Supervised and Unsupervised Methods for Detecting Anomalies in an Autonomous Long-Range Lunar Rover

Sofie Claridge[1], Jack Patterson[1], Zaki Hasnain[1], Ashish Goel[1], Katharine Patterson[1], Nicolas F. Rouquette[1], Caleb Wagner[1], Tristan D. Hasseler[1], and Michel D. Ingham[1],

[1] *Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109*
michel.d.ingham@jpl.nasa.gov

ABSTRACT

Autonomous space systems must navigate complex environments and accomplish concurrent tasks without continuous input and supervision from human operators. Interactions between subsystems and with the environment may lead to unintended behavior, resulting in downtime, delays, goal degradation, loss of functionality, or even catastrophic failures. The Endurance mission concept requires a lunar rover to traverse thousands of kilometers over a long period of time (multiple years) across the South Pole-Aitken Basin on the far side of the Moon. Onboard system health management is therefore required to identify anomalies and faults that pose a risk to mission objectives. To this end, we propose training both unsupervised and supervised machine learning models to detect and isolate anomalies onboard a rover over the course of a mission, supported by a pipeline for anomaly data generation that enables training and evaluation. We generate synthetic anomaly signatures using a low-fidelity mission simulator that outputs labeled datasets to enable supervised learning. We present results from a field experiment in which we deploy this supervised model as a ROS network node in a rover-ground network. In parallel, we train unsupervised models to detect anomalies in the mobility system by training on experimental field data and present results that verify the ability to detect anomalies observed by field operators as well as anomalies that were not detected by the operators. Our work demonstrates how machine learning models can detect anomalies onboard by leveraging multiple data sources, including pre-launch test data and operational data from earlier phases of the mission, and provides a pathway for improving anomaly detection as a rover's mission progresses.

1. INTRODUCTION

1.1. Motivation

Autonomous operation is critical for enabling future lunar rover missions to explore larger surface areas over longer periods of time (Baker et al., 2024; Hartman, 2021). In addition to the challenges imposed by navigating uncharted and potentially hazardous environments, the latency in telecommunication with ground operators further stresses the need for reliable autonomous operation of spacecraft. To achieve such automation, careful orchestration between onboard system-level autonomy (e.g., onboard planning, scheduling, and execution (Ingham et al., 2024)) and all the various rover subsystems and components is required. System health management (SHM) is the onboard autonomous capability that addresses these requirements, maintaining safety of critical systems, and enabling the mission to continue operating, even in the presence of anomalies, between ground telecommunication windows.

1.2. Anomaly types

The types of anomalies a long-range lunar rover mission may encounter could originate from erroneous ground uplinks, model errors, system-level reasoning, a fault or degradation in a subsystem or component, unexpected interactions between subsystems, or from the lunar environment. This results in a wide range of potential complex rover anomalies. Local rule-based and statistical thresholding methods offer interpretability and low computational cost, however they may struggle in capturing the nonlinear dependencies and high dimensional trends that nuanced anomaly detection demands.

1.3. SHM functions

A complete SHM system consists of five general functions: 1) detection of an ongoing or concluded anomaly, 2) isolation of an anomaly, by determining its origin within the system, 3) diagnosis of an anomaly, by identifying its cause (e.g., the particular component fault mode or environmental effect that

triggered the anomaly), 4) prognosis by describing the future progression of an ongoing anomaly or predicting the probability of a future anomaly, and 5) response to current or anticipated anomalous conditions by identifying maintenance, mitigation, or recovery actions. In this paper, we focus on the first two of these five functions.

1.4. SHM metrics

A key metric for the success of SHM is the early detection of anomalies so that response control functions are allowed time to reason. However, disparate subsystem-based fault detectors may fail to pick up complex faults and may only trigger when component-specific operation limits are violated. Time series machine learning models designed for anomaly detection may be able to detect cross-subsystem anomalies and flag them early enough for onboard system autonomy to respond before the rover sustains damage or the scheduled tasks are significantly disrupted or delayed. Further, an appropriate balance between anomaly detection precision and recall must be tailored to each component or subsystem based on risk tolerance and mission concept of operations (ConOps). Finally, onboard SHM tools must operate within computational constraints and must execute estimation at a cadence that is compatible with the rate of operation.

1.5. Learning based approach to SHM

Traditional rule-based SHM methods may lack the complexity required to understand system-level behavior which is based on the mapping between goal intention, commanded control arguments, and subsystem and component response. In addition, the difficulty of collating multiple rule-based monitors to efficiently inform system-level autonomy scales unfavorably with the number of monitored components. Beyond rule-based approaches, SHM has also leveraged model-based reasoning techniques grounded in physics or behavioral models. These methods can provide strong interpretability and diagnostic capability by explicitly capturing system dynamics, but may be difficult to develop and maintain for complex, high-dimensional systems with many interacting components. Learning-based SHM tools offer an alternative approach, but also introduce their own challenges. Particularly, sufficiently large datasets containing signatures of anomalies and nominal behavior are required to properly train anomaly detection models. Supervised methods require data representative of anomalous behavior, or accurate models thereof, which may be challenging to obtain. On the other hand, unsupervised methods can be trained solely on nominal data and flag any departure from the distribution of nominal behavior as anomalous.

Due to the unique requirements of space missions, and indeed the proposed Endurance rover mission, only simulations and ground-based testing can provide trainable data prelaunch. This limits the application of learning techniques to only those subsystems or components for which sufficient data is available prior to launch. However, the unique opportunity that learning-based methods provide to long-duration missions is the ability to reduce risk as the mission progresses. Once a rover begins operation, nominal telemetry can be downlinked and used to retrain unsupervised models, thereby refining the model's understanding of nominal behavior. Similarly, telemetry from anomalies encountered during operation can be used to retrain supervised learning models. Although both these learning approaches are complementary and should be deployed in parallel, unsupervised models should be of high priority to all long-duration missions since a large fraction of mission telemetry is nominal, and nominal distributions may evolve as a rover enters new environments. Indeed, traditional rule-based methods can also be tuned once a spacecraft begins operating, however, learning-based SHM approaches can adapt simultaneously across components and subsystems via model retraining. System architects should compare the difficulty of retraining machine learning SHM models to the effort required to manually tune the multiple rule-based monitors. In addition, learning-based methods have the ability to prognosticate future states based on recent telemetry.

