18th International Conference on Space Operations, Montreal, Canada, 26 - 30 May 2025.

SpaceOps-2025, ID # 397

A Transformer-Based Deep Learning Approach to Anomaly Detection of High-Bandwidth Multivariate Time-Series Satellite Communications

Alexey Yermakov[1]*, Kyongsik Yun[2], Lillian Ratliff[1], J. Nathan Kutz[1]

[1] Electrical & Computer Engineering, University of Washington, Seattle, Washington, United States
[2] Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, United States
* Corresponding Author, alexeyy@uw.edu

Abstract

Monitoring incoming spacecraft data to detect anomalies is essential for the continuation of a successful mission. Spacecraft Operations Engineers have traditionally relied on manual monitoring of satellite data, with limited statistical support for anomaly detection, focusing primarily on threshold-based monitoring of individual data items to identify and resolve any anomalous behavior; however, as the number of satellites launched increases exponentially every year with growing bandwidth per new satellite, the amount of data Operations Engineers need to monitor is increasing at an alarming rate. There is a pressing concern of monitoring all the incoming data without increasing operating costs.

In this paper, we demonstrate that modern deep-learning anomaly detection methods can aid existing Spacecraft Operations Engineers. Previous research has explored the use of machine learning to monitor incoming data and alert Operations Engineers about potential anomalies. We build upon the literature by taking a multivariate time-series dataset (DSN spacecraft monitor data) and feeding it into a transformer-based deep learning algorithm. Whereas previous work uses complete data with a single data type, this dataset proves challenging due to various data types per time-step and missing data. We refine the data preprocessing and architecture of the TranAD algorithm to improve its performance on both the DSN dataset and existing public anomaly detection datasets. We demonstrate that the modified TranAD algorithm, TranAD+, improves performance over MTAD-GAT, OmniAnomaly, and USAD on the DSN dataset. We also tested some public datasets, including SMAP, MSL, SWaT, SMD, and WADI, and demonstrated improved performance on SMAP, SWaT, and SMD. The improved performance promises the ability for Spacecraft Operations Engineers to instantaneously detect anomalies in the high-bandwidth regime of modern satellite communications.

Keywords: Anomaly Detection, Deep Learning, Transformers, Time-series.

Nomenclature

$\mathcal{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(n)}\}$	
$\mathcal{X}$	Arbitrary time-series
$\mathcal{X}_{test}$	Testing split for the time-series
$\mathcal{X}_{train}$	Testing split for the time-series
$\mathbf{x}^{(t)}$	*m*-dimensional vector at time-step *t* for a time-series
$x_i^{(t)}$	The *i*-th element of the *m*-dimensional vector at time-step *t* for a time-series
$\mathcal{Y} = \{y^{(1)}, y^{(2)}, ..., y^{(n)}\}$	
$\mathcal{Y}$	The corresponding time-series containing anomaly label information
$y^{(t)}$	Value of either 1 or 0 classifying a time-step as anomalous or not (respectively)
$\mathcal{Z} = \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(n)}\}$	
$\mathcal{Z}$	The corresponding time-series containing anomaly contribution information
$\mathbf{z}^{(t)}$	*m*-dimensional vector at time-step *t* for a time-series
$z_i^{(t)}$	The proportional contribution of parameter *i* to the anomaly at time-step *t* for a time-series

$W_t = \{\mathbf{x}^{(t-\ell+1)}, ..., \mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}\}$
W_t A sliding window of length ℓ at time *t*

$\mathcal{W} = \{W_1, W_2, ..., W_t\}$
$\mathcal{W}$ A sliding window-dataset composed of *t* sliding windows of a time-series dataset

$\min(\mathcal{P})$ The component-wise minimum values of the time-series across time
$\max(\mathcal{P})$ The component-wise maximum values of the time-series across time

$\mathbf{R}$ The set of real numbers
$\mathbf{K}$ The set of categorical values

Acronyms/Abbreviations

Deep Space Network (DSN)
Jet Propulsion Laboratory (JPL)
Long Short-Term Memory (LSTM)
Long Short-Term Memory Nonparametric Dynamic Thresholding (LSTM-NDT)
Multilayer Perceptron (MLP)
Multivariate Time-series Anomaly Detection via Graph Attention Network (MTAD-GAT)
National Aeronautics and Space Administration (NASA)
Peaks Over Threshold (POT)
Recurrent Neural Network (RNN)
Transformer-based Anomaly Detection model (TranAD)
UnSupervised Anomaly Detection on Multivariate Time Series (USAD)
Variational Autoencoder (VAE)
Large Language Model (LLM)

