight matrix and b_f is the bias of the forget gate, h_{t-1} is the output of the previous LSTM unit, and x_t is the input at the current time step.

The input gate updates the cell state based on new information. It consists of a sigmoid gate, which decides which values to update, and a tanh activation layer, which creates a vector of new candidate values to be added to the state:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{14}$$

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{15}$$

where i_t is the output of the input gate and $\tilde{C}_t$ is the vector of new candidate values. Then, the cell state C_T is updated by forgetting irrelevant components of the previous states via the forget gate and adding new candidate values scaled by the input gate:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{16}$$

The output gate determines the next hidden state h_t by filtering the cell state with the output gate:

$$\sigma_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{17}$$

$$h_t = o_t * tanh(C_t) \tag{18}$$

Various studies have utilized LSTM networks for fault detection and similar applications. For instance, Malhotra et al. design a stacked LSTM network to detect anomalies in time series data, such as for space shuttle and multi-sensor engines.[24] Another work leverages a convolutional LSTM for anomaly detection in videos.[25]

In our approach, we first train the LSTM with non-faulting data to forecast sequential telemetry signals. The network optimizes parameters by minimizing the loss function, which compares the predictions with observed values using a measure such as mean squared error (MSE), similar to Eq. 1. During inference, we feed sequential telemetry signals through the LSTM network and extract the mean squared error (MSE) loss at each time step, which can indicate the presence of a fault.

PCMCI Causal Discovery

PCMCI is a causal network discovery algorithm developed by Runge et al. for time series data.[18] It estimates causal parents using a two-step process beginning with the PC (Peter-Clark) algorithm for initial parent selection and the momentary conditional independence (MCI) algorithm for further pruning. The first step, the PC algorithm, is a Markov discovery algorithm using conditional independence tests. It selects an initial parent set, $\hat{P}(X_t^j)$, for each variable at time step t, $X_t^j \in \{X_t^1, X_t^2, \ldots, X_t^N\}$, by testing the conditional independence of each possible parent. The MCI condition further tests the selected parents:

$$MCI: X_{t-\tau}^i \not\perp\!\!\!\perp X_t^j | \hat{P}(X_t^j) \backslash \{X_{t-\tau}^i\}, \hat{P}(X_{t-\tau}^i) \tag{19}$$

by conditioning on both the parents and time shifted parents.

Runge et al. successfully demonstrate that PCMCI can reconstruct air flow mechanisms in the tropical Pacific using causal inferences.[18] Similarly, Gizzi et al. employ PCMCI to identify possible causal relationships within spacecraft systems, enabling direct prediction of which satellite system may contribute to a fault.[12] This usage of PCMCI aligns closely with our work, highlighting the utility of PCMCI in complex system diagnosis. In our approach, however, we primarily employ PCMCI as a tool for causal representation learning rather than for direct attribution. We focus on evaluating the strongest causal parents of each signal at each time step, capturing dependencies across the flight system that are crucial for real time fault diagnosis.

Decision Tree Classifier

XGBoost,[19] or Extreme Gradient Boosting, is a form of Gradient Boosted Decision Trees (GBDT), which learns an optimal decision tree by adding or pruning hierarchical trees to best fit the data. Boosting refers to a meta-learning strategy of iteratively training weak learners to improve performance on instances with high error rates.[26] GBDTs are trained sequentially, where each subsequent tree attempts to correct the errors made by the previous trees. It performs the following prediction update at each iteration t:

$$F_t(x_i) = F_{t-1}(x_i) + \gamma \cdot h_t(x_i) \tag{20}$$

where $F(x_i)$ is the prediction, γ is the learning rate, and $h_t(x_i)$ is the prediction of the weak learner h_t. Gradient boosting finds the optimal weak learner h_t at each iteration that minimizes the loss function $\mathcal{L}(y_i, F(x_i))$:

$$h_t(x_i) = -\frac{\delta \mathcal{L}(y_i, F(x_i))}{\delta F(x_i)} \tag{21}$$

GBDTs can flexibly learn on datasets with continuous or discrete variables, different scales and units, and can function with missing data. The robustness and flexibility of XGBoost enables its use with any number of data streams from different sources, accommodating almost any data type. Another significant benefit of decision trees is their interpretable nature. Given that fault detection algorithms have traditionally relied on threshold and rules-based systems, XGBoost is particularly well-suited for these tasks by automatically and efficiently learning an optimal set of interpretable rules.