In multivariate time series data, where each telemetry channel may reflect a different subsystem or physical process, explainability methods can be used to identify the variables that contribute most to a detection. Some anomalies are localized to a subset of sensors, while others emerge from subtle correlations across the feature space. Therefore, knowing which features a SHM model depends on to make a detection can clarify both the nature and origin of the fault. For neural networks, attention weights, attribution maps, or saliency-based methods can provide this needed information. However, the extent to which such interpretability is possible depends heavily on the choice of model architecture and is a key consideration in the design of an anomaly detection system.

1.6. Outline

Here, we focus on anomalies related to the rover mobility subsystem for the unsupervised model, and power and thermal subsystem anomalies for the supervised learning model. Our modeling focuses on the first two SHM functions, detection and isolation, using supervised and unsupervised machine learning models and associated explainability mechanisms. We describe how field test data is used to train an unsupervised anomaly detector, similar to what can be achieved in a real mission using data from early stages of operation. We then describe the process for using a physics-based mission simulator as a synthetic data generator to train the supervised learning model. Finally, we describe how the models can be deployed onboard in a robot operating system (ROS) architecture (Macenski et al., 2022).

2. METHODS

2.1. Unsupervised Anomaly Detection

2.1.1. Field Data and Preprocessing

ERNEST (Exploration Rover for Navigating Extreme and Sloped Terrains) is a testbed rover developed to facilitate the testing and validation of new autonomous mobility and navigation software for the proposed Endurance mission (Baker et al., 2025). ERNEST uses rosbag recording to store time-stamped ROS message streams from the rover's telemetry. Telemetry is organized into ROS topics corresponding to different rover components and subsystems (e.g. odometry and hardware), each representing a stream of messages generated by a specific sensor, estimator, or onboard process.

As part of testing, ERNEST was run for 30 hours in the desert at El Centro, California. The rover navigated the terrain in an autonomous mode with some manual interventions. Field notes from operators accompany the collected ERNEST ROS topics. These field notes logged successful runs, battery swaps and unexpected mobility events. Due to the nature of manual annotation, the timing of events in the field notes does not precisely match the onset of behaviors in the rosbags. Although not perfectly complete, these field notes provide a useful basis for understanding key events experienced by the rover over the duration of the field test.

To demonstrate the initial feasibility of using unsupervised machine learning to detect anomalous behavior in ERNEST, the analysis focuses on mobility-related anomalies. During the field campaign, multiple anomalous mobility events were recorded, including "rover tipped", "front wheel stuck in soft sand", and "stuck in rocks". A total of 18 mobility-related field note entries are considered anomalous and used to evaluate the unsupervised model. ROS topics are selected according to their relevance to rover mobility. The selected hardware topics included actuator states, joint states, rocker encoding and temperature data. Two odometry-related topics are also selected to give information about the rover's orientation and position. In total, these topics provide 332 data channels. These 332 channels form the input feature space for the anomaly detection model and are resampled to a rate of 1 Hz to ensure consistent temporal alignment.

The resulting unsupervised workflow consists of telemetry selection and alignment, preprocessing into windowed multivariate sequences, and reconstruction-based anomaly scoring using a transformer autoencoder. This telemetry preprocessing infrastructure is shared with the supervised anomaly detection pipeline described in Section 2.2.

2.1.2. Transformer Autoencoder Model

Autoencoders have proven effective for anomaly detection in multivariate time-series data (Lakhmiri et al., 2022; Sabzehi & Rollins, 2024). A basic autoencoder consists of an encoder network that compresses the input features into a latent space and a decoder network that reconstructs the original signal. When trained only on nominal data, the model learns to reliably produce nominal feature distributions. When an anomalous sequence is then fed through the network, the model struggles to reconstruct the sequence in the final decoder stage. This results in a high reconstruction loss, which is used to flag the data as anomalous.

Standard autoencoder architectures are well suited to static data but can fail to capture temporal dependencies. Not all mobility faults will happen instantaneously, and in fact it is precisely the small, gradually changing precursors in behavior that a model should ideally detect before a full-blown mobility failure.

To address this limitation, transformer layers are incorporated into the autoencoder architecture to learn contextual signals (Wang et al., 2022; Najafi et al., 2024). By operating on windowed sequences of data, the model is able to learn relationships across multiple timesteps. Transformer layers use an attention mechanism, allowing the model to assign higher priority to relevant timesteps within the input sequence. This enables the network to capture temporal dependencies and complex sequential relationships that a standard autoencoder would struggle to capture.

We employ such a transformer autoencoder model using time-series data gathered during the ERNEST test campaign. In addition to capturing temporal structure, careful consideration is given to the different data types present in the test set. While most channels contain continuous numerical data (e.g. voltages, velocity), others represent discrete state variables, where an integer corresponds to a specific rover state. A simple example is the binary on/off state of a component. A change in state should not be treated in the same way as a fluctuation in a continuous numerical value.

To address this, the data is separated into categorical and numerical channels. One-hot encoding is applied to the categorical data so that state transitions are represented as discrete classes rather than numerical values. Due to computational and time constraints, only binary categorical variables are included in this implementation. The reconstruction loss is then calculated differently depending on data type. For categorical channels, cross-entropy loss is used, calculating the difference between predicted and actual probabilities. For numerical channels, mean squared error (MSE) is used to quantify differences between reconstructed and true values. The overall reconstruction loss is computed as a weighted combination of the categorical and numerical components, ensuring that both terms contribute appropriately to the final anomaly score.

The overall transformer autoencoder architecture is shown in

Figure 1. The encoder compresses each input window into a latent representation using transformer layers, and the decoder reconstructs the original sequence from this representation. Through this design, both temporal dependencies and heterogeneous data types are incorporated into the anomaly detection framework.

