Semi-Supervised Deep Learning for Spacecraft Anomaly Detection

Tai Hung (Henry) Lu

Master of Science

School of Computer Science
McGill University
Montreal, Quebec, Canada

August 2022

A thesis submitted to McGill University in partial
fulfillment of the requirements of the degree of
Master of Science

Abstract

Space is a challenging environment, not only for humans but also for machines. Reliably and efficiently detecting anomalies that appear on spacecraft is a key step towards preventing the loss of onboard components, the mission, or even lives. Spacecraft health monitoring practices today mainly rely on prior knowledge of complex systems, such as checking if telemetry values fall outside pre-established limits. Deep learning techniques offer a more data-driven solution, capable of processing enormous amounts of telemetry and producing valuable insights. Several such methods have emerged which learn patterns of nominal behaviour in a semi-supervised manner, without the need for anomaly labels. In this thesis we explore semi-supervised deep learning-based methods for detecting spacecraft anomalies. Inspired by three recent works, we examine variations of recurrent neural network architectures which adopt elements of generative adversarial networks and graph attention networks to predict anomaly scores from input sequences of data. These models are paired with techniques for calculating anomaly thresholds and then evaluated on a real dataset of Near-Earth Object Surveillance Satellite (NEOSSat) telemetry and anomalies. Our experiments show that models built with gated recurrent units achieve the best overall performance when combined with *Peaks-Over-Threshold* for setting thresholds plus a subsequent anomaly pruning step. In addition to detecting almost all of the same anomalies as humans with a manageable false positive rate, such models can leverage graph attention layers to produce attention scores as further tools for investigating anomalies.

Abrégé

L'espace est un environnement exigeant, non seulement pour les humains mais aussi pour les machines. La détection fiable et efficace des anomalies qui apparaissent sur les engins spatiaux est une étape clé pour éviter la perte de composants à bord, de la mission, voire de vies humaines. Aujourd'hui, les pratiques de surveillance de l'état de santé des engins spatiaux reposent principalement sur la connaissance préalable de systèmes complexes, par exemple en vérifiant si les valeurs de télémétrie sortent des limites préétablies. Les techniques d'apprentissage profond offrent une solution plus axée sur les données, capable de traiter d'énormes quantités de télémétrie et de produire des informations précieuses. Plusieurs méthodes de ce type ont vu le jour, qui apprennent à modéliser et à prédire le comportement attendu d'un système de manière semi-supervisée, sans avoir besoin d'étiquettes d'anomalie. Dans cette thèse, nous explorons des méthodes semi-supervisées basées sur l'apprentissage profond pour détecter les anomalies dans la télémétrie des engins spatiaux. Inspirés par trois travaux récents, nous examinons des variantes d'architectures de réseaux neuronaux récurrents qui adoptent des éléments des réseaux antagonistes génératifs et des réseaux d'attention de graphes pour prédire les scores d'anomalie à partir de séquences de données d'entrée. Ces modèles sont combinés à des techniques de calcul des seuils d'anomalie, puis évalués sur un ensemble réel de données de télémétrie et d'anomalies du Near-Earth Object Surveillance Satellite (NEOSSat). Nos expériences montrent que les modèles construits avec des unités récurrentes à portes obtiennent les meilleures performances globales lorsqu'ils sont combinés avec *Peaks-Over-Threshold* pour définir les seuils et une étape ultérieure d'élagage des anomalies. En plus de détecter presque toutes les mêmes anomalies que les humains avec un taux de faux positifs gérable, ces modèles permettent d'exploiter les couches d'attention des graphes pour produire des scores d'attention comme outils supplémentaires d'investigation des anomalies.

Contributions

The writing of this thesis, as well as the accompanying research, software programming, experiments, and technical analysis, have been done soley by Tai Hung (Henry) Lu under the supervision of Professor David Meger. Nathaniel Cziranka-Crooks and Tyler Hrynyk have assisted in retrieving the dataset and labels described in Section 3.1. Additionally, Tyler Hrynyk has performed the verification of false positives mentioned in Section 7.2.

Acknowledgements

I've often heard astronauts say it takes a village to send them into space. I'm no astronaut and I haven't been to space yet, but I have felt a similar level of support from a small community who believed in my mission to write a Master's thesis.

To Professor David Meger, your guidance and wisdom from the very beginning of this journey is the reason I am able to finish and submit this work with a sense of accomplishment and pride. Thank you for your patience as I learned what it meant to be a researcher in this field. At times when I felt I was at a mental standstill, our discussions always left me with renewed motivation and ideas.