Ikram et al. apply a similar strategy, integrating outputs from several deep learning models and using XGBoost to identify cybersecurity intrusions.[27] Another study combines k-means clustering with decision trees for detecting communication anomalies, employing XGBoost to generate interpretable classification rules.[28]

In our approach, we concatenate the features learned from the five representation learners, then train XGBoost to classify the aggregated features at each time step as either non-faulting or one of 11 fault types. We will elaborate on the individual fault cases in the following section.

FAULT SIMULATION TOOL

To address the shortage and sparsity of relevant, complete, and labeled faulting datasets for satellite telemetry faults, we developed our own faulting telemetry simulation tool, called *SatFaultSim*, to facilitate the algorithmic evaluation of our approach. The tool eliminates the need to compile disparate data and hand label telemetry samples and also provides a controlled environment for comparative studies.

By leveraging the expert domain knowledge of satellite engineers and analysing telemetry data from real satellite missions operated by the Air Force Research Laboratory (AFRL) and the MIT Space Telecommunications, Astronomy and Radiation (STAR) Laboratory, we design the tool to mimic various satellite subsystems and common fault types. SatFaultSim currently supports 14 different telemetry variables, including heater voltages and current draws; battery voltages, currents and temperature readings; solar panel temperature measurements; radio communication current draws during pre-determined downlink opportunities, and additional voltage, current and temperature readings for various components throughout the onboard computer system. The tool simulates telemetry samples in 90-minute orbits, including eclipse periods, at adjustable sampling rates. It can generate both non-faulting data and 11 different fault scenarios that fall under 4 broader categories: ionizing radiation faults (IRF), system resets (SR), thermal faults (K), and current faults (I), recorded in Table 1.

SatFaultSim can be easily modified and extended. User input through configuration files enables adjustment of statistical values of both faulting and non-faulting data, along with the frequency and type of faults, the length of the simulated data, and the sampling rate. This flexibility ensures that the simulation can accommodate for variances in different satellite systems. The tool also includes randomness in the data generation process to imitate natural variations and prevent over-fitting during model training. While the simulations do not capture every possible fault, the tool can be extended to accommodate additional fault types without interfering with the existing functionality.

EXPERIMENTS

We train and evaluate our ensemble representation learner and XGBoost classifier approach using 10 orbits of data simulated by SatFaultSim. We first train the ensemble representation learning algorithms using simulated non-faulting telemetry data. Then, we feed a set of simulated data containing *both* faulting and non-faulting samples through the trained ensemble algorithms to create a training set for the decision tree classifer. The data input to the representation learners is a sliding window of 20 telemetry samples. The output consists of 92 total features, in which features 0-14 contain the AE reconstruction error values for each signal, feature 15 is the GMM likelihood score, features 16-30 are the KF residual error values for each signal, feature 31 is the LSTM loss, features 32-76 encode the 3 strongest causal parents of each signal from the PCMCI algorithm, and features 77-91 are the raw signal values at the current time step. We split the aggregated features into training and test sets, shown in Table 2, and fit the XGBoost model to the training set using a tree depth of 6 for 100 iterations with early stopping.