1. Introduction

NASA's Deep Space Network is an array of antennas located around the globe that communicate with spacecraft at least 10,000 miles away from Earth [1]. Among its functions, the DSN is critical for making observations about the Solar System, the universe, and supports a select amount of Earth-orbiting missions [1]. Despite the importance of the DSN, the network is currently oversubscribed, at times reaching more than 40% more demand than the network is capable of supplying [2]. Making matter worse, the current networking capabilities of the DSN are not able to keep up with the increase of missions and increases in data throughput [3]. For example, in 2023 the Psyche mission was launched with laser communication capabilities from deep space in an effort to achieve data rates up to 10 to 100 times faster than current radio systems [4, 5]. Furthermore, the DSN is expected to provide a role in supporting the crewed Artemis missions to the moon, further increasing expected bandwidth usage from the DSN [2].

While there are plans to expand the network capabilities of the DSN with additional satellites being constructed [2], there is still the problem of hiring personnel to work on the Deep Space Network. Due to recent budget cuts, JPL had to lay off 8% of their workforce, causing additional strain on the already burdened DSN [6]. As a result, there is a demand for existing Spacecraft Operations Engineers to be able to parse the increase in incoming data.

Incoming spacecraft data for the DSN are represented as a multivariate time-series dataset, where each parameter can contain numerical, categorical, or missing information. In this paper, we demonstrate that modern advances in deep learning are capable of taking the load off of Operations Engineers in detecting anomalies for spacecraft data. We can take previous time-series data for any spacecraft, train a deep-learning anomaly detection method on the data to obtain an initialized model, and then use that model to flag incoming data as anomalous or not in real-time. We then provide downstream uses of this model by demonstrating how its output can be used in conjunction with an LLM to produce easily digestible reports on anomalous observations.

2. Previous Work

Anomaly detection is a broadly applicable field. It started as part of research for the U.S. Army in 1969 [7]. Since then, the field has expanded to apply anomaly detection to a variety of different domains, from healthcare [8] to cybersecurity [9]. Approaching anomaly detection from the machine learning perspective allows for general models to be used for a variety of downstream tasks. In this paper we deal with multivariate time-series data. One of the first

deep-learning based examples of anomaly detection on multivariate time-series data used LSTMs, a type of recurrent neural network [10]. An LSTM-based method was used to predict future time-steps from a window of the most recent time-steps and output an error value for each incoming time-series data point. They then took the error scores and proposed a nonparametric dynamic thresholding method to label anomalous regions of the time-series data. While they achieved good results, the method falls short in practice today due to having a single model per parameter. This shortcoming increases training time, inference time, and compute resources needed to be able to detect anomalies in real-time.

Omnianomaly improves upon the LSTM-NDT method by replacing LSTMs with RNNs and uses a Variational AutoEncoder to account for stochasticity in time-series predictions [11]. They also use a Peak-Over-Threshold method from Extreme Value Theory to perform automatic threshold selection from the generated anomaly values [12]. This method outperforms previous models in part due to the model being able to take an entire time-series step at once, not needing to create a model per parameter as in LSTM-NDT.

MTAD-GAT takes a different approach, where once again the individual parameters are modeled independently with convolutions. Two graph attention networks are then used to model relationships between features across time [13]. The outputs of the GATs are then passed through an RNN, a VAE, and an MLP to produce an anomaly score. The result is a model that outperforms both OmniAnomaly and LSTM-NDT on the datasets published.

USAD uses a simple MLP for the core of their architecture, but uses a novel training strategy by using an auto-encoder architecture [14]. As a result of their core model being simple, USAD spends several orders of magnitude less time in training than Omnianomaly. This approach benefits industry applications significantly by reducing turnaround time from training to anomaly detection.

Finally, TranAD improves upon USAD by taking the MLP component of the architecture and replacing it with Transformers [15]. By keeping the auto-encoder component, training times remain small, even outperforming USAD. The use of transformers results in TranAD being more parallelized, more powerful, and quicker to train than previous methods, motivating it as the base method for this work.

