CEAS Space Journal
https://doi.org/10.1007/s12567-024-00589-1

ORIGINAL PAPER

Synthetic satellite telemetry data for machine learning

Clemens Schefels[1] · Leonard Schlag[1] · Kathrin Helmsauer[1]

Received: 28 March 2024 / Revised: 18 December 2024 / Accepted: 19 December 2024

Abstract
For many machine learning tasks, labeled data are crucial. Even though there are methods that can be trained with data with only few labels, most of the tasks require many labels. In satellite operations, a huge amount of data are generated by the telemetry parameters of a satellite that keep track of its status. Modern satellites collect telemetry data of thousands of parameters. For example, the GRACE Follow-On satellites, operated by the German Space Operations Center (GSOC) at the German Aerospace Center (DLR), define about 80,000 unique housekeeping parameters each. However, all these telemetry data lack a complete/holistic set of labels. These data are usually unpredictable, hard to reproduce, and very diverse. As a consequence, expert knowledge is necessary to label these data, e.g., with anomalies. Moreover, labeling data by hand can be very time-consuming and, therefore, expensive. To overcome these obstacles, we implemented a synthetic satellite telemetry data library that is able to (a) generate a large variety of telemetry-like data, (b) add a plethora of well-defined anomalies to these data, and (c) deliver the labels for these injected anomalies. With these data, we are now able to train, validate, and test our machine learning models. Furthermore, we can compare different models with reproducible data. Since satellite telemetry data are often strictly confidential, we can share these synthetic data easily with our research partners.

Keywords Satellite telemetry · Machine learning · Anomaly detection · Synthetic data · Labeled data · Software development

1 Introduction

Nowadays, the usage of Machine Learning (ML) software tools is routine in many disciplines, also in the space domain. These tools are able to analyze huge amount of data and unburden humans from monotonous tasks. Many of the ML software tools use statistical methods to learn from examples, i.e., from labeled data. In our field of expertise—satellite operations—anomaly detection in satellite telemetry data is a typical use case. Here, ML software tools are scanning through huge amount of telemetry data and check for anomalous behavior, which is traditionally a task of system engineers. However, to implement such ML tools, labeled data is needed. Those labels, which indicate nominal and anomalous data, represent the examples from which the tool learns. With such a data set, the ML model of the tool can be trained, so that it learns how nominal data looks like and is then able to distinguish anomalous from nominal behavior. In general, huge amounts of satellite data are already available but, most of the time, they lack labels. And since labeling data by hand is a very time-consuming and therefore expensive task, the need for an automatic solutions is given. As described in Sect. 2, some labeled telemetry data sets can be found online. However, these data sets are the results of former projects and built for a specific use case. They may not completely cover the need of new projects and may need to be adapted; again a time-consuming task. Therefore, a more customizable solution is needed.

In this article, we present our Python library for the generation of synthetic satellite telemetry data. In particular, we focus on its flexible and generic architecture. With this library, we can create data sets for ML training that contain various types of telemetry data. Inspired by the possibility to inject different kinds of anomalies into the generated telemetry data and document them by labels, we present use cases for miscellaneous scenarios. The generated training and validations data sets, for example, are suitable for many kinds of ML or deep learning applications. Moreover, only with a diverse and large set of labeled data, ML models can

✉ Clemens Schefels
clemens.schefels@dlr.de

[1] German Aerospace Center (DLR), German Space Operations Center (GSOC), 82234 Weßling, Bavaria, Germany

Published online: 11 January 2025

be trained more robust and make the initial training easier and more generic. Such generated models can be transferred and retrained for specific use cases. Another use case would be the test of already existing ML software tools on the one hand as a benchmark tool and, on the other hand, as quality check for already-in-production software, where effects of model updates have to be controlled. With these use cases, we demonstrate the capabilities of our synthetic satellite telemetry data library.

This paper is structured as follows: in Sect. 2, we present related work and focus on publicly available data sets as well as on open source frameworks which can generate synthetic telemetry data. The next Sect. 3 is dedicated to our library for synthetic satellite telemetry data with all its features and its derived architecture. How we use this library is described in Sect. 4 and demonstrated on actual use cases, like supporting the process of software development and the development our anomaly detection ML tool, ATHMoS (Automated Telemetry Health Monitoring System) [1, 2]. In Sect. 5 on Proofs of Concept, we demonstrate the capabilities of the library by mocking real satellite parameters and using these data to train an anomaly detection framework in two different scenarios. Future work in Sect. 6 gives a short outlook on planned future extensions of our library.

This article finishes with a summary and a conclusion in the last section, Sect. 7.

2 Related work

This section first gives an overview of publicly available data sets, a common source for labeled data sets to train and test ML models on. Further in this section, programming frameworks that generate time series data will be discussed. The focus of this section, as of the whole paper, will be on data similar to satellite telemetry.

2.1 Public data sets

As mentioned in the introduction, labeled data are necessary for training a ML model. A common source for these kinds of data are publicly available labeled data sets that can be easily accessed, e.g., by downloading them from the Internet. These data sets often originate from research projects or publicly funded projects and are derived from real world as well as synthetic systems. Since these public data sets are widely used, most of their downsides are known and discussed on public forums. One disadvantage of these data sets is that they stay fixed after being published and cannot be easily expanded, e.g., with more labels. Also, while most sources or data owners are well known and reliable, it is not a given that public data sets contain what is promised by their publishers. It is also not ensured that the data sets will be available in the future or that possible changes to the data sets by the data owners are tracked and previous versions remain available. Furthermore, one has to take the license under which the data set is distributed into account. It may actually forbid certain use cases, e.g., the training of commercial or military ML models. Lastly, the ownership of a model trained with public data sets may not to be clear. We want to provide a short overview of publicly available data sets relevant to the topic of this paper.

The first data set [3] derives from telemetry data of the European Space Agency's (ESA) *Mars Express* orbiter [4]. It was published about six years ago during a hackathon [5] to predict the average current in each of the 33 thermal power lines. The data consists of a training set containing three Martian years worth of context and electric current measurements data. The provided test set includes the context data only and entails, among other things, mission operations plan files, solar angles, and distances.

Notably, the *ESA Anomaly Dataset* [6], is providing a large-scale satellite telemetry data set annotated with curated anomalies from three ESA missions. Developed through an 18-month consortium project involving Airbus Defence and Space, KP Labs, and the European Space Agency's European Space Operations Centre (ESOC), this comprehensive data set enables benchmarking and validation of anomaly detection models and approaches [7, 8]. It is part of ESA's Artificial Intelligence for Automation (A^2I) Roadmap initiative [9], launched in 2021 to harness AI capabilities for automating space operations.

The project *Telemanom* utilizes LSTMs, implemented with TensorFlow (i.e., Keras), to detect anomalies in multivariate sensor data [10, 11]. It provides telemetry from real spacecrafts with labeled anomalies, the *Soil Moisture Active Passive* (SMAP) satellite [12] and the *Curiosity* rover on mars (MSL) [13].