To Nathaniel Cziranka-Crooks and Tyler Hrynyk, thank you for introducing me to the world of satellite operations and the intricacies of NEOSSat. Your availability, knowledge, and friendships have turned what should have been a bumpy ride into a smooth experience.

To Jean-François Cusson, thank you for supporting me as I balanced my priorities and for giving me the freedom to manage my own career. I felt your encouragement and confidence on a daily basis.

To my reviewers: Chelsea Taylor, Gabriel Descôteaux, Harley Wiltzer, thank you for taking the time to read, comment on, and challenge parts of my thesis. Your feedback has been invaluable.

Thank you to the many others who pushed me to start, carry on, and complete this milestone in my life.

Lastly, to my parents, thank you for raising me.

Contents

List of Figures

List of Tables

Acronyms

ATHMoS	Automated Telemetry Health Monitoring System
BPTT	Backpropagation Through Time
CSA	Canadian Space Agency
CNES	Centre national d'études spatiales
CNN	Convolutional Neural Network
DAGMM	Deep Autoencoding Gaussian Mixture Model
DTW	Dynamic Time Warping
DBSCAN	Density Based Spatial Clustering in Applications with Noise
ESA	European Space Agency
EWMA	Exponentially-Weighted Moving Average
FDIR	Fault Detection, Isolation, and Recovery
GAN	Generative Adversarial Network
GAT	Graph Attention Network
GCN	Graph Convolution Network
GEO	Geostationary Earth Orbit
GMM	Gaussian Mixture Mode
GNN	Graph Neural Network
GPD	Generalized Pareto Distribution
GPS	Global Positioning System
GRU	Gated Recurrent Unit
GSOC	German Space Operations Center
ISS	International Space Station

JAXA	Japan Aerospace Exploration Agency
JPL	Jet Propulsion Laboratory
JS	Jensen-Shannon (divergence)
JWST	James Webb Space Telescope
KARI	Korea Aerospace Research Institute
KL	Kullback-Leibler (divergence)
LoOP	Local Outlier Probabilities
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MLE	Maximum Likelihood Estimation
MPPCA	Mixture of Probabilistic Principal Component Analyzers
MSE	Mean Squared Error
MSL	Mars Science Laboratory
MTAD-GAT	Multivariate Time-series Anomaly Detection via Graph Attention Network
NASA	National Aeronautics and Space Administration
NEOSSat	Near-Earth Object Surveillance Satellite
OC-SVM	One-Class Support Vector Machines
OOL	Out-Of-Limits
PCA	Principal Component Analysis
POT	Peaks-Over-Threshold
ReLU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SAA	South Atlantic Anomaly
SEE	Single Event Effect
SMAP	Soil Moisture Active Passive
TéSA	Télécommunications Spatiales et Aéronautiques
UNOOSA	United Nations Office for Outer Space Affairs
VAE	Variational Autoencoder
WGAN-GP	Wasserstein-GAN with Gradient Penalty

1

Introduction

The amount of resources invested in large-scale space missions is astronomical. Endeavours such as the International Space Station (ISS) and the more recent James Webb Space Telescope (JWST) are multibillion-dollar projects representing decades of work by international collaborators. While excessive efforts are made to ensure the success of such projects, space is a harsh and largely unknown environment; so many things could go wrong, leading to interruptions, setbacks, or even the premature end of a mission. Humans are currently in an era of renewed interest in space exploration, with government agencies and private corporations turning their attention towards the Moon, Mars, and beyond (Canada, 2019; NASA, 2019). As our species pushes the limits and ventures deeper into unexplored territory, more surprises and more chances of failure can be expected along the way.

Fault detection, isolation, and recovery (FDIR) systems have been developed and implemented on spacecraft to improve the stability of missions and increase autonomy (Olive, 2012). The first step in these systems is detecting and investigating any anomalous behaviour that could indicate potential failures. Detecting and resolving these early could prevent catastrophic outcomes such as loss of control, and ultimately the mission (Morgan, 2005). Even temporary failures interrupt the collection of valuable science data. The sooner anomalies are discovered, the sooner the spacecraft can return to its nominal mission. This is easier said than done as the space setting presents unique challenges compared to detecting anomalies on Earth-based systems.

Special environmental factors that pose hazards to spacecraft include extreme temperature cycles, risk of space debris, and charged particle radiation (Fujimaki et al., 2005). In addition to the harsh environment, the remoteness of spacecraft prevents them from being directly inspected and repaired when needed. Instead, operations engineers monitor the health of spacecraft by analyzing telemetry signals for off-nominal behaviour. This is an intimidating task since individual spacecraft can have thousands of telemetry channels (Carlton et al., 2018)—including voltages, currents, temperatures, status indicators, and other sensor readings—and more space assets are being launched into orbit now than ever before, as shown in Figure 1.1. The enormous amount of data that must be monitored necessitates an automated and reliable solution.