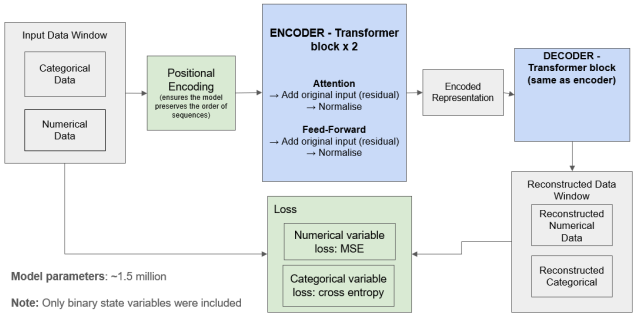


Figure 1. Architecture of Transformer Autoencoder

Each windowed sequence then has a corresponding reconstruction loss after being fed through the model. Higher reconstruction losses indicate that a sequence is anomalous. Here, the term anomaly score refers to this reconstruction loss value.

To train the model on an assumed baseline of nominal mobility behavior, we split the field test data into a nominal training set and a test set that encompasses the events annotated in the field notes. The test set is constructed by placing a five-minute buffer on either side of each recorded anomaly event, and the remaining data is allocated to training. Although this does not perfectly isolate anomalous mobility events, since faults may have occurred that were neither known nor recorded, it provides a reasonable approximation when working with complex field data and imprecise field notes. This data sorting is analogous to our proposed initial use of operational telemetry from early mission stages. Under this setup, the model is trained to reconstruct nominal telemetry windows, while anomalous windows are expected to reconstruct poorly and therefore yield elevated anomaly scores.

2.2. Supervised Anomaly Detection

The supervised anomaly detection approach builds on the same telemetry processing infrastructure used for the unsupervised analysis, but replaces field-test data with simulation-generated telemetry and explicit anomaly labels. To support this, we use the MuSE simulation environment (Bandyopadhyay et al., 2024), which provides a controlled setting for generating rover telemetry under both nominal and off-nominal operating conditions. In this work, we focus on the low-fidelity (LoFi) MuSE configuration, which enables the generation of a sufficiently large dataset for supervised learning while retaining relevant mission-level context for rover operations. The overall goal of this workflow is to establish an end-to-end path from anomaly injection in simulation to labeled dataset generation for model training and onboard model inference within the ROS-based autonomy stack, where such SHM models interface to other onboard subsystems.

2.2.1. Dataset Generation

To construct the supervised dataset, we define a suite of injected anomalies corresponding to controlled departures from nominal rover behavior. These anomalies are introduced directly into the MuSE simulation environment, which models rover subsystems including thermal behavior, power consumption, and battery charging dynamics. In the low-fidelity configuration used in this work, these subsystems are represented through parameterized tables describing nominal device heating rates, power draw, and battery charge rates.

During nominal rover operations within the simulated task network, only a subset of rover devices are active. In particular, 21 of the 43 modeled devices participate in the preheat and drive tasks used in the simulations. Anomaly injections therefore operate on this subset of active devices.

Anomalies are introduced by perturbing the nominal subsystem parameters during simulation. Specifically, multiplicative scaling factors are applied to thermal heating rates, power draw values, or battery charge rates associated with rover devices. These perturbations emulate departures from expected subsystem behavior, such as excessive heating, reduced heating rates, abnormal power consumption, or deviations in battery charging behavior. The dataset includes both single-device and multi-device anomaly cases, with anomalous behavior represented as either increases or decreases relative to nominal values. This anomaly generation approach can be generalized to a wide variety of signals, so long as the underlying simulator is representative of operational conditions.

Anomaly parameters are generated offline for each simulation run. For every injected anomaly, a start time and duration are sampled within the simulation timeline, subject to guard intervals near the beginning and end of the run. Perturbation magnitudes are drawn from a Gaussian distribution centered on the nominal value (e.g. $\mu = 1.3$, $\sigma = 0.5$) and applied as multipliers to the corresponding thermal or power tables. This produces a set of events that together capture a range of anomaly severity while preserving the structure of the nominal system model.

During simulation execution, anomaly scenarios are enforced through a runtime control mechanism implemented using PyContract (Dams et al., 2022) reactors. Each simulation run is associated with a scenario definition file specifying the anomaly types, affected devices, perturbation scales, and activation intervals. At runtime, the reactor monitors the simulation timeline and dynamically modifies the relevant thermal or power

tables when an anomaly interval begins. When the interval ends, the nominal tables are restored. All anomaly events are logged alongside the generated telemetry, enabling precise labeling of anomalous intervals in the resulting dataset.

For this prototype implementation of the supervised pipeline, all anomalous runs are grouped into a single *off-nominal* class, producing a binary classification problem between nominal and off-nominal telemetry.

Table 1 summarizes the generated supervised dataset. In total, this low-fidelity dataset contains 184 simulation runs, comprising 10 nominal runs and 174 off-nominal runs. Although this run-level representation appears strongly skewed toward anomalous cases, the table does not reflect the effective class balance present in the underlying time-series data. Injected anomalies are active for only portions of the off-nominal runs, so when measured in terms of total simulated time the dataset remains dominated by nominal behavior.

Each anomalous run is associated with a generated simulation record, including the perturbed device or devices, anomaly start and end times, scaling factor, and the corresponding scenario and parameter-configuration files. This provides direct traceability from each training example back to its injection specification and makes it possible to later stratify anomaly detection and isolation results by anomaly family, perturbation magnitude, or subsystem.

Simulation runs are executed using an automated pipeline on a high-performance computing cluster. The dataset consists of both nominal and off-nominal simulations executed across multiple environmental and operational configurations, including variations in waypoint sequences, initial battery state-of-charge, and both day and night operating conditions. Across all runs, the generated dataset contains approximately 40 hours of simulated rover operation and roughly 200 GB of telemetry stored as ROS bag files. These bags contain the full rover telemetry produced during simulation, including thermal and power functional layer outputs, device state information, and additional subsystem telemetry streams generated by the autonomy stack.

2.2.2. Supervised Modeling Infrastructure

The primary objective of this work is to demonstrate the end-to-end anomaly detection pipeline rather than to optimize model architecture. The framework is designed to support future experimentation with more sophisticated models and higher levels of detection fidelity, including subsystem-level detection (e.g. thermal versus power anomalies), per-component anomaly identification, and classification of anomaly types such as elevated thermal rates or abnormal power consumption. To this end, we describe a setup which establishes the simulation and model training infrastructure, training and metrics protocols, and interfaces required to produce appropriately complex models as a mission gains access to higher fidelity simulators during mission design and actual anomaly signatures during operation.