The *Challenger USA Space Shuttle O-Ring* data set [14], published on the UC Irvine Machine Learning Repository [15], contains data related to the shuttle's O-ring during launch on 28 January, 1986. After the launch, the USA Space Shuttle Challenger exploded.

On the very same archive, a data set from a space shuttle landing control system [16] can be found, as well as the stats logs [17]. These space shuttle data sets are also available on the ODDS Web page [18], which provides access to a large collection of outlier detection data sets with ground truth.

For the sake of completeness, we want to mention four popular time series data sets not related to the space domain. The first one is the data set used for *The Third International Knowledge Discovery and Data Mining Tools Competition* [19], which was held in conjunction with *The Fifth International Conference on Knowledge Discovery and Data Mining* [20]. The competition task was to build a (computer) network intrusion detector. To this end, this data set

is composed of nine weeks of raw TCP dump data from a local-area network (LAN), which includes a wide variety of intrusions simulated in a military (computer) network environment.

The *UCR Time Series Classification Archive* [21, 22] provided by the University of California, Riverside, contains a wide variety of time series data sets from many different sources and is a common data set collection for benchmarking. It is preprocessed in the sense that the data is already normalized as well as split into a training and testing sets, the latter aiming towards reproducibility of the results.

Climate Data Online (CDO) provides free access to United States National Climatic Data Center's (NCDC) archive of global historical weather and climate data [23] in addition to station history information. These data include quality controlled daily, monthly, seasonal, and yearly measurements of temperature, precipitation, wind, and degree days as well as radar data and 30-year climate normals.

The last data collection is made up of 58 time series data files that are designed to provide data for research in streaming anomaly detection. This repository [24] contains the data and scripts which comprise the *Numenta Anomaly Benchmark* (NAB). It includes both real-world and artificial time series data containing labeled nominal and anomalous periods of behavior.

2.2 Programming frameworks

With programming frameworks that generate telemetry-like data, custom data sets can be built and adapted to certain needed characteristics. This makes frameworks very interesting for model training because the data sets can be adjusted to the specific problem domain. For benchmarking or comparing models with other competitors, the programming parameters used to generate the data sets can be shared. In this section, we will investigate frameworks that use classic solutions, i.e., deterministic approaches, to generate the data sets as well as solutions employing probabilistic approaches. An important remark is that none of the investigated solutions are capable to generate labeled data sets. Therefore, the focus lies on the generation of synthetic time series data resembling satellite telemetry.

There are many frameworks capable of generating synthetic time series data. For example, the open source library *TimeSynth* [25] can generate synthetic time series for model testing. The library can produce both regularly and irregularly sampled data. Its generic architecture allows the generation of a plethora of different signals, e.g., harmonic functions, pseudo-periodic signals, and many more.

The Python library *tsBNgen* [26] generates time series data sets based on an arbitrary Bayesian network structure. It handles discrete nodes by using a multinomial distribution, continuous nodes by using a Gaussian distribution, and can form a hybrid network using a mixture of discrete and continuous nodes.

In [27] Zhang et. al. propose a novel data-driven approach to synthetic data set generation for smart-grids. Using a real data set as input, its conditional probability distribution is learned using deep Generative Adversarial Networks (GANs). The generated synthetic samples are based on the learned distribution.

The paper [28] proposes a new probabilistic forecast model for multivariate time series also based on Conditional GANs.

This multivariate approach makes it interesting for generating more complex data sets even if the focus of the paper is on forecasting data and not on generating data sets.

The authors of the paper [29] focus on one dimensional times series and explore a "few shot" approach. With a "few shot" approach, a model can be trained with only little data, e.g., only a few labeled samples per class. To achieve that, they are using two GANs simultaneously to model fake time series examples. The ability to learn from little data makes this an interesting feature, also for the satellite domain. In that domain, since the usage of operational data as training data is very common, only short snippets of nominal telemetry data are often available.

In [30], two reconstruction methods are proposed for synthetic time series generation, named *Rank-wise* and *Step-wise methods*. For their use case to simulate wind speed, the authors demonstrate the potential of the developed models over other synthetic time series generation methods such as the Markov chain method or an autoregressive method.

SynSys [31] generates synthetic time series data that is composed of nested sequences using hidden Markov models and regression models. Initially, the models are trained on real data sets. The author's goal is to create realistic synthetic smart home sensor data that reflects human behavior.

Since large language models like *ChatGPT* [32] show astonishing results in numerous domains, the authors of [33] propose a foundation model for time series data, named *TimeGPT*. With that model, the generation, i.e., prediction of time series data is possible. However, the model with its framework is still in a early development stage and not open for a broad public, therefore, we could not test this promising new approach.

In the next section, we present our library for synthetic satellite telemetry data, that is capable of producing various telemetry like time series data *including* a wide variety of labeled anomalies.

3 Library for synthetic satellite telemetry data

The Synthetic Telemetry Data library can create satellite-like telemetry data including labeled anomalies. It provides various generators for synthetic telemetry data and different kinds of anomalies. Both the generated telemetry data resemble real satellite telemetry data, and the generated anomalies resemble typical anomalies that can usually be found in real satellite telemetry. These generated data provide robust training data sets for ML tools, especially in the space domain, where the lack of labeled data or even nominal data, that can be used for training tasks, is evident. The data, i.e., the signals and anomalies, are generated by deterministic processes and can be re-produced by re-using the same function-parameters.

This makes the library suitable for research and benchmarking of ML models where reproducible results are crucial.

Since Python became the de facto standard in data science, the synthetic telemetry library is implemented as a modular Python library and can be integrated seamlessly into Jupyter notebooks or any other Python-based data science tool. In general, the development follows the KISS-Principle, keep-it-short-and-simple, which results in a code basis that is both easy to maintain and to extend. With its simple and intuitive usage, the library is applicable even by inexperienced programmers and fits well into the whole Python ecosystem. With respect to a stable support, reliability, and security, the library only depends on few third-party Python libraries that are widely-used and well maintained such as Pandas, Numpy, or Scipy.

In the following parts, key design decisions explaining the architecture of the library are presented.

3.1 Architecture

For us, a signal is a one-dimensional sequence of plain data points. Telemetry, on the other hand, is a signal with added time information for each data point, which makes it a two dimensional structure; a time series. Therefore, the library is separated into two main modules, see Fig. 1: the Synthetic Signal module and the Synthetic Telemetry module.