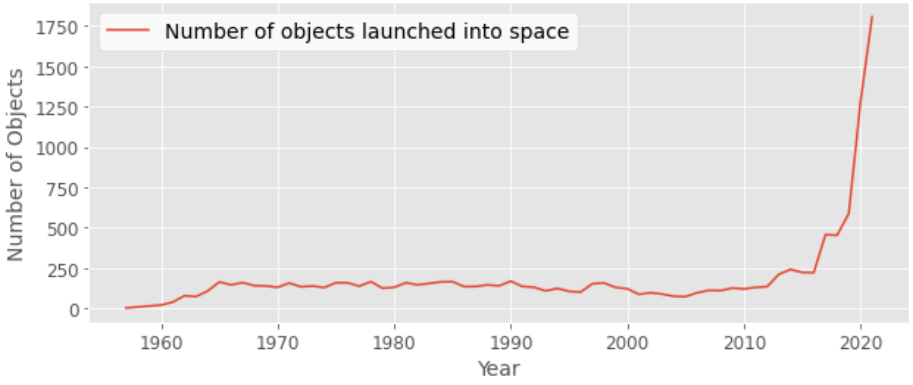

Figure 1.1: Number of objects launched into space between 1957 and 2021, compiled by the United Nations Office for Outer Space Affairs (UNOOSA, 2022).

Traditionally, automated detection systems have incorporated some combination of the following: cross-checking measurements between redundant pieces of hardware; estimating faulty states with a technique like the Kalman filter (Kalman, 1960); and raising alarms when telemetry signals exceed or fall below predefined limits (Wander and Förstner, 2013). The latter, referred to as out-of-limits (OOL), is still an attractive practice today because of its low computational cost and simplicity (Hundman et al., 2018; Martínez-Heras, 2012). However, this method relies heavily on *a priori* knowledge of complex spacecraft systems—knowledge that is acquired through theory and experimentation on the ground but may not reflect the spacecraft behaviour once in orbit (Fujimaki et al., 2005). Furthermore, experience from previous or existing missions cannot be straightforwardly applied

as the variation in workmanship, operational environment, and configuration means that no two spacecraft are identical (Carlton et al., 2018). Establishing and updating limits for every channel is an expensive endeavour that also fails to account for anomalies that fall within such limits (exemplified by Anomaly 2 in Figure 1.2). A better approach could be to take advantage of the huge volume of telemetry through a data-driven solution.

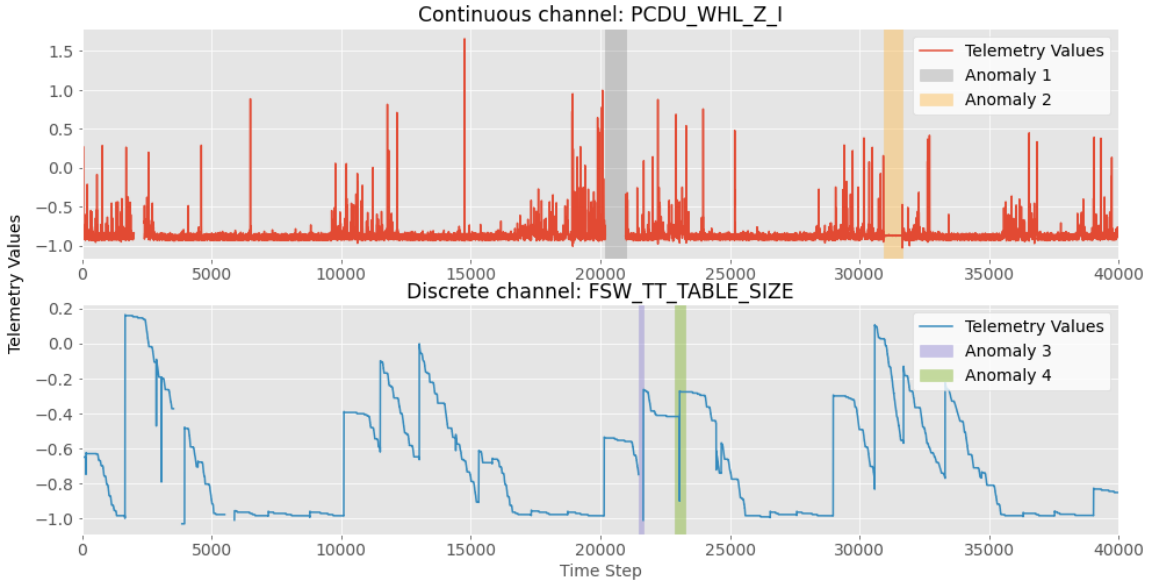

Figure 1.2: Samples of true anomalies among two telemetry channels: wheel current (top) and size of command buffer (bottom). The time steps shown are from two different time periods. Anomaly 2 is an example where OOL is ineffective because those values cannot be distinguished from nominal telemetry by setting upper and lower limits.