Telemetry generated by MuSE simulations is recorded as rosbag files and processed through a machine learning pipeline. The pipeline extracts relevant telemetry channels, aligns timestamps across topics, and resamples the data to a uniform cadence of 1 Hz. Topic streams are merged using the simulation timestamp as a common index, ensuring consistent temporal alignment across telemetry sources. Features are then assembled from the thermal and power functional layers, including per-device temperatures, power modes, remaining preheat durations, and battery energy and state-of-charge measurements. Array-valued telemetry fields are expanded into indexed scalar features corresponding to individual devices. This produces a feature vector of 174 dimensions, consisting of two power features (battery energy and battery state-of-charge) together with 172 thermal features derived from 43 indexed device slots, including temperatures, power modes, remaining preheat durations, and device identifiers.

The machine learning workflow is implemented using MLflow to manage experiment configuration, model training, and reproducible evaluation runs. This modular pipeline allows different model architectures and feature configurations to be evaluated using the same dataset and preprocessing steps.

Binary anomaly labels are derived from simulation anomaly metadata generated during MuSE execution. Each injected anomaly is defined by a start and end time in the simulation scenario configuration. During preprocessing, these intervals are expanded into per-timestep labels, producing a binary target variable indicating whether an anomaly is active at a given simulation time. Because anomalies occur only during portions of a run, both nominal and anomalous intervals may appear within the same simulation trajectory.

Training and evaluation datasets are constructed by splitting simulations at the scenario level rather than by individual timesteps. This ensures that telemetry from a given simulation run does not appear in both training and evaluation sets, preventing temporal leakage and more closely reflecting deployment conditions where models encounter previously unseen runs.

For the prototype supervised anomaly detector, we use a binary classification model that predicts whether the rover is operating under nominal or anomalous conditions. The model takes telemetry-derived features as input and outputs the probability that an anomaly is present.

The classifier is implemented as a logistic regression model using the scikit-learn library with L2 regularization and class weighting to mitigate label imbalance. Given input features $x_1, \ldots, x_n$, the model computes a weighted linear combination of the input features

Table 1. Summary of the low-fidelity simulation dataset used to train supervised models.

Run family	# runs	Variant	Perturbation
Nominal baseline & waypoint variants	10	Nominal	—
Temp., 1 device, initial preheat	21	Off-nominal	Increase
Temp., 1 device, initial preheat	21	Off-nominal	Decrease
Temp., 1 device, full preheat/drive	21	Off-nominal	Increase
Temp., 1 device, full preheat/drive	21	Off-nominal	Decrease
Power, 1 device	20	Off-nominal	Increase
Power, 1 device	20	Off-nominal	Decrease
Temp., 4 devices	6	Off-nominal	Increase
Temp., 4 devices	6	Off-nominal	Decrease
Power, 4 devices	6	Off-nominal	Increase
Power, 4 devices	6	Off-nominal	Decrease
Battery production rate	10	Off-nominal	Increase
Battery production rate	10	Off-nominal	Decrease
Total	**184**		

$$z = w_0 + \sum_{i=1}^{n} w_i x_i \tag{1}$$

which is mapped to an anomaly probability using the sigmoid function

$$p = \frac{1}{1 + e^{-z}}. \tag{2}$$

Model parameters are learned using labeled telemetry segments generated by the simulation pipeline and optimized using cross-entropy loss. The resulting classifier serves as a computationally lightweight baseline suitable for real-time inference within the rover autonomy stack. The current implementation focuses on binary anomaly detection, where the model predicts whether the rover is operating nominally or off-nominally.

2.2.3. ROS Deployment

The Robot Operating System (ROS) provides a distributed middleware for inter-process communication, where system components (nodes) exchange data via message streams. To evaluate how the trained model can be used in an operational setting, we implemented a ROS node that applies the trained classifier to real-time telemetry streams using the same input feature structure used during training. This provides a deployment pathway from the supervised-learning pipeline to a mission-representative runtime environment. Onboard the rover platform, telemetry is produced by the existing stack, passed through ROS topics, and consumed by a dedicated anomaly-detection node. This node is responsible for assembling model inputs from the incoming telemetry, executing the trained classifier, and publishing a predicted rover health state at a fixed rate. For linear models such as logistic regression, the node can additionally report the most influential telemetry variables contributing to each prediction by computing the signed contribution of each feature to the anomaly score.

In the current design, the anomaly-detection node consumes telemetry from a selected set of ROS topics and emits an anomaly state at 1 Hz, together with an anomaly score and related explanatory outputs. The ROS topics consumed and published by the node are listed in Table 2. The node is intentionally model-agnostic at the interface level, so the ROS-side deployment scaffold can remain stable even as the trained supervised model evolves. This allows improvements to model architecture, training strategy, or feature representation without requiring major changes to the ROS integration.

3. RESULTS

3.1. Unsupervised Anomaly Detection Results

Evaluating the unsupervised model using field operator notes revealed particular challenges. The labels were entered manually into a separate document during the field tests, meaning their exact timing was not guaranteed. In our test set, each anomaly event begins five minutes before the corresponding field note and ends five minutes after it. A mobility anomaly is considered detected (true positive) if the model's anomaly score spikes within three minutes of the manually recorded event time. Table 3 shows the results of the model for the field campaign test data. Of the 18 candidate anomalies, 5 are considered detected using this method. Negative times indicate that the anomaly score spiked before the recorded field note. Of the 13 "stuck in" notes, only 2 are detected. However, in both cases the actual anomaly occurred after the time recorded in the note, based on inspection of the telemetry. This means that the model's detections may be unrelated to the stuck behavior observed by field operators.

Table 2. ROS topics consumed and published by the supervised anomaly-detection node.

Topic	Message type	Direction
`/power/power_state`	`tlr_system_msgs/PowerState`	in
`/thermal/thermal_state`	`tlr_system_msgs/ThermalStateSummary`†	in
`/shm/anomaly_score`	`std_msgs/Float32`	out
`/shm/anomaly_state`	`std_msgs/Bool`	out
`/shm/anomaly_threshold`	`std_msgs/Float32`	out
`/shm/anomaly_topk`	`std_msgs/String`	out

† `ThermalStateSummary` contains a `ThermalDeviceState[]` array, with one entry per device.