3.1.1 Synthetic signals

The Synthetic Signal module can generate signals of various types. Currently, there are thirteen predefined signal types available. Moreover, the module provides the possibility to build signals from chunks of real satellite telemetry data, from combinations of arbitrary signals, or from

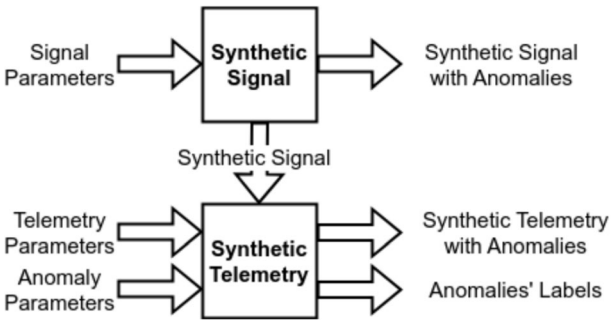


Fig. 1 Synthetic satellite telemetry library's architecture

(mathematical) functions. From our experience as a mission control center, these types already cover a very large range of typical satellite telemetry signals.

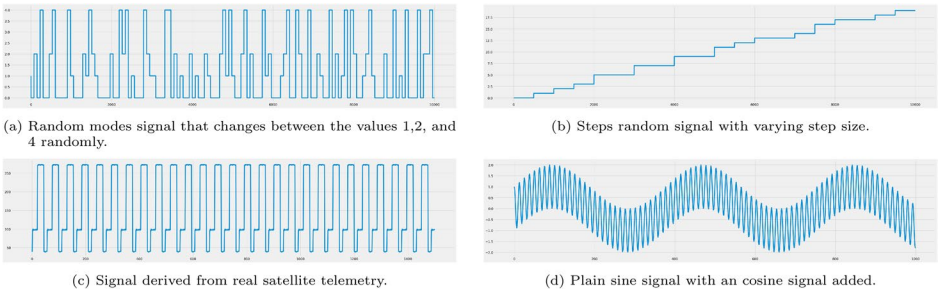
The implementation of the signal types uses the concept of Python Generator functions that behaves similar to an iterator. Therefore, generators are very memory efficient since they generate only as much data as needed or demanded. With this efficient concept, signal generators can produce continuous streams of data that can be used to simulate real-time telemetry reception. Moreover, with generators it is possible to change the configuration of the produced data during run-time. This means that the signal parameters can be changed, e.g., the frequency of a signal or the signal amplitude. That way, anomalies can be injected during the streaming process and provide an option to test and benchmark real-time systems.

The main library's features for the users are:

Predefined signals

The predefined signals form the basis for a fast and easy creation of custom data sets. Furthermore, these types can be used to create more complex signals by combining them to new signals, e.g., Fig. 2d shows an addition of a Sine signal generator and a Cosine signal generator. The implementation uses a generic Python generator that derives arbitrary signal data from a given input function definition. With that generator and predefined function definitions, we can provide some basic signal types to the users that are common in satellite telemetry data or that can be used as basis for more complex signal types. All thirteen predefined signal types are fully customizable by changing the function parameters like frequency, amplitude, etc.:

- *Flat*: just a straight line, i.e., a constant function.
- *Sine*: a plain Sine curve.
- *Cosine*: a plain Cosine curve.
- *Square*: a box-shaped signal with identical boxes dimensions.
- *Triangle*: a saw tooth signal, triangle shaped.
- *Step*: a flat signal with one step after a defined number of points.



(a) Random modes signal that changes between the values 1,2, and 4 randomly.

(b) Steps random signal with varying step size.

(c) Signal derived from real satellite telemetry.

(d) Plain sine signal with an cosine signal added.

Fig. 2 Synthetically generated signals with the number of points on the x-axis and the amplitude value on the *y*-axis

- *Steps*: a raising step function with steps of equal size.
- *Steps Random*: a raising step function with steps of random size (see Fig. 2b).
- *Modes*: a signal that changes between different levels, resembles different operation modes of satellites.
- *Modes Random*: a signal that changes between different levels randomly (see Fig. 2a).
- *Ramp*: a continuous increasing or decreasing signal, like the power level of a loading battery.

Signals from real telemetry snippets

In satellite operation, most of the telemetry data does not contain any anomaly because anomalies occur very rarely on satellites. Therefore, the library provides the option to generate signals from real satellite telemetry snippets as depicted in Fig. 2c. These snippets are repeated indefinitely. However, the generators built from snippets act and can be used exactly like the predefined synthetic signal generators which means all available anomalies are also applicable to these kind of generators. Using this method, the signal parameters like the frequency or the amplitude of a signal can be changed. With the option to take nominal satellite telemetry data and inject artificial anomalies (at well defined time points), we can generate very realistic training data with a representative nominal/anomalous rate for ML tasks.

Combinations of signals

Several signal generators of any kind can be combined by mathematical operator, e.g., plus, minus, or modulo, to a new signal generator. This type of signals resembles the principle of frequency modulation where a carrier signal wave is modified by the transmitted information, a common technique in telecommunications engineering. Thus, it allows the users to build complex signal combinations like a Cosine signal on top of a Sine signal, see Fig. 2d.

Signals from functions

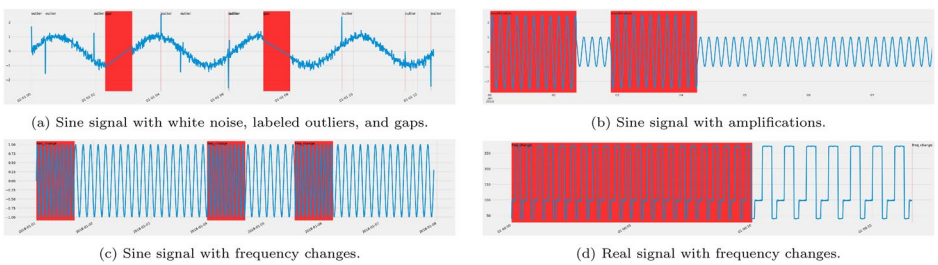
For users who need special signals which are not included into our predefined signal set, the library provides the possibility to generate signals from (mathematical) functions, i.e., Lambda expressions. That way, the user has the freedom to build very specific custom signal types which can be used like the predefined ones.

Injected anomalies

To us, an anomaly means that the signal data is different compared to the rest of the signal, in other words: the data is different from what is expected. If you are working with data streams, then you can already inject anomalies during the signal generation process. These anomalies are realized as generators, too, and can be combined in any possible way. Therefore, you can easily modify your Python program to use signals with any anomaly by just exchanging a signal generator with an anomaly generator. The anomalies will be added randomly to the signal as long as the generator is active and generates data. However, that way, no labels for the anomalies will be generated.

The library offers eight different anomaly types:

- *Outliers*: adds data points that differ significantly from other data points to the signal, visually a peak or a drop in the data (see Fig. 3a). In real satellite telemetry, this is a very common anomaly. Possible causes are transmission errors, interferences, or even hardware defects on board the satellite.



(a) Sine signal with white noise, labeled outliers, and gaps.

(b) Sine signal with amplifications.

(c) Sine signal with frequency changes.

(d) Real signal with frequency changes.