Deep learning algorithms have shown success in making intelligent predictions by learning from large datasets. While conventional "shallow" methods require manual extraction of features from raw data by domain experts, deep learning is able to capture meaningful representations which are tailored specifically to the desired goal, such as anomaly detection (LeCun et al., 2015; Pang et al., 2020).

For deep learning to be effective at classification tasks, a well-annotated dataset is often required (Rolnick et al., 2018). Anomaly labels are rare and incomplete in a space mission, especially during its early phases. New anomalies can exhibit completely novel characteristics which are absent from the anomalies identified in training data, limiting the

usefulness of such data in supervised learning settings. The reliance on anomaly labels precludes supervised methods, but alternative semi-supervised techniques have emerged that require only nominal samples in training data, which are more abundant and more obvious to identify. These techniques reduce the chances and consequences of mislabeling data in anomaly detection tasks (Chalapathy and Chawla, 2019).

1.1 Contributions of this Work

This thesis studies the potential of semi-supervised deep learning approaches in detecting anomalies from spacecraft telemetry. Our aim is to determine if such methods achieve sufficient performance to be considered for adoption into operational settings while pointing out the benefits and barriers to doing so. The main contributions of this work are summarized as follows:

- We explore different combinations of neural network architectures and thresholding techniques, inspired by recent trends in deep learning-based anomaly detection for time series data.
- We evaluate our chosen methods of interest on a real dataset of Near-Earth Object Surveillance Satellite (NEOSSat) telemetry and anomalies gathered by operations engineers over the course of one year.
- We identify the key components that drive strong anomaly detection performance in our problem setting as well as the unique challenges we face.

1.2 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 introduces the task of anomaly detection in space and reviews recent, related work on time series signals. Chapter 3 explains the common setup of our experiments, including details of the NEOSSat dataset and evaluation metrics. Chapters 4, 5, and 6 cover the technical background and experimental results of the three different anomaly detection methods we are evaluating. Chapter 7 compares the performance of each method and discusses our overall findings. We provide a summary and avenues for future work to conclude our thesis in Chapter 8.

2

Background and Related Work

An investigation into anomaly detection methods first requires an understanding of the nature of anomalies and existing detection systems used today by satellite operators. This chapter defines common terminology used throughout the thesis. An extensive review of relevant anomaly detection methods and related studies on spacecraft anomalies is presented as a survey of the current state of the art. Technical background for the methods used in our experiments is not covered here, but rather provided in their respective chapters dedicated to those methods.

2.1 Preliminaries

2.1.1 Spacecraft Anomalies

Anomalies are patterns in data that do not conform to a predefined notion of nominal behaviour (Chandola et al., 2009). Borrowing notation from Ruff et al. (2021), we can state this more formally using probabilistic terms: given the data space $\mathcal{X} \subseteq \mathbb{R}^D$ of some task, let the probability measure $\mathbb{P}^+$ on $\mathcal{X}$ represent the concept of nominal behaviour of that task. Then an anomaly is a data point that deviates from this notion of normality, and falls in a low probability region of $\mathbb{P}^+$. Assuming that the probability density function p^+ corresponding to $\mathbb{P}^+$ exists, the set of anomalies can be defined as

$$\mathcal{A} = \{\boldsymbol{x} \in \mathcal{X} \mid p^+(\boldsymbol{x}) \leqslant \tau\}, \tag{2.1}$$

where $\tau \geqslant 0$ is some threshold that represents the boundary of the region of nominal data.

Anomalies fall under the same umbrella as outliers and novelties, but the exact definitions and how they are handled in detection methods vary across application domains. For example, outliers may be considered as "measurement errors" to be removed during preprocessing while novelties might be new observations that must be added to training data (Ruff et al., 2021). For our purposes, we do not need to make distinctions between these terms, and instead focus on the concept of anomalies defined at the beginning of this section. We identify three different ways in which these anomalies may appear in a dataset (Chandola et al., 2009):

- As *point anomalies*—individual data points that are anomalous with respect to their surrounding data. These are the types of anomalies typically caught by limit-checking approaches.
- As *contextual anomalies*—data points that are anomalous conditional to a certain context, but not anomalous in other contexts. Detection of this class of anomalies requires appropriate contextual data to be available; for spacecraft this might mean orbital parameters or space weather variables.
- As *collective anomalies*—groups of data points which are anomalous when considered together but may not be when considered as individual data points.