3. Method

3.1 Problem Formulation

Consider a time-series $\mathcal{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(n)}\}$ where each time step $\mathbf{x}^{(t)}$ is an *m*-dimensional vector $\{x_1^{(t)}, x_2^{(t)}, ..., x_m^{(t)}\}^T$ of either real ($\mathbf{R}$) or categorical ($\mathbf{K}$) parameters. We only consider time-series where each index into the time-series at any step is of the same variable type, but each index can be of a different variable type. So, it is always the case that if $x_i^{(t)} \in \mathbf{R}$, then $x_i^{(t+1)} \in \mathbf{R}$ and any $x_i^{(t)}$ can be an element of $\mathbf{R}$ or $\mathbf{K}$. The goal is to be able to take the time-series $\mathcal{X}$ and produce outputs $\mathcal{Y} = \{y^{(1)}, y^{(2)}, ..., y^{(n)}\}$ and $\mathcal{Z} = \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(n)}\}$ where $y^{(t)} \in \{0, 1\}$ classifies a time-step as anomalous and $\mathbf{z}^{(t)} \in [0, 1]^m$ identifies the proportional contribution of each parameter to the flagging of the anomalous time-step $\left(\sum_{j=1}^{m} z_j^{(t)} = 1\right)$.

3.2 Data Preprocessing

To work with the categorical variables we implement two methods. The first is one-hot encoding each parameter in-place in the time-series. So, if the original number of parameters was *20* and parameter *1* was categorical with *3* possible values and parameter *5* was categorical with *7* possible values, the one-hot encoded time-series would then contain *30* numerical parameters. The second method is assigning a unique integer to each unique categorical value. This keeps the total number of parameters the same, but converts all categorical parameters to numerical parameters.

After converting the time-series to only contain numerical values, we deal with missing information by replacing them with *0*.

Then, we normalize the time-series by scaling the values for each parameter into the range [0,1]:

$$\mathbf{x}^{(i)} \leftarrow \frac{\mathbf{x}^{(i)} - \min(\mathcal{P})}{\max(\mathcal{P}) - \min(\mathcal{P}) + \epsilon}$$

Where $\min(\mathcal{P})$ and $\max(\mathcal{P})$ are the component-wise minimum and maximum values of the time-series across time and $\epsilon > 0$ is a small real number to prevent division by zero.

The dependence of a point $\mathrm{x}^{(m)}$ on previous points is modeled through the use of a sliding window of length ℓ. Let the window at time t be $\mathrm{W}_t = \{\mathbf{x}^{(t-\ell+1)}, ..., \mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}\}$. For any $t < \ell$ we use replication padding of $\mathrm{x}^{(1)}$ until a window length of ℓ is achieved. So, for $\ell = 5$ we have $\mathrm{W}_3 = \{\mathbf{x}^{(1)}, \mathbf{x}^{(1)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}\}$. Creating windows over the entire time-series dataset X produces the sliding window dataset $\mathrm{W} = \{\mathrm{W}_1, W_2, ..., W_t\}$.

To train the model and to evaluate its performance, we split a time-series dataset $\mathcal{X}$ into two disjoint parts: $\mathcal{X}_{train}$ and $\mathcal{X}_{test}$, where $\mathcal{X} = \mathcal{X}_{train} \cup \mathcal{X}_{test}$. The dataset can be split in two ways. First, it can be split along the time dimensions, where X_{train} has the earlier time-steps and $\mathcal{X}_{test}$ has the later time-steps. Second, it can be split along entire tracks, so a single track is either a training track or a testing track. We construct the windows from $\mathcal{X}_{train}$ and $\mathcal{X}_{test}$ to obtain $\mathcal{W}_{train}$ and $\mathcal{W}_{test}$.