Fig. 3 Synthetically generated telemetry with injected anomalies and labels. On the *x* axis the time information is displayed, on the *y* axis the amplitude

- *Gaps*: deletes data points from the signal flow to introduce gaps, see Fig. 3a. In real satellite operations, this is a common anomaly caused by, e.g., transmission errors.
- *White-Noise*: adds random samples from a normal (Gaussian) distribution to the signal (Fig. 3a). The white noise can be added to the complete signal duration or limited to only certain sections of the signal. Satellite telemetry can be affected by solar flares or other natural phenomena such as atmospheric noise that can cause this kind of anomaly.
- *Trends*: causes a continuous increase or decrease of the signal. On satellites, this anomaly can indicate a degradation of components like the battery cells.
- *Signal-shifts*: shifts the frequency of the signal by a certain factor. This can also happen in satellite telemetry, caused by, e.g., the Doppler effect [34].
- *Resolution changes*: increases or decreases the resolution of the signal. They appear in real data when the satellite has contact with the ground station – the direct up- and down-link to the ground station enables larger data rates because the generated data does not have to be stored on the limited on-board storage device of the satellite.
- *Frequency changes*: changes the frequency of the signal, see Fig. 3c and d. This is also a phenomenon that is induced by ground station contacts.
- *Signal amplifications*: increases or decreases the amplitude of the signal as seen in Fig. 3b. This can be caused by overlay effects or interferences of different signals.

Technically speaking, the plots in Fig. 3 show telemetry data with anomalies instead of plain signals. However, the only difference to plain signal plots, as shown in Fig. 2, is the depicted time information on the x axis instead of the number of points.

For some anomalies, e.g., white noise or trends, the duration can be defined. If the duration lasts the whole signal time span, the anomaly becomes the nominal behavior of the signal, e.g., like the white noise in Fig. 3a.

Some anomaly types can be seen as nominal behavior too, e.g., signal shifts, resolution, and frequency changes. These anomalies can be caused by the routine satellite operations like maneuvers or ground contacts procedures. Based on that, satellite operators would see them as nominal data, not as anomalies. However, we stick to our strict anomaly definition (see Sect. 3.1.1) and interpret every divergent behavior as anomalous behavior. In the end, the user can decide: if a divergent behavior is included into the training data set, then the trained algorithm will learn this divergent behavior as nominal behavior. If the divergent behavior is only put into the test and validation data sets, then the divergent behavior will be detected as anomalous behavior.

3.1.2 Synthetic telemetry

The library module Synthetic Telemetry adds the time information to the generated signal. Furthermore, it contains functions for visualizing, editing, and exporting time series data sets. But its main purpose is to add anomalies with labels to the telemetry data. This means that, in contrast to the Synthetic Signal module, labels are generated after the anomalies are injected. Labels include the kind of the anomaly, the start and end time stamp, and the parameters necessary for the recreation of the injected anomalies. All these data can be exported into a file and stored within a data set to guarantee the reproducibility of the data. The module provides exactly the same types of anomalies as the Synthetic Signal module (see Sect. 3.1.1). However, since the injection of the anomalies happens after the signal generation and with the added time information, which implies that we now handle finite data with a well defined start- and end-date, the user can define the properties of the anomalies with respect to the whole data set more precisely, e.g., the likelihood of an anomalous event within the data set. In Fig. 3, some example signals with their anomalies and labels are plotted using the whole functionality of the Synthetic Telemetry module.

3.2 Synthetic telemetry data set generation

The generation of synthetic telemetry data sets can be easily automated via Python scripts. For example, for anomaly detection in satellite telemetry data, a data set typically consists of training data (without anomalies) and test and validation data sets which include anomalies. The size of these data sets can be huge (with several gigabytes of data) and should include different kinds of signals combined with different kinds of anomalies. For our data sets, we use the following workflow, see Fig. 4.

First, a signal has to be generated, i.e., the plain data points without any time information with the help of the Synthetic Signal module. After that, the time information has to be added to the signal by the Synthetic Telemetry module and the desired anomalies have to be injected. Since the generation of each single telemetry parameter is independent, we use threads to generate more parameters in parallel (be careful with your computer memory (RAM)). Finally, the script stores a file for each telemetry parameter data with its configuration, which finalizes the data set generation.

4 Use cases

Our original motivation for developing the synthetic telemetry data library was to create a flexible data set generator for ML tools training data. However, after finishing the

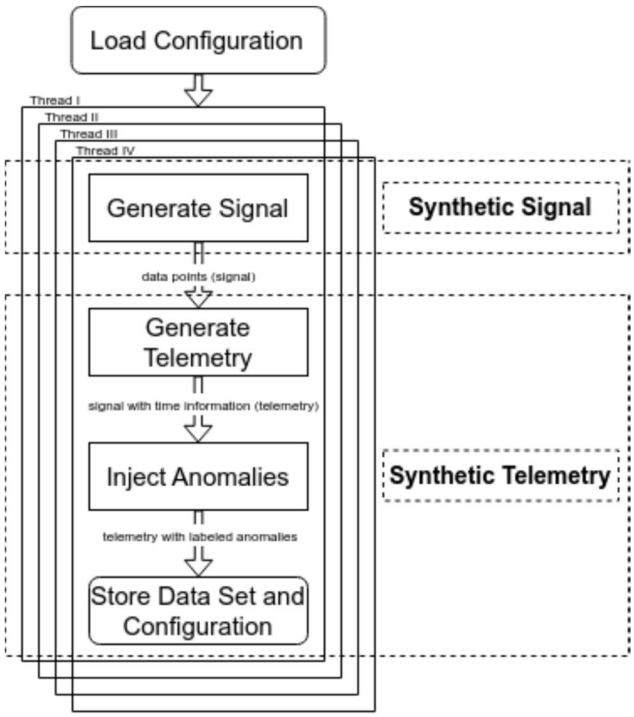


Fig. 4 Data set generation workflow

implementation of the library, we have identified several other fields of application that we present in this section.

4.1 Software development

Software development usually cycles over many phases. While there are many use cases to explore in the various phases, e.g., using synthetic data for more realistic mock-ups during the design phase or to get an understanding for the input when engineering the requirements, we want to describe two use cases we see as the most promising at the German Space Operations Center (GSOC). First, we describe the benefits of the synthetic data for testing and continuous integration (CI) of the software product. The second main use case is the development of prototypes which, especially for data science related applications based on ML methods, require reliable and labeled data as input for a proper evaluation.

4.1.1 Continuous integration and testing

Typical CI pipelines nowadays automate many steps such as checking the code style, building the product, or generating documentation and release notes. At GSOC, we often use these pipelines implemented in GitLab, our version control system, to ensure and improve the quality of our operational software such as our mission planning system [35, 36]. An example of a successful CI pipeline is depicted in Fig. 5.