Table 1: Simulated Fault Cases

Ionizing Radiation Fault	IRF_Z	Radiation results in zeros for some or all variables
	IRF_S	Radiation causes stale data (stuck until reset or power change)
	IRF_R	Radiation causes garbage values (bit flips)
System Reset	SR_C	System reset causes jump in data (all variables, including onboard clock)
	SR_N	System reset causes jump in data (all variables except clock, since clock is a ground clock)
Thermal Fault	K_L	Sensor loss causes NAN values (-999)
	K_F	Sensor failure causes abnormally high/low values, garbage thermal values, or stale values
	K_C	Nearby component failure causes abnormal low/decreasing thermal values or abnormal high/increasing thermal values
	K_H	High external temperatures over sensor maximum reading
Current Fault	I_O	Mechanical failure causes high current values
	I_U	Mechanical failure causes low current values

Table 2: Distribution of Fault Cases in Training and Test Sets

Fault Case	Train	Test
No fault	71653	18019
IRF_Z	2292	561
IRF_S	7213	1706
IRF_R	2197	563
SR_C	40	13
SR_N	45	7
K_L	2458	602
K_F	3509	861
K_C	5173	1280
K_H	3704	937
I_O	1894	462
I_U	4943	1265

Evaluation Metrics

To quantify the performance of our approach, we evaluate the detection and attribution accuracy, true positive rate (TPR) and false positive rate (FPR) for each individual faulting case and for the combined system. The accuracy compares the number of correct predictions to the total number of predictions:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{22}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. The TPR compares the number of correct detections to the total number of positive samples with the following ratio:

$$TPR = \frac{TP}{TP + FN} \tag{23}$$

while the FPR compares the number of correct rejections to the total number of negative samples with the following ratio:

$$FPR = \frac{FP}{TN + FP} \tag{24}$$

We also perform two computational performance benchmark tests using a desktop computer and a Raspberry Pi as a preliminary assessment of the viability of onboard implementation. The desktop computer has an Intel i7 2.60 GHz CPU with 12 threads and 32 GB RAM. The Raspberry Pi has a Cortex-A72 1.8 GHz processor with 4 threads with 1.8 GB

RAM. We measure the memory and compute time for training and inference of the proposed algorithmic approach on these platforms.

Table 3: Overall Performance

Metric	Accuracy (%)
Combined fault detection and attribution accuracy (all cases)	99.89
Combined fault detection and attribution accuracy (only faulting cases)	99.79
Detection accuracy	99.94
Detection True Positive Rate (TPR)	99.88
Detection False Positive Rate (FPR)	0.039
Average attribution accuracy	99.97
Average attribution TPR	99.85
Average attribution FPR	0.023

RESULTS

We record evaluation metrics for both detection and attribution tasks. Table 3 reports the overall performance metrics. Combined fault detection and attribution accuracy measures the number of correct fault detections and attributions to one of 11 fault cases. We evaluate this for the entire dataset (both non-fault and faulting cases) and achieve an accuracy of 99.89%. To ensure that the high accuracy is not dominated by the non-faulting cases, which make up the majority of the dataset, we also measure the combined fault detection and attribution accuracy of only faulting cases, which is 99.79%. This gives us confidence that the algorithm performs accurately for both faulting and non-faulting cases. Next, we evaluate overall detection performance, which indicates the algorithm's ability to distinguish the presence and absence of a fault (as opposed to attribution, which involves association of a fault to a specific event). The detection accuracy is 99.94%, the TPR is 99.88%, and the FPR is 0.039%. The high TPR indicates that there is a high frequency of true positives in proportion to the total number of positive samples. The low FPR indicates that there is a low frequency of false negatives in proportion to true negatives. We similarly evaluate these metrics for fault attribution, which measures the algorithm's ability to associate faulting data to the correct cause. The attribution accuracy is 99.97%, the TPR is 99.85% and the FPR is 0.023%. Overall, the high accuracies, high true positive rates, and low false positive rates all indicate that the prototype is exceptionally competent for fault detection and attribution of known faults within the simulated environment.