Chandola et al. (2009) give an example of a collective anomaly as an abnormal sequence of values in a single time series signal, but the collection can be across multiple signals too. As Boumghar et al. (2021) point out, spacecraft behaviour is not only defined by the values of a sole telemetry signal, it is also defined from the collective status of all telemetry.

On spacecraft, anomalies can be traced to several possible factors. Firstly, the space environment contains high-energy charged particles due to solar wind and cosmic rays (Galvan et al., 2014). Near Earth, these charged particles are captured in zones known as Van Allen radiation belts from their interactions with Earth's magnetosphere. Satellites orbiting through these zones are susceptible to complications like spacecraft charging, where surface or internal build-up of electrons can discharge and damage onboard instruments (Fennell et al., 2001). A region of particularly high radiation levels due to a weak local

magnetic field is known as the South Atlantic Anomaly (SAA; Pavón-Carrasco and De Santis, 2016). This region can be so hazardous to onboard electronics that measures are taken to reduce science activities during SAA orbit intersections (Space Telescope Science Institute, 2012). Another product of charged particle radiation is a Single Event Effect (SEE), which occurs when an individual particle carries and deposits enough charge to disrupt a sensitive component (Baker, 2000). Its effect can vary from "soft" errors such as a change of state of an integrated circuit to permanent failures of electronics due to a latchup or burnout (Fleetwood et al., 2000).

Space also offers challenging thermal conditions caused by exposure and lack thereof to the Sun (Morgan, 2005). Onboard instruments only operate within a narrow range of temperatures, and will cease to function if the temperatures become too extreme. The cycling between hot and cold is an additional hazard to components through the stress from uneven thermal expansion and contraction. Careful thermal regulation must be put in place to prevent temporary or permanent faults.

Human decisions and operations are further sources of anomalies. Mistakes can be made in all phases of a mission: in design choices, during assembly and integration of the spacecraft, and post-launch when sending commands. These mistakes can manifest as benign or severe anomalies at unpredictable times. Even human activities external to the mission, whether intentional or not, can have considerable impacts. Orbital debris such as defunct spacecraft, discarded rocket stages, and fragments of destroyed satellites poses an ever-increasing threat to space missions. A more thorough discussion of different satellite anomalies that can be caused due to the dynamic space environment or human activity is found in (Galvan et al., 2014), citing examples from past missions.

While not all anomalies are problematic (Geiger et al., 2020), in the space context, the risk of anomalies leading to irrecoverable failures means that any such behaviour must be detected and investigated without significant delay.

2.1.2 Spacecraft Telemetry

Anomalies which occur on unmanned spacecraft must be detected from telemetry (or in some cases, a lack thereof). This data consists of information related to the health and state

of the spacecraft, called *housekeeping telemetry*, as well as data collected by the onboard science instruments, like images (Zacchei et al., 2003). Science data is not utilized in any of the anomaly detection methods we explore, since scientific objectives can vary greatly and we want to find a solution that generalizes well across spacecraft. Therefore we primarily refer to housekeeping data when we talk about telemetry. Table 2.1 shows the types of housekeeping data we will use in our experiments.

		Number of Channels		
Packet ID	Description	Discrete	Continuous	Total
GPS1	GPS status and readings	46	15	61
GPS2	GPS status and readings	46	15	61
PCDU_ST	Power distribution unit statuses	30	0	30
PCDU_I	Power distribution unit currents	0	17	17
RX_MY	Transponder voltages, statuses	7	9	16
RX_PY	Transponder voltages, statuses	7	9	16
ST_TLM	Attitude sensor telemetry	19	25	44
FSW_TT	Time-tagged command statuses	21	0	21
BOT	Back orbit telemetry statuses	39	0	39
DBG_SCI	Readout electronics state variables	18	2	20
DBG_ST	Readout electronics state variables	17	2	19
Total		250	94	344

Table 2.1: Types of telemetry packets used in our experiments and the number of channels in each. A *packet* refers to a collection of related channels.