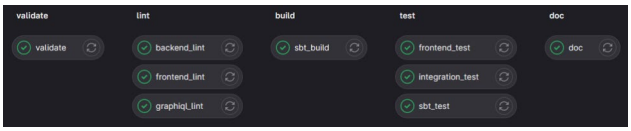


Fig. 5 Example of a successful CI pipeline at the GSOC including code validation, style checks, building, testing and generation of documentation

When applying CI to software implementing ML methods, the evaluation of the algorithm's performance or the optimization and generation of the ML model are also often an important part of the CI pipeline. Both of these steps almost always require an accurately labeled data set to base their results on. While using a synthetic data set for the model generation could lead to overfitting depending on the ML method and its application, it brings a great benefit to the performance evaluation of the algorithm in the CI pipeline.

The labeled synthetic data can be used to assess both the computational performance of an algorithm as well as its performance in terms of accuracy. To evaluate the computational performance, the data set can be constructed to contain samples with a realistic frequency over a typical timespan which the algorithm should be able to handle. Evaluating the algorithm on the same hardware in each triggered pipeline and recording its execution time allows to track whether changes or updates to the software or its imported libraries slow down the algorithm. The accuracy of an algorithm can be evaluated using a labeled data set fitting the algorithm's application. For, e.g., measuring the performance of ATHMoS, our telemetry anomaly detection ML tool, synthetic data containing various labeled anomalies can be used to compute evaluation metrics such as the F1 score or the receiver operating characteristic (ROC). A deterioration of the algorithm or its configuration can be measured in each pipeline execution and ensures the software yields good results with respect to the synthetic baseline. Should new behaviors occur or requirements change, the synthetic data set can easily and reliably be regenerated to reflect the changes.

4.1.2 Prototyping

When developing software prototypes for ML applications, labeled data is vital as it provides a basis for the evaluation of new approaches. In the domain of spacecraft operations, a large and reliable data set is often not available or of confidential nature. The synthetic labeled data set can solve these issues.

One example of the data set's benefits is the development of a real-time version of ATHMoS, the anomaly detection system at the GSOC. ATHMoS was originally designed as a batch processing system for low-orbit satellites. In this use

case, telemetry data is only available during ground contact and all data is downloaded and analyzed afterwards. However, GSOC also operates geostationary satellites. For those satellites, an almost permanent connection to the satellite is available and we aim to optimize ATHMoS to run as a real-time system as well. Therefore, we implemented a CI pipeline that is using the synthetic data to measure how fast the system reacts to new behavior and which data rate can be handled realistically. After each run, this pipeline generates a web page containing a summary of the results of the measurements. From the results, developers can quickly get feedback about their new optimizations. Figure 6 depicts the result page with the top plot showing the training and test data with the computed anomaly scores for a simple sine wave shaped telemetry. For a better comparison, the training and test data are plotted as line plots underneath.

Another example is the classification of various behaviors, both anomalous as well as nominal. As the synthetic data generator also provides labels describing the type of the introduced behaviors, methods correctly classifying the different behaviors can be researched and prototyped using the synthetic data set.

4.2 Anomaly detection

Synthetic data also has many applications for anomaly detection. While using labeled, real-world data would be optimal, it is also often not feasible to spend many engineers' hours on labeling to get such data. Instead, one can rely on synthetic data which can include artificial anomalies and their labels to create fully labeled data sets.

Most importantly, one can use labeled synthetic data to test different AI methods, e.g., while developing your own algorithm, and also to benchmark different AI models using standardized training and test sets. Finally, one might run into situations where there is not sufficient training data available, e.g., because the system is very new. Using the synthetic data generator we can use expert knowledge to create artificial time series which are sufficiently close to the real data. This synthetic data can be used to train the initial AI model for anomaly detection.

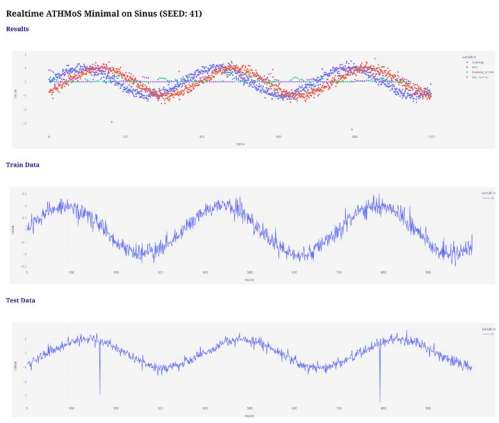


Fig. 6 Data science CI pipeline results of real-time ATHMoS

4.2.1 Method testing

When developing a new method for anomaly detection, one has to check that a sufficient amount of anomalous behavior has been detected, in particular that both the rate of false negative (missed) detections and the rate of false positive detections is sufficiently small. Since AI developers often do not have labeled data available – just like our satellite data at GSOC – the only way to verify the results of different AI models is to verify detections as anomalies, e.g., by discussing detections with the our satellite engineers. In contrast, without labeled data, it is hard to make sure the models haven't missed any significant anomalies.

Using fully labeled, synthetic data enables developers to automatically compute both the false negative and false positive rate, and tweak their models to lower those rates.

4.2.2 Benchmarking

Similar to the use case in the previous section, one can use the fully labeled, synthetic data set to benchmark different ML algorithms. A complete set of labels allows the developer to determine the number of true positives (detected anomalies) and the number of true negatives (not detected nominal behavior) as well as the number of false negative (not detected anomalies) and false positive detections (detected nominal behaviour). Using those numbers, one can compute different metrics to evaluate the precision of the model, such as the $F1$ score or the AUROC score.

To provide an example, we benchmarked the algorithm used in ATHMoS, the outlier probability via intrinsic dimension (OPVID) [1], against the local outlier probability algorithm (LoOP) [37]. Four different synthetic signals derived from real telemetry signals along with injected anomalies were used for this benchmark, see Sect. 3.1.2. In Fig. 7, we can see that ATHMoS performs better for this type of data and injected anomalies with respect to the AUROC metric. In addition to the model precision, a standardized data set can also be used to evaluate the performance of the model. This includes the time needed for training the model and testing/inference, and the data capacity, in particular for storing the model.

4.2.3 Training of AI methods for anomaly detection

A final use case at GSOC is using synthetic data to train an AI model where we currently have insufficient data. When a new satellite launches, the engineers might know how

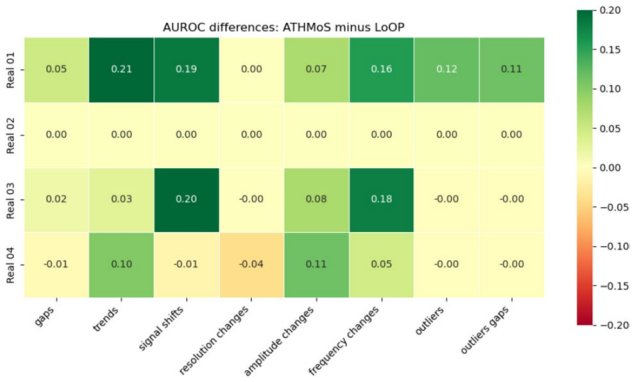


Fig. 7 AUROC metric of ATHMoS benchmarks (OPVID vs. LoOP)

a certain telemetry parameter should behave and want to supervise this parameter as soon as possible. Using the synthetic data generator, one can generate a time series which mimics nominal behavior for this parameter. Then, one can train an AI model for anomaly detection to start detecting anomalies beginning shortly after the Launch and Early Orbit Phase (LEOP) instead of waiting a year or more for sensible data. To improve the model going from synthetic to real data, one can include frequent re-trainings using as much real data as are available.