We further investigate per-fault performance metrics to evaluate the consistency of the algorithmic approach across different fault types, shown in Table 4. For each of the 11 fault cases, we evaluate the attribution accuracy, which measures the algorithm's ability to correctly attribute the particular fault given fault detection. We also evaluate the combined detection and attribution accuracy, which measures the algorithm's ability to both detect and attribute the fault. We also report the attribution TPR and FPR to gain insight on the rate of true and false negatives across fault types. We illustrate the error rates for each fault case in Figure 3, which conveys low false positive and negative rates throughout, with a maximum error rate of only 0.013%. Overall, the high attribution accuracy, high combined detection and attribution accuracy, high attribution TPR, and low attribution FPR values across all fault cases indicate that the prototype is able to successfully identify diverse fault types.

Table 4: Per-Fault Performance

Fault Type	Attribution Accuracy	Detection and Attribution Accuracy	Attribution FPR
IRF_Z	99.98%	99.73%	0.026%
IRF_S	99.98%	99.88%	0.031%
IRF_R	99.90%	99.11%	0.013%
SR_C	100.0%	100.0%	0.000%
SR_N	100.0%	100.0%	0.000%
K_L	99.98%	99.83%	0.026%
K_F	99.96%	99.77%	0.000%
K_C	99.93%	99.65%	0.072%
K_H	99.96%	99.84%	0.041%
I_O	100.0%	100.0%	0.000%
I_U	99.96%	99.84%	0.043%

Feature Importance

XGBoost enables examination of the learned optimal tree hierarchy, which can provide interpretability and insight into the model behavior. Specifically, we can calculate the F-score, which indicates the number of times each feature was split on within the decision tree. Figure 4 displays the importance of different input features based on their F-score value. Feature numbers can be referenced from the Experiments section. Based on this evaluation, the most

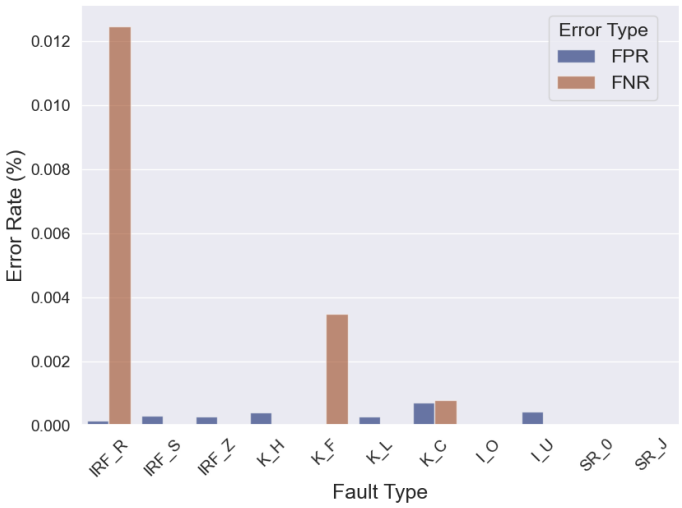

Figure 3: False Positive and Negative Rates Across Fault Types

important features from our experiment include the clock's KF residual error and the solar panel temperature's AE reconstruction error.