Spacecraft telemetry is made up of thousands of signals recorded by various sensors and onboard components (Carlton et al., 2018). These time series signals, or *channels*[1], are heterogeneous, non-stationary, and noisy (Hundman et al., 2018). They can be categorized as discrete, such as status indicators or counters; or continuous, like temperature or voltage readings (Yairi et al., 2017)—samples of both categories are displayed in Figure 1.2. Channels have different sampling frequencies and diverse ranges of values. Since spacecraft often have multiple operational modes, the distribution of data is also multimodal in nature. Among the numerous channels, irrelevant signals need to be picked out

[1]Some works prefer to use the term *parameter* to describe a telemetry signal, which we refrain from doing since that term is sufficiently overloaded with meanings in the machine learning field.

and discarded since an abundance of irrelevant features may impede a model's ability to distinguish between nominal and abnormal data points (Zimek et al., 2012).

Telemetry is acquired when a spacecraft is in contact with a ground station, either directly or through relay satellites. This means that spacecraft engineers may not be immediately aware of anomalous behaviour if it occurs between ground contacts or there is a significant backlog of telemetry which cannot be downloaded in a single pass. Communication delays can also arise as a result of the spacecraft's distance from Earth if travelling far beyond Earth's orbit. Complications in transmission can lead to noisy or missing data samples that must be corrected through common signal preprocessing steps (Pilastre, Boussouf, et al., 2020). All of these characteristics displayed by spacecraft data must be considered when analyzing them in order to find anomalies. Fortunately, once an anomaly is detected, it can usually be associated with unique indicators or *signatures* in telemetry that help spacecraft operators identify reoccurring cases.

2.1.3 Anomaly Detection

We formally state the anomaly detection problem, continuing from the notation defined in Equation 2.1 and (Ruff et al., 2021). If $\mathbb{P}$ is the true distribution that generates data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, the objective is to develop a model that is able to classify whether a new test data point $\tilde{\boldsymbol{x}} \in \mathcal{X}$ is an anomaly, or in other words whether $\tilde{\boldsymbol{x}} \in \mathcal{A}$. Such a model can be derived from human expertise (i.e., knowledge-based), or learned from $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ (i.e., data-driven). Apart from spacecraft anomalies, this area of research also addresses computer network intrusion detection, fraud detection in bank transactions, medical diagnoses, and finding anomalous patterns from Earth science data (Aggarwal, 2013). While the overarching problem is the same across domains, a variety of methods can be applied depending on the type of data used and availability of labels.

Inputs to anomaly detection methods can be image-based, sequential, or graph-structured; different techniques have proven effective for each. Pang et al. (2020) observe that current anomaly detection methods detect from a single type of input, but more complex anomalies might only be found through analyzing multiple data types together. For time series, there is a further distinction between methods that analyze univariate signals versus those

that can handle multivariate data (Blázquez-García et al., 2021). No matter the input type, anomaly detection methods tend to output binary labels or anomaly scores that represent the degree of anomalousness (Aggarwal, 2013). The latter is preferred as they can be used to rank anomalies for subsequent investigation.

Anomaly detection methods can also be categorized by their level of supervision. *Supervised* techniques leverage data that is labeled as nominal or anomalous in order to classify new data points. *Semi-supervised* approaches only use either labeled nominal data for training, or partial labels of both nominal and anomalous data (Ruff et al., 2020). Pang et al. (2020) point out that some works introduce the former case as *unsupervised*; however, we consider a model as unsupervised when it does not make use of any labels at all. These models depend heavily on assumptions on distributions of abnormalities to be effective.

Challenges

Even when sufficient labels are available to attempt supervised learning, *a priori* knowledge of anomalies is always going to be incomplete. While we can look for certain anomalies that we predict may happen or have occurred previously, the difficulty lies with the unexpected and novel anomalies. Without knowing their signatures, learning to detect these anomalies is not trivial (Geiger et al., 2020).

When we stated our definition of anomaly earlier, we assumed some boundary existed between nominal and anomalous behaviour. Over the lifetime of a space mission, this boundary can evolve due to gradual or sudden degradation of hardware from the various space hazards listed in Section 2.1.1. Spacecraft behaviour also changes during different operational modes, and when transitioning through different phases of the mission. Ahn et al. (2020) give an example of a lunar orbiter that experiences different disturbances on its way to the moon and throughout the manoeuvres it takes to insert into its final orbit. These factors mean that the notion of nominal behaviour must be recalibrated while a spacecraft is in operations.

One of the goals of developing an anomaly detection system is to assist spacecraft operators in monitoring the health of satellites. A barrier to adopting such a system comes from trusting its results. Failing to detect anomalies or raising too many false alarms that

must be investigated increases the burden on operators rather than reducing it. Once an anomaly is predicted, another challenge to address is how to identify the possible root causes. Deep learning methods in particular have reputations as black box models that produce predictions with low interpretability (Castelvecchi, 2016).