5 Proofs of concept

After introducing the main concepts and showcasing the library's capabilities through relevant use cases, this section now demonstrates their practical application with two compelling examples. First, we utilize the library to synthesize some representative satellite parameters, to allow us to accurately mimic their characteristics and then compare the resulting simulated data against actual measurements from real satellite parameters. Second, we leverage the library to mock a specific parameter, train our in-house anomaly detection framework ATHMoS [38, 39], and then employ the trained model to detect anomalous behavior in real data from that parameter.

5.1 Mimic real satellite telemetry data

Our library allows us to generate realistic satellite telemetry data by simulating various parameters. This capability is particularly useful for testing and evaluation purposes (see Sect. 4), where access to real satellite data may be limited or restricted. To demonstrate the effectiveness of our library, we generate synthetic telemetry data for a set of key parameters and compare it against actual measurements from a real satellite. Notably, since our real satellite data are confidential, we have anonymized it for presentation purposes; in contrast, no such treatment was necessary for the synthesized data.

In detail, the four key parameters are GPS_X, which monitors the GPS position of the satellite with a precision of centimeters. Additionally, TEMP provides critical thermal information for the satellite's systems, measured in degrees Celsius. The satellite's electric power/distribution subsystem (EPS) is also monitored through two key parameters: EPS_MAIN, which tracks the main electric power bus voltage, and EPS_BAT, which monitors the battery electric current.

Figure 8 contains the plots of data of four real parameters in the left column. In the right column, four synthetic parameters are plotted. They are generated with our library and mimic the behavior of the real parameters. The last listed real parameter, EPS_BAT, contains some outliers which have also been mocked in the synthetic parameter. At first glance, the similarity between the real and the synthetic parameters is striking. To provide a more precise assessment, we present statistical comparisons between the real and synthetic parameters in Table 1. The primary discrepancy between the real and synthetic data for Parameter GPS_X is revealed at the 50th percentile. The discrepancy at the 50th percentile of Parameter GPS_X is caused by an imperfect fit to the underlying sine wave, and can be addressed by refining the parameter's hyper-parameters. Despite this discrepancy, the synthetic data remain suitable for training purposes, as we demonstrate in the following paragraph.

5.2 AI training with synthetic telemetry data

Now, with a synthetic telemetry data that accurately mimics the characteristics of a real satellite parameter, we can leverage this to train our in-house anomaly detection framework ATHMoS. By training ATHMoS on this synthesized data, we equip it to identify anomalous behavior in the actual parameters.

ATHMoS [40] typically uses one year of past data to generate a trained model. We accomplish this by splitting our input data into short intervals, each spanning 1.5 h. The input data, i.e., the training set, consists of one year of synthetic data of Parameter GPS_X without any anomalies. As a next step, we compute feature vectors containing descriptive statistics for each interval. Based on each feature vector's k-nearest neighborhood, a probabilistic outlier score using the intrinsic dimension is calculated [41]. The k-nearest neighborhood, the intrinsic dimension, and the probabilistic intrinsic dimension outlier score make up our trained model.

New telemetry data, here the data from the real parameter, are tested against the computed model by applying the OPVID [1] algorithm, resulting in scores between 0.0 and 1.0 with scores greater or equal to 0.9 typically categorized as an anomaly. This score roughly describes how similar the descriptive feature vector computed for the new

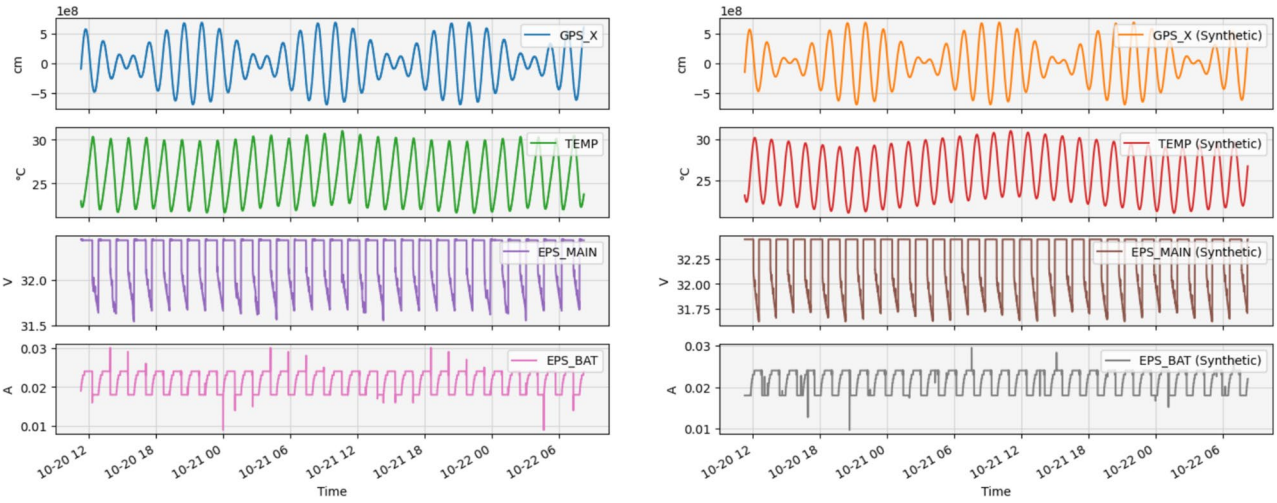


Fig. 8 Comparison of real satellite telemetry data (on the left) with synthetic generated data (on the right)

Table 1 Statistical descriptions of real and synthetic telemetry parameters

	GPS_X		TEMP		EPS_MAIN		EPS_BAT	
	Real	Synth.	Real	Synth.	Real	Synth.	Real	Synth.
Count	20,955.00	20,955.00	20,955.00	20,955.00	20,955.00	20,955.00	20,955.00	20,955.00
Mean	1,842,159.45	185,204.14	25.91	25.81	32.24	32.26	0.02	0.02
Std	337,831,022.45	334,272,129.07	2.72	2.89	0.28	0.28	0.00	0.00
Min	− 692,044,768.00	− 692,842,010.05	21.58	20.95	31.54	31.62	0.01	0.01
25%	− 233,764,666.00	− 222,814,226.24	23.36	23.05	31.97	31.98	0.02	0.02
50%	3,420,925.00	11,184,317.54	25.81	25.77	32.44	32.45	0.02	0.02
75%	254,040,262.00	249,563,670.86	28.38	28.58	32.44	32.45	0.02	0.02
Max	685,000,557.00	684,347,800.10	31.03	31.03	32.45	32.45	0.03	0.03

telemetry data is to the ones used in the trained model. For this set, we took about one week of the real satellite data and used our library to inject two kinds of anomalies, eight outliers and one signal amplification. With the knowledge of the characteristics of the anomalies, we are able to measure the quality of our data as AI training set.