3.3 Model

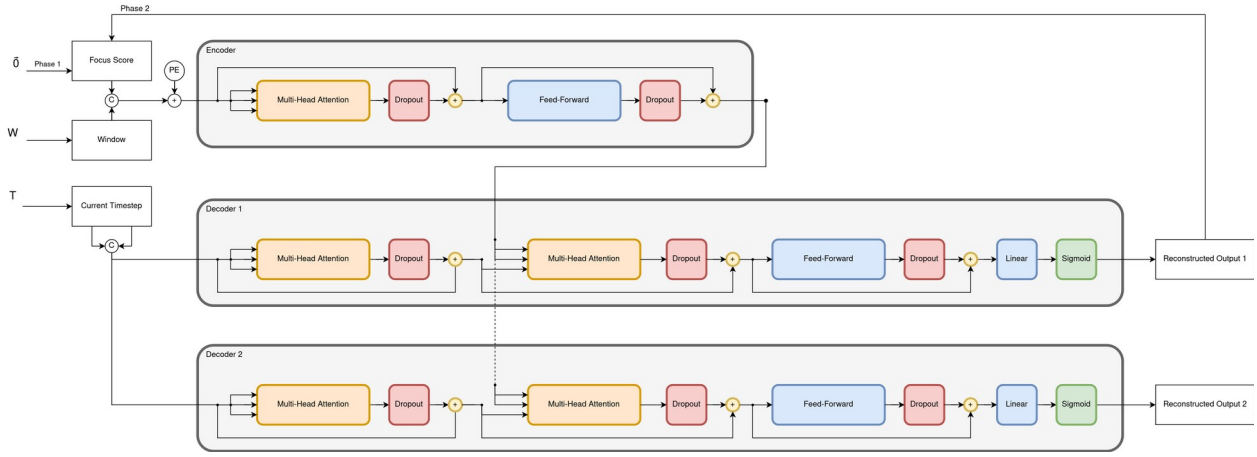

Figure 1: The TranAD+ Model

We modify the transformer-based deep learning model TranAD for anomaly detection (see Figure 1) [15]. The model's architecture trains in a self-supervised manner, removing the need for labeled data. As a result, downstream applications only require an unlabeled time-series dataset to train the model. The model also outputs anomaly scores per-timestep and per-parameter. This allows for Operations Engineers to not only be able to identify which time-steps are anomalous, but how much each parameter contributed to the flagging of the anomalous region.

Once the model is trained on the training data, the model predicts the testing data and compares it with the true testing value to obtain an anomaly score per-parameter. For each time-step, the anomaly scores are averaged across all parameters and the Peak-Over-Threshold method is used to identify which threshold should be used for labeling anomalous regions of the time-series [12]. Any averaged parameter value that is above this threshold is marked as anomalous for that time-step. Afterwards, the parameter values before averaging can be used to identify how much each parameter contributed to the flagging of the anomaly.

The TranAD architecture is modified such that the feed-forward dimension of the model is swapped out from being a bottleneck to an inverse bottleneck. The intuition behind this design change is to prevent unnecessary loss of information when dealing with large numbers of parameters, especially after one-hot encoding. Furthermore, during inference we obtain anomaly scores from both decoder outputs, call them D_1 and D_2. Taking inspiration from USAD, the per-parameter anomaly scores for the time-series are assigned to be $\mathcal{A} = \alpha D_1 + \beta D_2$, where $\alpha, \beta \in [0, 1]$ are hyperparameters [14]. The original TranAD paper assigned $\alpha = \beta = 0.5$ and the code had $\alpha = 0, \beta = 1$ [15]. We call our updated model TranAD+[1].

[1] The source code for TranAD+ is available at https://github.com/yyexela/TranADPlus.

4. Experiments

4.1 Datasets

We use five publicly available datasets in addition to the DSN dataset to compare TranAD+ with the aforementioned baseline models.

- *Soil Moisture Active Passing (SMAP)*: is a dataset from a satellite launched by NASA for monitoring Earth's soil moisture [10].
- *Mars Science Laboratory (MSL)*: is a dataset from NASA's Curiosity rover. The authors of LSTM-NDT who released this dataset state that MSL is much less predictable than SMAP and is thus a more challenging dataset for anomaly detection [10].
- *Secure Water Treatment (SWaT)*: is a dataset collected from a water treatment plant. The dataset contains sensor and actuator values during seven days of nominal operations and four days of faults [16].
- *Water Distribution (WADI)*: is a dataset collected from a water distribution testbed. The dataset contains sensor and actuator values during fourteen days of nominal operations and two days of faults. It is a larger extension of the SWaT dataset from the same research group. Some time steps contain missing information [17].
- *Server Machine Datset (SMD)*: is a dataset collected from 28 different machines over a period of five weeks. The parameters in the time-series dataset include CPU load, network usage, memory usage, etc [11].
- *Deep Space Network 1k (DSN_1k)*[2]: is a new dataset presented in this paper obtained from the Deep Space Network. The data contains both numerical and categorical information relating to various telemetry information received from the satellite. Additionally, some time-steps contain missing information.

Table 1: Dataset statistics for SMAP, MSL, SWaT, WADI, SMD, and DSN_1k.