Table 3. Mobility Field Events and Prediction Timing

Field Note	Count	Detected	Time (min)
Stuck in (bush, gulch, rock, ditch, obstacle, tree)	13	2	1.48 (avg)
Robot was getting a wee tippy on the slope	1	1	-2.73
Wheel 1 stopped spinning	1	0	N/A
Robot tipped	1	0	N/A
Front left wheel stuck in soft sand	1	1	-2.05
Rover almost tipped	1	1	-0.37
AVERAGE ALL	**18**	**5**	**-1.07**

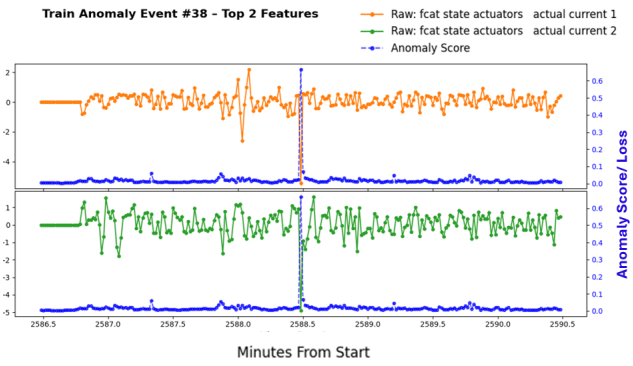


Figure 2. Example of anomaly spike in the training set.

This suggests that the unsupervised model largely does not capture these "stuck in" anomalies. This outcome is not necessarily negative, particularly as the failure to detect anomalies occurred consistently across all "stuck in" cases. Instead, it indicates that the available data streams may not contain sufficient information to identify when the rover is "stuck" in this manner. This is further complicated by the fact that during the field campaign, human intervention occurred when the ERNEST rover encountered obstacles of this type.

Figure 2 illustrates one example of a flagged anomaly event in the training data. It shows a sudden spike in two of the recorded actuator currents, corresponding to a sharp increase in the anomaly score. Although this particular current spike may not represent an actual fault, its sudden change in value is the type of behavior that an anomaly detection system should be capable of identifying. If such events represent normal system operation, then future work should consider enacting similar maneuvers in field tests. Indeed, larger training sets may help a model to interpret more rare events as natural variations, thereby reducing the amount of false positives.

Running the transformer autoencoder model on all the data, both training and test, reveals anomaly spikes outside of the field-operator-annotated mobility events. Figure 3 shows anomaly scores across the entire dataset. An anomaly event is defined by grouping together anomaly-score spikes greater than 0.7 that occur within 5 seconds of one another. 48 anomaly events were identified using this method by running the model on the entire dataset. Judging whether or not each event is truly anomalous would require additional data analysis, but at a minimum it illustrates timestamps where the model indicates that behavior deviates from the norm. On the other hand, these could be entirely nominal events but the model is failing to reconstruct them because the training data does not contain sufficient similar examples. Since this exercise is limited to data from an initial field test, we cannot determine this conclusively; however, in a real mission, such anomaly spikes could indicate novel behaviors.

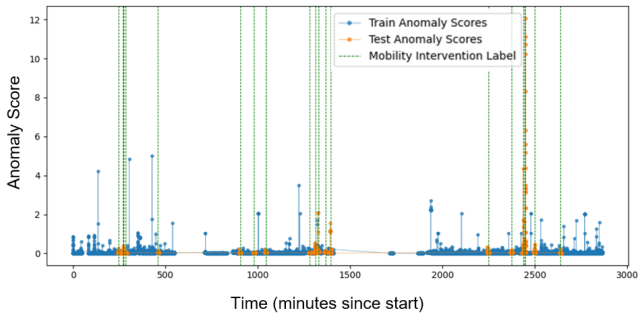


Figure 3. Anomaly scores across entire train (blue) and test (orange) set. Anomalies detected in both the train and test set. Green lines indicate the timestamps of the 18 detailed anomaly events in Table 3.

The two non-stuck events which were not detected were 'Robot tipped' and 'Wheel 1 stopped spinning'. However, further in-

vestigation reveals that the tipped event, which is an extreme mobility fault, occurs 15 minutes before the field note is actually made. It appears that the rover is turned off immediately after it tipped, with the field note being made after it is turned back on. This highlights practical challenges in model assessment using real field test data. Figure 4 illustrates this event and the anomaly spike corresponding to when the rover actually tipped. This means the model does detect the tipping, just outside of the field note window defined at the outset. When investigating the Wheel 1 event, it can be seen that there are some spikes in the anomaly score two minutes before the noted event time, however these spikes are considered too low to be classified as an anomaly event under the selected threshold. This highlights the inherent challenge in selecting an appropriate threshold for the anomaly score. As this work is a limited proof-of-concept, we did not explore dynamic or adaptive threshold methods, but those techniques can be used to optimize model precision and recall.

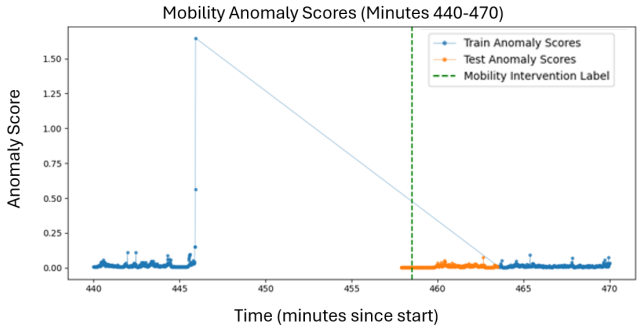


Figure 4. Spike in anomaly score before rover tipping event (event noted at timestamp with green line)

To evaluate increases in anomaly scores in relation to the mobility events, it is important to identify the specific features that cause a timestamp to be flagged as anomalous. This helps reveal what the model is detecting as abnormal behavior and whether it can be attributed to the mobility fault. Since an autoencoder works by reconstructing input features, explainability can be incorporated by examining which features have the highest reconstruction loss and therefore contribute most to a timestamp being flagged as anomalous. Table 4 shows the top contributing channels for the detected anomaly events. This provides a basic level of explainability as to why the model produces detections at certain times, and indicates channels and components that ought to be investigated. With 332 total channels, this is especially crucial. The robot tipping event provides a clear example of realistic channels being flagged, with angular rotation as the number one contributor. This makes sense as a tipping action affects the rotation of the rover. Similarly, for the "tippy" event, the yaw is considered to be the key factor. The almost tipped event is more nuanced, with modes of operation emerging as the key channels. What the model is actually picking up in this case is likely manual intervention during this event. These results are encouraging as the model was not manually trained on labeled mobility anomaly events, yet it is still able to flag several mobility-related events and highlight plausible contributing channels.