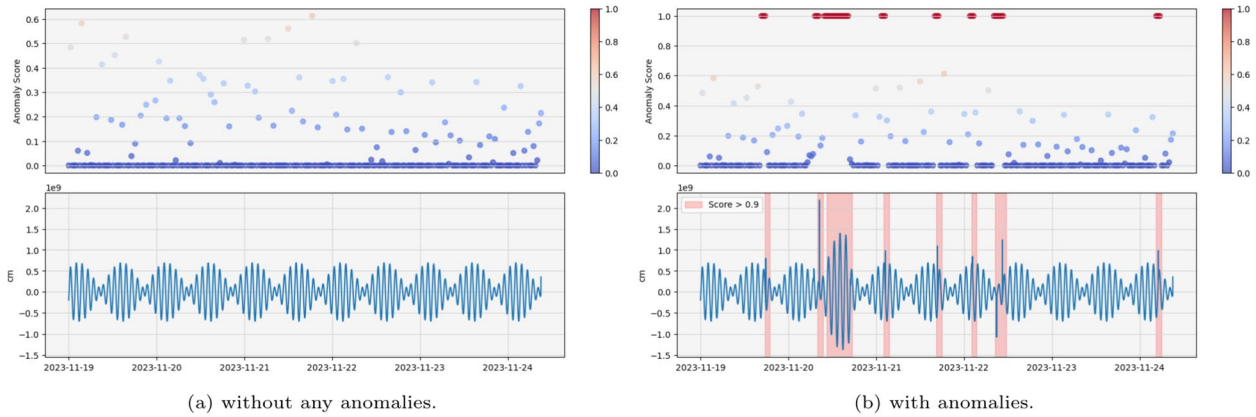
To validate that our synthetic data accurately replicates real satellite parameter characteristics, we test against the trained ATHMoS model to nominal telemetry data from an actual satellite (i.e., data without any anomalies). As shown in Fig. 9a, the resulting anomaly scores are consistently low and confirm the absence of anomalous behavior.

To further demonstrate the effectiveness of our approach, we intentionally introduce anomalies into the actual telemetry data and re-run the trained ATHMoS model on this modified data set. As evident in Fig. 9b, all injected anomalies are accurately detected by the model, resulting in high anomaly scores.

6 Future work

In this section, we want to give a short outlook on planned future extensions of our library.

The most obvious extension is to implement the possibility to represent dependencies and correlations between telemetry parameters. As the individual system components of a satellite depend on each other, the behavior of one parameter influences the behavior of other parameters. For example, the state of the battery depends on the state of the solar panels: If sunlight shines on the solar panels, they produce power and charge the battery. The telemetry data of both components reflect this dependency. More complex dependencies are the result of maneuvers, which normally involve plenty of satellite components and affect their states. Synthetically generating dependencies between telemetry parameters would allow us to train



(a) without any anomalies. (b) with anomalies.

Fig. 9 Anomaly score and telemetry data of the real satellite Parameter GPS_X

models on multivariate training sets and open the way for the detection of more complex anomalies.

Including periodic behavior in synthetic telemetry data is another candidate for an extension. As we know, low-earth orbit (LEO) satellites fly in orbit about 90 min around the earth. This cycle is apparent in many telemetry parameters of a LEO satellite. Another example of a periodic behavior is the eclipse cycle, when the satellite enters the shadow of the earth. In this phase, certain configuration are triggered to prepare the satellite, e.g., for safe energy as the solar panels will not produce any power during the eclipse. A configuration of periodic behavior would be a great asset to make the synthetic telemetry data more realistic.

Another idea is to implement data generators on the basis of generative adversarial networks (GANs). This deep learning technique is used in many disciplines for the generation of synthetic data. Since it learns from real telemetry data, it could produce very realistic data and, furthermore, when trained on multivariate data, even dependencies between single telemetry parameters. However, it is challenging to obtain a reliable nominal training data set and to get labels for the anomalies injected by the GAN.

Lastly, the library can be extended to inject more realistic anomalies into the generated data. For now, we only support basic anomaly types. Therefore, more complex anomalies or even patterns of real anomalies would be a benefit for our library.

7 Summary and conclusion

In this paper, we present a Python library for synthetic satellite telemetry data generation. This novel library can generate plenty of different types of telemetry signals and can inject various types of anomalies into the generated data sets including their labels. In the section on related work, we review public available telemetry data sets (with labeled anomalies) and other frameworks for telemetry data generation. Our library is original because it combines the advantages of both worlds: on the one hand, the possibility to generate data sets with labeled anomalies *and* on the other hand the flexibility of generating an arbitrary amount of customized satellite telemetry signals for ML tasks.

Its modular structure, and simple and intuitive usage ensure that our library fits well into the whole Python ecosystem and enables data scientists to easily integrate our library into their tools. Moreover, we demonstrate its applicability with several use cases in the space domain. We describe the integration into our in-house software development process, where we use the library for the development of new prototypes, for continuous integration and for testing of already deployed software. Lastly, we show the usability of the generated data in ML development, e.g., for training data sets and for benchmarking different ML models and prove our concepts within two scenarios: the mocking of satellite parameters and the AI training with synthetic data.

Acknowledgements The authors are thankful to Steffen Zimmermann, Dr. Martin Wickler, Mila Stillman, and all members of the DLR MBT-Team for their valuable support and fruitful discussions.