2.2 Data-Driven Methods for Detecting Spacecraft Anomalies

This section contains a review of data-driven methods specifically for detecting anomalies from spacecraft telemetry. These are alternative solutions mainly proposed by researchers working with their national space agencies.

Detecting Changes in Causal Associations Fujimaki et al. (2005) hypothesize that the behaviour of spacecraft is characterized by the causal associations in the system, and unexpected changes in such associations indicate anomalies. To extract causal associations, they first map telemetry containing n time series channels into a nonlinear space $\mathcal{H}$ consisting of product features of degree d :

$$\Phi : \mathcal{X} = \mathbb{R}^n \rightarrow \mathcal{H} = \mathbb{R}^m.$$

The dimensionality of feature space $\mathcal{H}$ is given by (Schölkopf and Smola, 2003)

$$m = \frac{(n+d-1)!}{d!(n-1)!}. \tag{2.2}$$

In practice the polynomial kernel is used since the feature space computation is intractable when n and d are large (see Equation 2.2). Kernel principal component analysis (PCA; Schölkopf et al., 1998) is then applied to find the directions of principal axes in the high dimensional space $\mathcal{H}$. These directions are shown to alter drastically if non-trivial changes occur in the causal relationships of the system in the original data space $\mathcal{X}$; therefore they are representative of the causal associations. An anomaly score θ is calculated as the dif-

ference between directional vectors of nominal data $\boldsymbol{v}^1$ and of test data $\boldsymbol{v}^2$:

$$\theta = \arccos \left| \left\langle \boldsymbol{v}^1, \boldsymbol{v}^2 \right\rangle \right|,$$

where $\langle \cdot, \cdot \rangle$ is the dot product operator. To evaluate its effectiveness, this method is applied on simulated telemetry of an orbital transfer vehicle provided by Japan Aerospace Exploration Agency (JAXA).

Novelty Detection Martínez-Heras (2012) propose a telemetry monitoring paradigm for spacecraft using novelty detection. The term novelty in this case is preferred over anomaly to include behaviour that is nominal but in a new way. Their approach uses Local Outlier Probabilities (LoOP; Kriegel et al., 2009) to find novel data points based on the density of its k-nearest neighbours. LoOP is derived from Local Outlier Factor (LOF; Breunig et al., 2000) and offers advantages in being less sensitive to the choice of hyperparameter k as well as outputting a score that translates to the probability that novel behaviour is detected. Each data point consists of simple statistical features—average, standard deviation, minimum, and maximum—over a fixed time period. A prototype of this novelty detection was applied on the XMM-Newton mission and successfully reported novelties which were confirmed by the flight control team at the European Space Agency (ESA). The idea was later integrated into the XMM Early Warning System (XEWS; Kirsch, 2012) to assist in detecting early degradation of spacecraft components, as well as in the MetOp satellites operated by the European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT; Trollope et al., 2018).

NOSTRADAMUS The New Operational SofTwaRe for Automatic Detection of Anomalies based on Machine-learning and Unsupervised feature Selection (NOSTRADAMUS) is an anomaly detection solution jointly developed by le Centre national d'études spatiales (CNES)—the French space agency—and a cooperative laboratory known as Télécommunications Spatiales et Aéronautiques (TéSA; Fuertes et al., 2016; Fuertes et al., 2018). Their system consists of three steps. First, a variety of features such as mean, standard deviation, skewness, kurtosis, and energy are computed from windows of housekeeping telemetry. Then PCA is applied to reduce the dimensionality of data. The final step is to learn a decision boundary separating nominal data points from anomalies using one-class support

vector machines (OC-SVM; Schölkopf et al., 1999). NOSTRADAMUS was deployed in parallel with existing telemetry surveillance systems for a CNES-operated satellite. Initial operational tests revealed that too many false alarms were raised and the detection results were not human-interpretable without data post-processing.

ATHMoS The Automated Telemetry Health Monitoring System (ATHMoS), located at the German Space Operations Center (GSOC), consists of modules of anomaly detection algorithms that analyze and provide insight on spacecraft telemetry. One such module is Project Sibyl (Verzola et al., 2016), which also employs LoOP to detect outliers in windows of data. A difference between this system and the one described by Martínez-Heras (2012) is an additional preprocessing step consisting in Density Based Spatial Clustering in Applications with Noise (DBSCAN; Ester et al., 1996) that automates the selection of a nominal set of telemetry for input into the novelty detection component. When demonstrated on telemetry from the Columbus Orbital Laboratory on the ISS, this unsupervised preprocessing step achieved comparable results to hand-selection of nominal data by domain experts, while reducing much of the manual effort needed.