Dataset:	SMAP	MSL	SWaT
Number of tracks	54	27	1
Total Training Length	138,004 (24.05%)	58,317 (44.16%)	495,000 (52.39%)
Total Testing Length	435,826 (75.95%)	73,729 (55.84%)	449,919 (47.61%)
Total Length	573,830	132,046	944,919
Number of Anomalies	55,922 (12.83%)	7,766 (10.53%)	54,621 (12.14%)
Number of Parameters	25	55	51
Types of Parameters	Float	Float	Float
Number of NANs	0 (0%)	0 (0%)	0 (0%)

Dataset:	WADI	SMD	DSN_1k
Number of tracks	1	28	999
Total Training Length	784,571 (81.95%)	708,405 (50%)	3,367,256 (77.76%)
Total Testing Length	172,801 (18.05%)	708,420 (50%)	962,832 (22.24%)
Total Length	957,372	1,416,825	4,330,088
Number of Anomalies	9,977 (5.77%)	29,444 (4.16%)	247,247 (25.68%)
Number of Parameters	128	38	129 (98 float, 31 byte)
Types of Parameters	Float	Float	Float, Byte
Number of NANs	3,829,522 (3.13%)	0 (0%)	43,956,130 (7.87%)

We show the total number of time-series tracks, the track lengths in the training and testing splits (and what percentage of the total dataset each makes up), the total number of anomalies (and the percentage of the testing split they make up), the number of parameters, the types of parameters, and the number of missing values (and the percentage of the total dataset they make up).

Table 1 contains statistics about each dataset, including track lengths, numbers of parameters, their types, anomalies, and the amount of missing information. The statistics demonstrate that DSN_1k is the largest public anomaly detection set available, at three times larger than the SMD dataset in terms of raw track length. In addition

[2] The DSN_1k dataset is available at https://osf.io/t9c2a/.

to having the most time-steps, DSN_1k has the most number of parameters, even before one-hot encoding the byte values. Furthermore, DSN_1k is the dataset with the largest amount of missing time-series information and is the only dataset with byte parameters. These parameters are inherently categorical and thus need special consideration during the data pre-processing stage. All of these factors make the DSN_1k dataset more modern for comparing time-series anomaly detection models, giving deep learning models more data to perform at a higher capacity.

4.2 Evaluation Metrics

We use precision, recall, AUC, and F1 score to evaluate the performance of anomaly detection between models on each dataset. Recall:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

Where *TP* is the number of true positives, *FP* is number of false positives, *TN* is number of true negatives, and *FN* is number of false negatives.

5. Results

To train the models we take each dataset and concatenate all the training tracks together for a single training track and similarly for the testing tracks. The below tables show the results of each model trained on the above datasets using the default hyper-parameters of each baseline model.

Table 2: F1, Precision, Recall and AUC for the DSN_1k dataset.

	Dataset	DSN_1k			
Model		F1	Precision	Recall	AUC
MTAD-GAT	POT	0.7033	**0.9399**	0.5619	0.7747
	Best F1	0.8445	0.7434	0.9774	0.9304
OmniAnomaly	POT	0.4087	0.2568	**1.0000**	0.5000
	Best F1	0.7434	0.7116	0.7783	0.8346
USAD	POT	0.8185	0.7484	0.9031	0.8991
	Best F1	0.8138	0.7246	0.9279	0.9030
TranAD	POT	0.8785	0.8092	0.9607	**0.9413**
	Best F1	**0.8805**	0.8951	0.8663	0.9157
TranAD+	POT	**0.9247**	0.8653	0.9929	**0.9698**
	Best F1	0.8044	**0.9662**	0.6890	0.8403

Table 3: F1, Precision, Recall and AUC for the SMAP dataset.

	Dataset	SMAP			
Model		F1	Precision	Recall	AUC
MTAD-GAT	POT	0.6718	0.9097	0.5326	0.7624
	Best F1	**0.8166**	0.7751	**0.8628**	**0.9130**
OmniAnomaly	POT	0.6205	0.6429	0.5997	0.7753
	Best F1	0.7050	**0.9541**	0.5590	0.7775
USAD	POT	0.5340	0.7884	0.4037	0.6939
	Best F1	0.5700	0.7436	0.4621	0.7193
TranAD	POT	0.6871	0.8532	0.5751	0.7803
	Best F1	0.6667	0.7506	0.5997	0.7852
TranAD+	POT	0.8424	0.8777	0.8099	0.8966
	Best F1	**0.8491**	0.7838	**0.9263**	**0.9445**

Table 4: F1, Precision, Recall and AUC for the MSL dataset.