Table 4. Top contributing data channels for each flagged mobility event

Event	Contributing Feature	%
Robot getting tippy	Yaw	35
	Actuator 7 cmd pos	17
	Joint 2 velocity	14
Robot tipped	$\Delta\theta$ (angular rotation)	21
	IMU rate	20
	Angular rate	17
Front left wheel stuck	Actuator 6 cmd pos	45
	Actuator 6 EGD cmd pos	45
	Actuator 5 cmd velocity	3
Rover almost tipped	Actuator 4 mode of operation	18
	Actuator 3 mode of operation	18
	Actuator 2 mode of operation	17
Rover stuck in rocks	Actuator 3 cmd velocity	18
	Actuator 0 cmd velocity	18
	Actuator 3 actual velocity	17

These results demonstrate how an anomaly detection model can be used to flag anomalous events. The reconstruction loss of specific channels can provide insights into why the model predicts an anomaly. This explainability is essential when considering the implementation of a real-world anomaly detection system which needs to provide response mechanisms with actionable information. Future work should also address challenges associated with variables that have multiple non-binary states, as well as continuous variables that naturally exhibit spike patterns, like the current spike shown in Figure 2. One of the main motivations for using a machine learning model instead of threshold-based monitors is to pick up anomalous patterns which are not necessarily just data spikes, especially in the context of detecting strange behavior before more catastrophic faults can occur. Therefore, we emphasize metrics which capture time-to-detection.

3.2. Supervised Anomaly Detection Results

The supervised modeling results evaluate two aspects of the proposed framework: baseline anomaly-detection performance on the synthetic dataset, and successful deployment of the trained model as a ROS-integrated inference node during live

rover operation. In this work, the emphasis is on demonstrating the feasibility of the end-to-end supervised pipeline rather than on optimizing anomaly-classification performance during operation.

3.2.1. Model Performance on Synthetic Dataset

Table 5 summarizes the performance of the binary anomaly detector on the synthetic dataset. Across all anomaly categories, the model achieves high true positive rates (TPR $\geq$ 0.926), indicating that the injected anomalies are generally detected by the classifier. However, precision remains comparatively low, resulting in modest F1 scores. This behavior is expected given the strong imbalance between nominal and anomalous telemetry samples, together with the use of a simple baseline classifier and a fixed decision threshold.

Detection latency is also low, with anomalies identified within approximately 1–2 seconds on average. This corresponds to roughly one to two telemetry update cycles at the 1 Hz sampling rate used by the system, indicating that the model is capable of responding quickly to deviations in the simulated telemetry streams. These results provide a baseline for future improvements in model architecture, feature representation, and threshold selection.

3.2.2. ROS Deployment and Operational Demonstration

The second stage of evaluation examines whether the trained supervised detector can be integrated into the real-time ROS-based rover autonomy stack and operate alongside the existing autonomy components. In addition to training a model on synthetic telemetry, this required connecting the detector to the runtime messaging architecture used during field testing and verifying that it can consume real-time telemetry and publish inference outputs in real time.

During the field demonstration, the ERNEST system was operated using a three-machine architecture consisting of the rover platform, a ground-control workstation, and an auxiliary development computer. The rover produced telemetry from the autonomy stack, which was transmitted through the ROS network to the other machines in the system. The auxiliary computer hosted a virtual power and thermal subsystem used to emulate environmental conditions and system behavior during the demonstration, including scenarios such as lunar night operation.

For the purposes of this experiment, the anomaly detection node was executed on the auxiliary computer, where it consumed the same telemetry streams used by the rover autonomy system. The node operated within the ROS network alongside the other system components and published anomaly score, anomaly state, threshold, and explanation outputs. Figure 5 shows the ROS-integrated deployment architecture used in the field test.

During active field operation, the detector output stream operated at the intended 1 Hz cadence. The median update interval was 1.000 s, with a 95th percentile interval of 1.025 s. The effective telemetry rate supplied to the detector also remained at 1 Hz, indicating that the deployed node remained synchronized with the incoming power and thermal subsystem telemetry streams throughout the demonstration.

Because the anomaly detection node was executed on an auxiliary system rather than the rover onboard computer, and the ROS field deployment was intended to validate software integration rather than provide a separately labeled evaluation dataset, the reported timing results characterize the end-to-end ROS deployment rather than standalone onboard inference latency or classification accuracy. Nevertheless, they demonstrate that the detector operated stably in real time within the field-test environment and that the full supervised workflow, from simulation-based anomaly generation through offline model training to ROS-based live inference, was successfully exercised in a field-test runtime setting.

4. DISCUSSION

As space missions increasingly incorporate machine learning into onboard autonomy, they often cannot rely on historical mission data due to changes in architecture and operational environment, or because they may be exploring a remote destination for the first time. In addition, inherent gaps in developing a system for a non-terrestrial environment mean that model training data may only be available from high-fidelity physics-based simulators and limited experimental results. This makes leveraging supervised learning particularly challenging because it requires datasets that capture the range of ways a given anomaly may appear in telemetry. Once the mission enters operations, telemetry from real anomalies can be downlinked to ground teams to train supervised learning models, thereby improving supervised model performance. However, it is not realistic to assume that simulation, experimental, and operational telemetry datasets capture the full range of possible anomalous events. Accordingly, mission architects should consider combining supervised and unsupervised anomaly detection models with model-based approaches (e.g., physics- or behavior-based reasoning) to enable robust detection and isolation.

Unsupervised models avoid the need for labeled anomaly data because they can be trained using only nominal telemetry. As a result, the cost of expensive high-fidelity simulations, representative experimental tests, and telemetry downlinks can be reduced by focusing on nominal data. This is particularly important for long-duration missions, which may accumulate substantially more nominal data over successive operational cycles, enabling periodic retraining of unsupervised models to continuously improve onboard anomaly detection capabilities. In this work, the unsupervised analysis focused

Table 5. Binary anomaly detector performance on the synthetic test dataset.