Author contributions C.S. wrote the main manuscript text, L.S. and K.H. wrote most of Section 4. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. O'Meara, C., Schlag, L., Faltenbacher, L., Wickler, M.: ATHMoS: automated telemetry health monitoring system at GSOC using outlier detection and supervised machine learning. In: Proceedings of the 14th International Conference on Space Operations (SpaceOps 2016), 5 (2016)
2. Schefels, C., Schlag, L., Del Moro, A., Helmsauer, K., Lesch, T., Göttfert, T.: Bringing a machine-learning based novelty detection software tool from research to production. In: Proceedings of the 17th International Conference on Space Operations (SpaceOps 2023), 5 (2023)
3. Märtens, M., Izzo, D.: Mars express power challenge dataset (2016). https://doi.org/10.5281/zenodo.6327379
4. Mars express–investigating the red planet. https://www.esa.int/Science_Exploration/Space_Science/Mars_Express/. Accessed 21 Aug 2023
5. Mars express power challenge. https://kelvins.esa.int/mars-express-power-challenge/. Accessed 21 Aug 2023
6. Andrzejewski, J., Ruszczak, B., Nalepa, J., Lakey, D., Collins, P., Kolmas, A., Bartesaghi, M., Martínez-Heras, J.: ESA anomaly dataset. https://zenodo.org/records/12528696. Accessed 21 Nov 2024
7. Kotowski, K., Haskamp, C., Andrzejewski, J., Ruszczak, B., Nalepa, J., Lakey, D., Collins, P., Kolmas, A., Bartesaghi, M., Martinez-Heras, J., De Canio, G.: European space agency benchmark for anomaly detection in satellite telemetry (2024). arXiv:2406.17826
8. ESA anomaly detection benchmark. https://github.com/kplabs-pl/ESA-ADB. Accessed 21 Nov 2024
9. De Canio, G., Eggleston, J., Fauste, J., Palowski, A.M., Spada, M.: Development of an actionable AI roadmap for automating mission operations. In: Proceedings of the 17th International Conference on Space Operations (SpaceOps 2023), 5 (2023)
10. Telemanom—anomaly detection in time series data using LSTMS and automatic thresholding. https://github.com/khundman/telemanom. Accessed 21 Aug 2023
11. Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Soderstrom, T.: Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM (2018). https://doi.org/10.1145/3219819.3219845
12. Soil Moisture Active Passive (SMAP). https://smap.jpl.nasa.gov/. Accessed 21 Aug 2023
13. Mars Curiosity Rover (MCR). https://mars.nasa.gov/msl/. Accessed 21 Aug 2023
14. Draper, D.: Challenger USA space shuttle o-ring. UCI Machine Learning Repository (1993). https://doi.org/10.24432/C5PW2T
15. UC Irvine Machine Learning Repository. https://archive.ics.uci.edu/. Accessed 21 Aug 2023
16. Shuttle Landing Control. UCI machine learning repository (1988). https://doi.org/10.24432/C57S34
17. Statlog (Shuttle). UCI machine learning repository. https://doi.org/10.24432/C5WS31
18. Outlier Detection Datasets (ODDs). https://odds.cs.stonybrook.edu/. Accessed 21 Aug 2023
19. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed 21 Aug 2023
20. KDD '99: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA (1999). Association for Computing Machinery
21. UCR time series classification archive. https://www.cs.ucr.edu/%7Eeamonn/time_series_data_2018/. Accessed 21 Aug 2023
22. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.-C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The UCR time series archive (2019). https://doi.org/10.48550/arXiv.1810.07758
23. Climate data online: dataset discovery. https://www.ncdc.noaa.gov/cdo-web/datasets/. Accessed 21 Aug 2023
24. The Numenta Anomaly Benchmark (NAB) build status. https://github.com/numenta/NAB/. Accessed 21 Aug 2023
25. Maat, J.R., Malali, A., Protopapas, P.: Timesynth–multipurpose library for synthetic time series. https://github.com/TimeSynth/TimeSynth/ (2017). Accessed 21 Aug 2023
26. Tadayon, M., Pottie, G.: tsBNgen: a python library to generate time series data from an arbitrary dynamic Bayesian network structure. https://github.com/manitadayon/tsBNgen. Accessed 21 Aug 2023
27. Zhang, C., Kuppannagari, S.R., Kannan, R., Prasanna, V.K.: Generative adversarial network for synthetic time series data generation in smart grids. In: 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1–6 (2018). https://doi.org/10.1109/SmartGridComm.2018.8587464
28. Koochali, Alireza, Dengel, Andreas, Ahmed, Sheraz: If you like it, GAN it-probabilistic multivariate times series forecast with GAN. Eng. Proc. **5**, 40 (2021). https://doi.org/10.3390/engproc2021005040
29. Smith, K.E., Smith, A.O.: Conditional GAN for timeseries generation. arXiv preprint (2020). https://doi.org/10.48550/arXiv.2006.16477
30. Bokde, Neeraj Dhanraj, Feijóo, Andrés, Al-Ansari, Nadhir, Yaseen, Zaher Mundher: A comparison between reconstruction methods for generation of synthetic time series applied to wind speed simulation. IEEE Access **7**, 135386–135398 (2019). https://doi.org/10.1109/ACCESS.2019.2941826
31. Dahmen, J., Cook, D.: SynSys: a synthetic data generation system for healthcare applications. Sensors (2019). https://doi.org/10.3390/s19051181
32. OpenAI. Blog: introducing ChatGPT. https://openai.com/blog/chatgpt. Accessed 11 Mar 2024
33. Garza, A., Mergenthaler-Canseco, M.: TimeGPT-1. arXiv preprint (2023). https://doi.org/10.48550/arXiv.2310.03589
34. Narayana, S., Muralishankar, R., Venkatesha Prasad, R., Rao, V.S.: Recovering bits from thin air: demodulation of bandpass sampled noisy signals for space IoT. In: Proceedings of the 18th International Conference on Information Processing in Sensor Networks, IPSN '19, pp. 1–12. New York, NY, USA (2019). Association for Computing Machinery. https://doi.org/10.1145/3302506.3310384
35. Wiebigke, A., Krenss, J., Hartung, J., Wiesner, S., Nibler, R., Fürbacher, A.: PintaOnWeb—the front end of GSOC's next generation mission planning systems. In: Proceedings of the 17th International Conference on Space Operations (SpaceOps 2023), 5 (2023)

36. Langs, A., Oertlin, J., Trifin, F.: Applying continuous integration for operational products in the mission preparation environment. In: Proceedings of the 17th International Conference on Space Operations (SpaceOps 2023), 5 (2023)
37. Kriegel, H.-P., Kröger, P., Schubert, E., Zimek, A.: LoOP: local outlier probabilities. In: Proceeding of the 18th ACM Conference on Information and Knowledge Management—CIKM '09, pp. 1649–1652. ACM Press (2009). https://doi.org/10.1145/1645953.1646195
38. Schlag, L., Dauth, M., Braun, A.: The GSOC satellite telemetry analysis framework. In: Deutscher Luft- und Raumfahrtkongress 2019 (DLRK 2019), vol. 9 (2019)
39. Schlag, L., O'Meara, C., Wickler, M.: Numerical analysis of automated anomaly detection algorithms for satellite telemetry. In: Proceedings of the 15th International Conference on Space Operations (SpaceOps 2018), 5 (2018)
40. Schlag, L., Schefels, C., Helmsauer, K.: Applying machine learning to routine satellite ground segment operations by means of automated anomaly detection. In: Aerospace Europe Conference 2023 (EUCASS-CEAS 2023), vol. 07 (2023)
41. Von Brünken, J., Houle, M.E., Zimek, A.: Intrinsic dimensional outlier detection in high-dimensional data. NII Tech. Rep. **2015**(3), 1–12 (2015)