	Dataset	MSL			
Model		F1	Precision	Recall	AUC
MTAD-GAT	POT	0.7249	0.9272	0.5950	0.7948
	Best F1	**0.9191**	**0.9282**	0.9101	**0.9509**
OmniAnomaly	POT	0.8833	0.9144	0.8542	0.9224
	Best F1	0.8833	0.9144	0.8542	0.9224
USAD	POT	0.2944	0.8069	0.1801	0.5875
	Best F1	0.5339	0.5100	0.5601	0.7483
TranAD	POT	0.5918	0.8317	0.4593	0.7242
	Best F1	0.4926	0.3302	**0.9687**	0.8691
TranAD+	POT	0.8901	0.8487	0.9356	**0.9580**
	Best F1	0.8196	0.7352	0.9258	0.9437

Table 5: F1, Precision, Recall and AUC for the SWaT dataset.

	Dataset	SWaT			
Model		F1	Precision	Recall	AUC
MTAD-GAT	POT	0.2745	0.1605	**0.9493**	0.6314
	Best F1	**0.8116**	0.9966	0.6845	0.8421
OmniAnomaly	POT	0.5791	0.4497	0.8130	0.8378
	Best F1	0.7852	0.8551	0.7259	**0.8544**
USAD	POT	0.8046	0.9921	0.6766	0.8379
	Best F1	0.8061	**0.9968**	0.6766	0.8382
TranAD	POT	0.3081	0.1850	0.9201	0.6801
	Best F1	0.7995	0.9773	0.6764	0.8371
TranAD+	POT	0.3290	0.2003	0.9201	0.7063
	Best F1	**0.8239**	0.9865	0.7074	0.8530

Table 6: F1, Precision, Recall and AUC for the SMD dataset.

	Dataset	SMD			
Model		F1	Precision	Recall	AUC
MTAD-GAT	POT	0.6698	0.5871	0.7798	0.8780
	Best F1	**0.6973**	**0.6566**	0.7432	0.8632
OmniAnomaly	POT	0.4719	0.3216	**0.8858**	**0.9024**
	Best F1	0.5028	0.5320	0.4767	0.7292
USAD	POT	0.4241	0.4292	0.4191	0.6974
	Best F1	0.4334	0.4475	0.4201	0.6988
TranAD	POT	0.5460	0.4947	0.6092	0.7911
	Best F1	0.5581	0.5191	0.6034	0.7896
TranAD+	POT	0.7129	0.5784	**0.9289**	**0.9498**
	Best F1	**0.8509**	**0.8272**	0.8759	0.9340

Table 7: F1, Precision, Recall and AUC for the WADI dataset.

	Dataset	WADI			
Model		F1	Precision	Recall	AUC
MTAD-GAT	POT	0.1679	0.0923	**0.9327**	0.6850
	Best F1	0.1680	0.0923	0.9327	0.6853
OmniAnomaly	POT	0.1920	0.1200	0.4787	0.6318
	Best F1	**0.5867**	0.9983	0.4155	**0.7077**
USAD	POT	0.1568	0.0861	0.8736	0.6531
	Best F1	0.2789	**1.0000**	0.1621	0.5810
TranAD	POT	0.1674	0.0920	0.9327	0.6843
	Best F1	0.2788	0.9975	0.1621	0.5810
TranAD+	POT	0.5084	0.5739	0.4563	**0.7178**
	Best F1	0.2788	0.9975	0.1621	0.5810

Table 2 through Table 7 show the F1, Precision, Recall, and AUC scores of the baseline methods as well as TranAD+ for the studied datasets. We see that TranAD+ outperforms the baseline F1 scores on DSN_1k, SMAP, SWaT, and SMD. OmniAnomaly scored the best F1 on WADI and MTAD-GAT scored the best F1 on MSL. However, on both WADI and MSL, TranAD+ obtained the highest AUC.

One thing to note is that sometimes the brute-force search for a threshold (Best F1) was outperformed by POT. This is likely due to the brute-force search not being extensive enough to identify the optimal threshold, as a simple grid-search was performed. This is the case for TranAD+'s result on the WADI dataset, as an example.