Anomaly	TPR	FPR	Precision	F1	Mean Det. Time (s)
High Temp Rate	0.986	0.0176	0.121	0.216	1.58
Low Temp Rate	1.000	0.0202	0.158	0.238	1.27
High Power Draw	1.000	0.0240	0.101	0.184	2.14
Low Power Draw	1.000	0.0218	0.115	0.205	1.89
High Battery Charge	0.926	0.0199	0.133	0.233	1.88
Low Battery Charge	0.952	0.0207	0.142	0.247	1.95

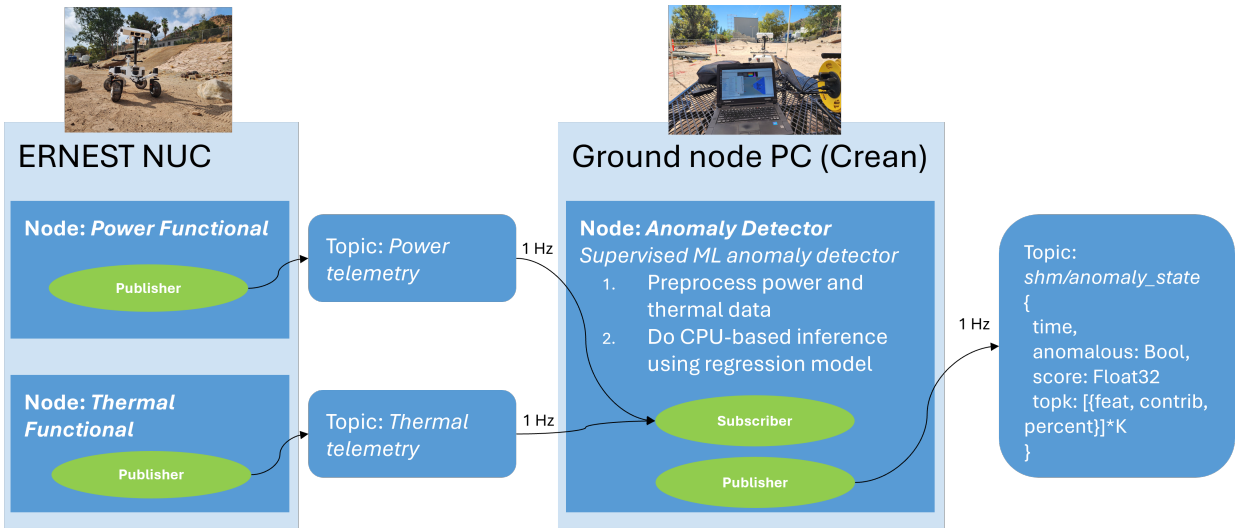


Figure 5. ROS-integrated deployment architecture used during the supervised field demonstration. Telemetry from the rover autonomy stack was distributed across the ROS network, the anomaly detection node consumed the relevant power and thermal streams, and the node published anomaly score, anomaly state, threshold, and explanatory outputs.

on the mobility subsystem as a representative test case. Although terrain and mechanics will differ in the real mission, this setting provides a useful basis for understanding which types of anomalies can be detected from the available onboard telemetry. The unsupervised transformer-based autoencoder produced anomaly detections that corresponded to several events recorded in the operator field notes and also detected additional cases that were not labeled by operators but still showed evidence of abnormal behavior upon inspection of the telemetry streams.

If sufficient labeled anomaly data is available (e.g., from off-nominal simulation runs or field testing), supervised models can support more targeted anomaly identification and isolation. In this work, the supervised approach leveraged labeled anomaly data to generate anomaly predictions and identify contributing telemetry channels. The supervised pipeline produced a binary detector for power and thermal anomalies and demonstrated a deployment path from simulation-based training to ROS-based live inference during field operations.

This work highlights some practical challenges in simulator-based dataset generation and the use of field-test data for model training. Field-test data may require considerable preprocessing before unsupervised models can be trained, and these experiments may include operator interventions that interrupt capture of the full progression of an event. Building such unsupervised models also requires careful selection of telemetry channels, including whether to aggregate signals from existing monitors, rely entirely on measurement data, or combine the two. The associated choice of model architecture is also particularly important for signals that require longer contextual history. In practice, onboard deployment of machine learning models requires sufficient compute and memory resources, and the interfaces that provide input features to the model must be designed to operate reliably to ensure real-time responsiveness. Finally, to avoid brittle model failure when any single input channel becomes unavailable, the models should be trained on datasets that include such conditions.

Despite these challenges, learning-based SHM has many advantages, particularly as more relevant mission data become available through improved physics simulators, missions with repeating ConOps, and long-duration missions. These models can operate on both command streams and measurement telemetry from sensors and subsystems. As a result, they can

capture patterns in telemetry that reflect both expected and unexpected system behavior. Such capabilities go beyond local threshold-based monitors and motivate mission-specific trade studies to determine how best to combine data-driven and physics-informed SHM approaches.

5. CONCLUSION

Autonomous lunar rover missions will require SHM approaches that can detect anomalous behavior using limited pre-launch data and evolving operational telemetry. This work investigated complementary supervised and unsupervised learning methods for rover anomaly detection using field-test data, simulation-generated anomaly data, and ROS-based deployment.

For the unsupervised approach, data from a 30-hour ERNEST field campaign were used to train a transformer autoencoder for mobility anomaly detection. The model detected 5 of 18 manually recorded candidate mobility anomalies, with further analysis showing that raw detection counts were affected by imprecise field-note timing and by whether the selected telemetry channels captured the relevant failure mode. The model also identified additional anomaly-score spikes outside the annotated events, and per-channel reconstruction loss provided explanatory information linking several detections to plausible mobility-related signals, including angular rotation and actuator behavior.

For the supervised approach, low-fidelity MuSE simulation data were used to train a binary detector for power and thermal anomalies, which was then deployed as a ROS-integrated inference node during live rover operation. On the synthetic test dataset, the classifier achieved high true positive rates across anomaly categories (TPR $\geq$ 0.926), with mean detection times of approximately 1–2 s, although precision and F1 scores remained modest. During the ROS field demonstration, the detector operated at the intended 1 Hz cadence, demonstrating the full supervised workflow from simulation-based anomaly generation through offline model training to ROS-based live inference.