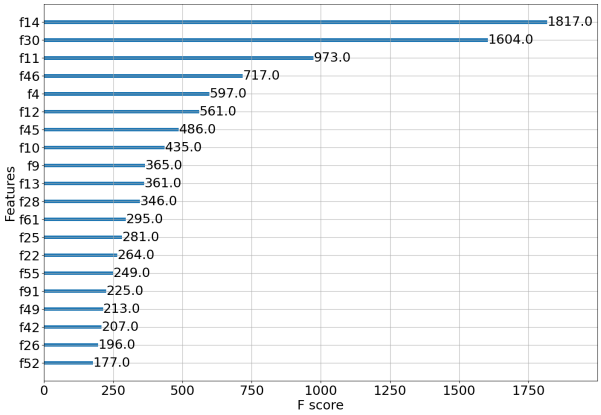

Figure 4: Ranked XGBoost Feature Importance

Computational Load Performance

We report training and evaluation times on both desktop and flight-like hardware to characterize computational performance, recorded in Table 5. We trained the fault detection and attribution model, then ran the evaluation using 115,000 time steps with 14 telemetry variables, consisting of 1.7 million total data points. We highlight the low process memory usage (under 100 Mb) and rapid evaluation time (0.9s for the desktop and 3.0s for the Raspberry Pi to evaluate over 100,000 samples) amounting to less than 0.0001 seconds per sample, which is more than sufficient for real-time use on most onboard computers.

Discussion

Our evaluation conveys that our prototype operates with high accuracy and precision using the simulated telemetry data. The XGBoost model leverages the data representations from the ensemble methods to efficiently learn patterns in faulting and non-faulting data. Low error rates across 11 different fault cases confirm that the system can accurately identify a diverse array of fault scenarios. While the simulated faults may not encompass every potential error that could occur during flight, a custom fault resilience system that leverages this framework can provide in-orbit resilience using a priori knowledge of the system to anticipate and diagnose common faults. Additionally, hardware tests reveal a minimal computational load, revealing its potential viability for integration with onboard flight systems.

FUTURE WORK

There are many opportunities to build upon the existing algorithmic framework and simulation tool, such as the following possible directions:

1. *Roles of different representation learners:* An in-depth study on the role and importance of each representation learner for the classification step may enable simplification of the framework. This can minimize the extraction of unimportant features and thereby save computational resources.

2. *Enhancing robustness, adaptability, and scalability:* While our methodology yields highly promising results in diagnosing known faults

Table 5: Platform Specifications and Performance

Platform	RAM Size	Memory Usage	CPU	Threads	Train Time	Eval. Time
Desktop	32 GB	<100 Mb	Intel i7 2.60GHz	12	17m 40s	0.9s
Raspberry Pi	1.8 GB	<100 Mb	Cortex-A72 1.8GHz	4	67m 51s	3.0s

within a simulated environment, it predominantly relies on supervised techniques and may struggle with previously unseen or rare real-world fault cases. To address this limitation, it is essential to integrate semi-supervised and un-supervised techniques, such as active learning and non-parametric methods, into the framework. Additionally, exploring the scalability of our method to accommodate higher-dimensional telemetry data streams is crucial for ensuring its applicability to more complex systems.

3. *Onboard execution for satellite missions:* We would like to integrate our framework into a satellite flight system for real-time fault resilience. This would involve implementing the algorithms on an onboard computer architecture and evaluating its performance with real satellite telemetry signals.

4. *Integration into a broader satellite decision-making system:* Our approach is modular and can be used within a larger planning and decision-making system, as shown in Figure 2. There is growing interest in how autonomous onboard decision-making systems can leverage fault predictions to react and resolve possible errors. A promising direction for further development includes creating optimization algorithms for planning and scheduling recovery actions when a fault is detected.

5. *Enhance SatFaultSim capabilities:* Future work will focus on diversifying the supported fault types using additional datasets from satellite missions. This involves not only modeling common faults, but also rare and complex anomalous behaviors. This approach will enhance the system's robustness and ability to handle a broader spectrum of potential issues.

Our research demonstrates the effectiveness of ensemble learning and decision trees for detecting and attributing known fault types. Additionally, our telemetry simulation tool has proven highly valuable for training and evaluating the algorithmic framework. The numerous future directions highlight the potential of both the simulation tool and initial system prototype to drive future research for improving onboard autonomous fault resilience capabilities.

Acknowledgments

This work was funded by Air Force STTR Grant FA9453-22-CA-124. We would like to thank Michelle Simon for her research support as our AFRL technical monitor.

References

[1] Giacomo Curzi, Dario Modenini, and Paolo Tortora. Large constellations of small satellites: A survey of near future challenges and missions. *Aerospace*, 7(9):133, 2020.