One clear observation is that TranAD+ tends to perform better on larger datasets. This explains TranAD+'s poor performance on the MSL dataset, the smallest dataset considered. TranAD+ was beaten by both MTAD-GAT and OmniAnomaly for best F1 on this dataset.

The large gap in performance between TranAD and TranAD+ is explained in part by the inverse bottleneck, but also in the training methodology. The original code for TranAD trains for only 5 epochs with a fixed α and β for the decoder outputs. This is insufficient in part as training a transformer-based network requires more steps, and searching over more hyperparameters is bound to improve results.

6. Conclusions

This work successfully demonstrates that Operations Engineers can benefit from modern deep learning architectures for automatic time-series anomaly detection. By presenting the challenging new dataset DSN_1k, we show that a transformer-based auto-encoder model is able to learn the time-series data in a self-supervised manner to then be able to detect if incoming data is anomalous. Our model, TranAD+, obtains the best F1 score on four out of the six presented datasets, including DSN_1k.

The direct application of this work is that Operations Engineers and their employers should look towards implementing these kinds of models in their pipelines to offload the burden of monitoring incoming data to computers. These models will then provide Operations Engineers specific times in incoming data that are labeled as potentially anomalous events with the parameters that Operations Engineers should look at. This allows existing Operations Engineering teams to be able to handle a significantly larger amount of data, preparing industry leaders such as JPL to be able to handle increased data transmission rates from improved technology and additional missions.

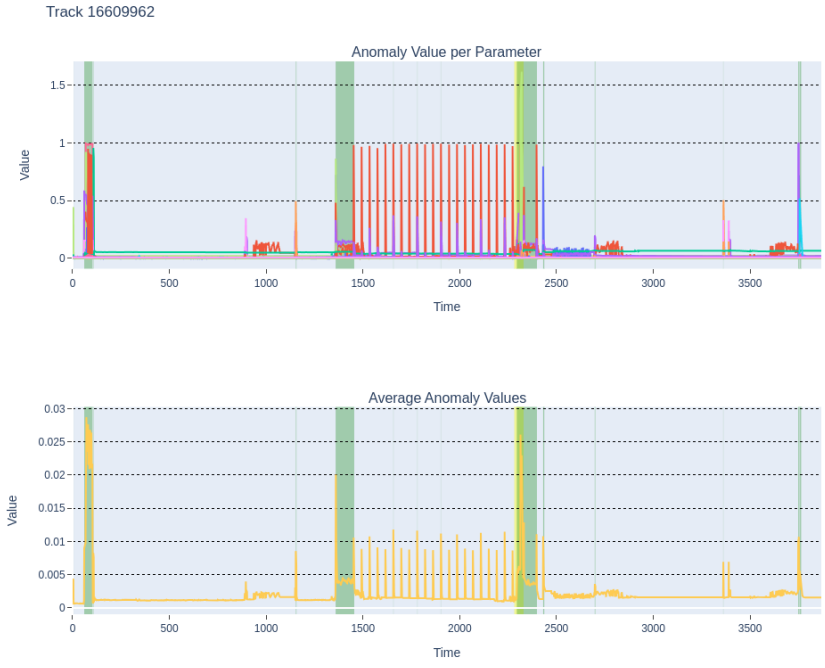

Figure 2: Example of TranAD+ labeling anomalous regions (green) on an anoamly labeled (yellow) track from DSN 1k.

As an example of a downstream application, incoming time-series data can immediately be labeled as anomalous or not and displayed in an easy-to-read figure (see Figure 2). This figure also shows the contributions of individual parameters to the anomaly regions. It is then possible to extract these values and present Operations Engineers with a text-based output of the parameter contributions.

The static output can be further modified by passing it into a GPT-based model to produce pre-generated anomaly reports. One can easily imagine that with some prompt engineering, an LLM can generate detailed outputs. Prompt engineering is flexible, so the variability in downstream applications and reports that could be generated allows for seamless pipeline integration and allows valuable anomaly information to be presented in a digestible format.