Together, these results demonstrate complementary paths for learning-based rover SHM, with supervised models detecting known off-nominal behaviors from labeled data and unsupervised models flagging unexpected deviations from nominal telemetry. Future work should focus on higher-fidelity simulation, larger and more systematically labeled field datasets, improved thresholding and model architectures, and tighter integration between anomaly detection, explainability, and onboard response.

ACKNOWLEDGMENT

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration and funded in part through the internal Research and Technology Development program. This work was also supported by the 2025 New Zealand Space Scholarship, provided by the New Zealand government. © 2026 California Institute of Technology. Government sponsorship acknowledged.

REFERENCES

Baker, J. D., Elliott, J. O., Keane, J. T., Khan, N. R., Kornfeld, R. P., Nayar, H. D., & Nesnas, I. A. (2024). The Endurance lunar rover sample return mission. In *IEEE aerospace conference proceedings*. IEEE Computer Society. doi: 10.1109/AERO58975.2024.10520939

Baker, J. D., Stone, H. W., Elliott, J. O., Keane, J. T., Kornfeld, R. P., Nayar, H. D., & Nesnas, I. A. (2025). The Endurance mission progress. In *IEEE aerospace conference proceedings*. IEEE Computer Society. doi: 10.1109/AERO63441.2025.11068531

Bandyopadhyay, S., Gaut, A., Rouquette, N., Jain, A., Amini, R., Hasnain, Z., ... Ingham, M. D. (2024). Endurance mission-level simulation architecture for autonomy development. In *AIAA aviation forum and ASCEND 2024* (p. 4834).

Dams, D., Havelund, K., & Kauffman, S. (2022). A Python library for trace analysis. In T. Dang & V. Stolz (Eds.), *Runtime verification* (pp. 264–273). Cham: Springer International Publishing.

Hartman, C. N. (2021). Planetary science and astrobiology decadal survey 2023-2032. *NSF Award Number 2040016. Directorate for Mathematical and Physical Sciences*, *20*(2040016), 40016.

Ingham, M. D., Hasnain, Z., Amini, R., Ardito, S., Bandyopadhyay, S., Bocchino, R., ... Rouquette, N. (2024). Onboard planning and execution of mobility and telecommunications for the endurance lunar rover. In *AIAA Aviation Forum and ASCEND 2024* (p. 4889).

Lakhmiri, D., Alimo, R., & Digabel, S. L. (2022, 3). Anomaly detection for data accountability of Mars telemetry data. *Expert Systems with Applications*, *189*. doi: 10.1016/j.eswa.2021.116060

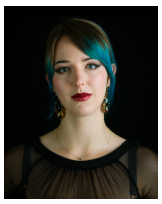
Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, *7*(66), eabm6074. doi: 10.1126/scirobotics.abm6074

Najafi, S. A., Asemani, M. H., & Setoodeh, P. (2024). *Attention and autoencoder hybrid model for unsupervised online anomaly detection.* Retrieved from `https://arxiv.org/abs/2401.03322`

Sabzehi, M., & Rollins, P. (2024). Enhancing rover mobility monitoring: Autoencoder-driven anomaly detection for Curiosity. In *IEEE aerospace conference proceedings*. IEEE Computer Society. doi: 10.1109/AERO58975.2024.10521179

Wang, X., Pi, D., Zhang, X., Liu, H., & Guo, C. (2022, 3). Variational transformer-based anomaly detection approach for multivariate time series. *Measurement: Journal of the International Measurement Confederation*, *191*. doi: 10.1016/j.measurement.2022.110791

BIOGRAPHIES



Sofie Claridge received her B.Sc. and M.Sc. in Computer Science from Victoria University of Wellington, New Zealand. She is currently pursuing her Ph.D. at the Robinson Research Institute, where her research focuses on machine learning techniques for enhancing optical sensing in superconducting magnets. Her internship at JPL was supported by the New Zealand Space Scholarship.

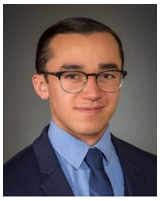
Jack Patterson received his B.E. (Hons.) in Software Engineering from the University of Canterbury, New Zealand. He is currently pursuing his Ph.D. in Computer Science at the University of Canterbury, where his research focuses on deep learning methods for observational studies of the outer Solar System. His internship at JPL was supported by the New Zealand Space Scholarship.

Zaki Hasnain received his B.Sc. in Engineering Science and Mechanics from Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA. He received his M.Sc. in Computer Science and Ph.D. in Mechanical Engineering from the University of Southern California, Los Angeles, California, USA.

Ashish Goel is a robotics technologist in JPL's Robotic Surface Mobility group. He received his Masters and Ph.D. in Aeronautics and Astronautics from Stanford University and his bachelor's degree in Engineering Physics from Indian Institute of Technology Bombay.

Katharine Patterson completed her M.Sc. in Robotics, Systems and Control at ETH Zurich, Switzerland, in 2026 and received her B.Sc. in Computer Science from Brown University, Providence, Rhode Island, USA, in 2015.

Nicolas Rouquette is a Principal Computer Scientist at JPL. He received his Master's and Ph.D. in Computer Science from the University of Southern California, Los Angeles, USA, and an ingénieur diploma from the Ecole Supérieure d'Ingénieurs en Électricité et Électrotechnique, France.



Caleb Wagner received his B.Sc. in Robotics Engineering and B.Sc. in Computer Science from Worcester Polytechnic Institute in Worcester, Massachusetts. He received his Master of Science in Computer Science from Georgia Institute of Technology in Atlanta, Georgia.

Tristan D. Hasseler is a Robotics Technologist at JPL. He focuses primarily on multibody dynamics simulation. He received his B.S. in Aeronautics and Astronautics from the University of Washington. He received his M.S. in Aeronautics and Astronautics with a focus on Controls from Stanford University.

Michel Ingham received his Bachelor's degree in Honours Mechanical Engineering from McGill University in Montreal, Canada. He received his Master of Science and Doctor of Science degrees from the Massachusetts Institute of Technology's Department of Aeronautics and Astronautics (Cambridge, Massachusetts, USA).