[2] Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[3] Redouane Boumghar, Adithya Venkataswaran, Hugh Brown, and Xabier Crespo. Behaviour-based anomaly detection in spacecraft using deep learning. 2021.

[4] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395, 2018.

[5] Ashley Kelly Carlton. *Fault detection algorithms for spacecraft monitoring and environmental sensing*. PhD thesis, Massachusetts Institute of Technology, 2016.

[6] Massimo Tipaldi and Bernhard Bruenjes. Survey on fault detection, isolation, and recovery strategies in the space domain. *Journal of*

Aerospace Information Systems, 12(2):235–256, 2015.

[7] Sara K Ibrahim, Ayman Ahmed, M Amal Eldin Zeidan, and Ibrahim E Ziedan. Machine learning techniques for satellite fault diagnosis. *Ain Shams Engineering Journal*, 11(1):45–56, 2020.

[8] N Tudoroiu and K Khorasani. Satellite fault diagnosis using a bank of interacting kalman filters. *IEEE Transactions on Aerospace and Electronic Systems*, 43(4):1334–1350, 2007.

[9] ZQ Li, Liying Ma, and Khashayar Khorasani. A dynamic neural network-based reaction wheel fault diagnosis for satellites. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3714–3721. IEEE, 2006.

[10] Mingliang Suo, Baolong Zhu, Ruoming An, Huimin Sun, Shengzhong Xu, and Zhenhua Yu. Data-driven fault diagnosis of satellite power system using fuzzy bayes risk and svm. *Aerospace Science and Technology*, 84:1092–1105, 2019.

[11] Siyun Xie, Xiafu Peng, Xunyu Zhong, and Chengrui Liu. Fault diagnosis of the satellite power system based on the bayesian network. In *2013 8th International Conference on Computer Science & Education*, pages 1004–1008. IEEE, 2013.

[12] Evana Gizzi, Hayley Owens, Nicholas Pellegrino, Christopher Trombley, James Marshall, and Jivko Sinapov. Autonomous system-level fault diagnosis in satellites using housekeeping telemetry. *Small Satellite Conference*, 2022.

[13] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14:241–258, 2020.

[14] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374, 2023.

[15] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[16] Douglas Reynolds. *Gaussian Mixture Models*, pages 659–663. Springer US, Boston, MA, 2009.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11):eaau4996, 2019.

[19] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[20] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special lecture on IE*, 2(1):1–18, 2015.

[21] Hongzheng Fang, Hui Shi, Yunfan Dong, Huanzhen Fan, and Shuai Ren. Spacecraft power system fault diagnosis based on dnn. pages 1–5, 2017.

[22] Marc A Carbone and Kenneth A Loparo. Fault detection and diagnosis in spacecraft electrical power systems. *Journal of Aerospace Information Systems*, 20(6):308–318, 2023.

[23] Lishuai Li, R John Hansman, Rafael Palacios, and Roy Welsch. Anomaly detection via a gaussian mixture model for flight operation and safety monitoring. *Transportation Research Part C: Emerging Technologies*, 64:45–57, 2016.

[24] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *Esann*, volume 2015, page 89, 2015.

[25] Weixin Luo, Wen Liu, and Shenghua Gao. Remembering history with convolutional lstm for anomaly detection. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 439–444, 2017.

[26] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[27] Sumaiya Thaseen Ikram, Aswani Kumar Cherukuri, Babu Poorva, Pamidi Sai Ushasree, Yishuo Zhang, Xiao Liu, and Gang Li. Anomaly detection using xgboost ensemble of deep neural network models. *Cybernetics and information technologies*, 21(3):175–188, 2021.

[28] João Henriques, Filipe Caldeira, Tiago Cruz, and Paulo Simões. Combining k-means and xgboost models for anomaly detection using log datasets. *Electronics*, 9(7):1164, 2020.