LLM Input

```
Your task is to generate a report explaining anomalous behavior in a time-
series dataset. Make the report concise, yet understandable for an operations
engineer. You have the following information about a detected anomaly:

Anomaly 8 parameters:
  CAR_TRK_LOOP_LOCK_STATUS (54.49%)
  SYMBOL_TRK_LOOP_STATUS (23.99%)
  RESIDUAL_DATA_POWER (4.35%)
Contribute to 82.83% of anomaly score at times
  "2023-05-29 19:23:01" to "2023-05-29 19:24:37"
  (index 3125 to 3144)
```

LLM Output

```
Anomaly Report

Anomaly Description:

An anomalous pattern has been detected in the time-series dataset between
    "2023-05-29 19:23:01" and "2023-05-29 19:24:37" (index 3125 to 3144).
    This anomaly accounts for 82.83% of the total anomaly score.

Key Contributing Parameters:

The following parameters contribute significantly to the anomaly:

    CAR_TRK_LOOP_LOCK_STATUS: 54.49%
    SYMBOL_TRK_LOOP_STATUS: 23.99%
    RESIDUAL_DATA_POWER: 4.35%

These three parameters cumulatively contribute to the majority of the anomaly
    score, indicating a potential issue related to tracking loop lock status,
    symbol tracking, and residual data power.

Recommendations:

    Investigate the system's tracking loop lock status and symbol tracking
        functionality during the specified time period.
    Review the residual data power levels to determine if there were any
        unusual fluctuations.
    Analyze the system logs and performance metrics to identify any
        underlying causes for the anomaly.

Next Steps:

    Further investigation is required to determine the root cause of the
```

```
        anomaly.
    Additional data analysis and system checks may be necessary to resolve
        the issue.

Please review the system and investigate the contributing parameters to
    resolve the anomaly.
```

We hope to see that industry adopts the powerful models coming out of deep learning for anomaly detection to increase the reliability of systems and to resolve anomalies at quicker speeds than those capable by humans alone.

Acknowledgements

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

[1] Wikipedia Authors. *NASA Deep Space Network*. Sept. 2024. url: https://en.wikipedia.org/wiki/NASA_Deep_Space_Network.
[2] NASA Office of Inspector General. *Audit of NASA's Deep Space Network*. June 2023. url: https://www.nasa.gov/wp-content/uploads/2023/08/ig-23-016.pdf.
[3] Kyongsik Yun, Rishi Verma, and Umaa Rebbapragada. "Time Series Comparisons in Deep Space Network". In: *arXiv preprint arXiv:2111.01393* (2021).
[4] Psyche Press Kit. *Deep Space Optical Communications (DSOC)*. June 2023. url: https://www.jpl.nasa.gov/press-kits/psyche/dsoc/.
[5] National Aeronautics and Space Administration. *Deep Space Optical Communications (DSOC) Technology Demonstration*. July 2023. url: https://www.nasa.gov/wp- content/uploads/2023/07/dsoc-fact-sheet-06152023.pdf?emrc=b01937.
[6] Laurie Leshin. *JPL Workforce Update*. Feb. 2024. url: https://www.jpl.nasa.gov/news/jpl-workforce-update/.
[7] Frank E Grubbs. "Procedures for detecting outlying observations in samples". In: *Technometrics* 11.1 (1969), pp. 1–21.
[8] Hadi Banaee, Mobyen Uddin Ahmed, and Amy Loutfi. "Data mining for wearable sensors in health monitoring systems: a review of recent trends and challenges". In: *Sensors* 13.12 (2013), pp. 17472–17500.
[9] Anita K Jones and Robert S Sielken. "Computer system intrusion detection: A survey". In: *Computer Science Technical Report* (2000), pp. 1–25.
[10] Kyle Hundman et al. "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 387–395.
[11] Ya Su et al. "Robust anomaly detection for multivariate time series through stochastic recurrent neural network". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2828–2837.
[12] Alban Siffer et al. "Anomaly detection in streams with extreme value theory". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 1067–1075.
[13] Hang Zhao et al. "Multivariate time-series anomaly detection via graph attention network". In: *2020 IEEE international conference on data mining (ICDM)*. IEEE. 2020, pp. 841–850.
[14] Julien Audibert et al. "Usad: Unsupervised anomaly detection on multivariate time series". In: *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020, pp. 3395–3404.
[15] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. "Tranad: Deep transformer networks for anomaly detection in multivariate time series data". In: *arXiv preprint arXiv:2201.07284* (2022).
[16] Jonathan Goh et al. "A dataset to support research in the design of secure water treatment systems". In: *Critical Information Infrastructures Security: 11th International Conference, CRITIS 2016, Paris, France, October 10–12, 2016, Revised Selected Papers 11*. Springer. 2017, pp. 88–99.
[17] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. "WADI: a water distribution testbed for research in the design of secure cyber physical systems". In: *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*. 2017, pp. 25–28.