Another module, contributed by O'Meara et al. (2018), combines the existing outlier detection process in ATHMoS with deep learning. An autoencoder neural network extracts features from time intervals of telemetry, which are concatenated to a set of human-engineered statistical features. The resulting feature vector is used as input in Outlier Probability Via Intrinsic Dimension (OPVID), a novel density-based algorithm sharing similarities with LoOP. At the same time, the ability for the autoencoder to reconstruct its input data is an indication of the anomalousness of that data. Both the output from OPVID and reconstruction error are considered in detecting anomalies. The authors also describe an effort to forecast spacecraft behaviour by training Long Short-Term Memory networks (LSTMs; Hochreiter and Schmidhuber, 1997) on not only nominal data, but all available data. The outputs of the LSTMs are future telemetry values that are then input into the anomaly detection system to predict anomalies in the future.

Probabilistic Clustering and Dimensionality Reduction Yairi et al. (2017) decide to model real-valued continuous data separately from discrete variables. They leverage the dimensionality reduction and clustering capability of mixture of probabilistic principal com-

ponents analyzers (MPPCA; Tipping and Bishop, 1999) on continuous values, arguing that a finite number of clusters are generally formed in a low-dimensional subspace as a result of the different spacecraft operational modes. Meanwhile, discrete-valued data are handled by mixtures of categorical distributions. The combined model of MPPCA with categorical distributions is a joint probability distribution, the parameters of which are learned from nominal past telemetry. Once trained, the model predicts the degree of anomalousness for new data samples. The authors choose not to find a threshold separating anomalies from nominal observations, and instead inspect the cases that yield the largest anomaly scores. This system was tested on telemetry from the Small Demonstration Satellite 4 (SDS-4), a microsatellite mission operated by JAXA.

Event Detection with Space Environment Correlation A study by Carlton et al. (2018) uses statistical methods to detect potential faults and interesting events across multiple geostationary Earth orbit (GEO) satellites from telemetry provided by commercial satellite operators Intelsat and Inmarsat. They attempt to find transient events such as spikes in data using the Tukey method (David and Tukey, 1977; Wang et al., 2011) on windows of telemetry. At the same time, change point detection is performed using piecewise linear approximations (PLAs; Shatkay and Zdonik, 1996). The events detected through these algorithms are compared to space weather data and lists of known satellite anomalies near GEO to find correlations with space environmental factors.

Sparse Decomposition and Dictionary Learning Another technique (Pilastre, Boussouf, et al., 2020) out of CNES and TéSA is based on sparse representation theory (Bruckstein et al., 2009), the goal of which is to approximate a signal $\mathbf{y} \in \mathbb{R}^N$ as a linear combination of a dictionary $\mathbf{\Phi} \in \mathbb{R}^{N \times L}$ and a sparse coefficient vector $\mathbf{x} \in \mathbb{R}^L$. The columns of the dictionary, called *atoms*, are derived from nominal telemetry such that nominal samples are well approximated as $\mathbf{y} \approx \mathbf{\Phi x}$ while maintaining the sparsity of $\mathbf{x}$. In the anomaly detection setting, $\mathbf{y}$ is a mixed signal consisting of discrete and continuous telemetry from a window of time and is decomposed as $\mathbf{y} = \mathbf{\Phi x} + \mathbf{e} + \mathbf{b}$, where $\mathbf{\Phi x}$ represents nominal data, $\mathbf{e}$ is an additive error term, and $\mathbf{b}$ is noise. Thus the decomposition residuals $\mathbf{e}$ and $\mathbf{b}$ provide an estimate of the anomalousness of the input signal. This Anomaly Detection using DICTionary (ADDICT) method proved to be comparable in performance to those introduced by Yairi et al. (2017) and Fuertes et al. (2016) in experiments on a reduced

spacecraft dataset. Later iterations of ADDICT incorporating convolutional sparse representations (Pilastre, Silva, et al., 2020) and weights for each telemetry channel (Pilastre et al., 2021) showed noticeable improvements on the same reduced dataset.

LSTM Forecasting with Dynamic Thresholding Hundman et al. (2018) from NASA's Jet Propulsion Laboratory (JPL) also make use of LSTMs to forecast telemetry; past nominal data is combined with commands executed on the spacecraft to output predictions of future values of a single channel. Larger errors between predicted values and actual incoming telemetry signify more anomalous time steps. Their method, called *Telemanom*, also dynamically finds thresholds for windows of error scores to separate nominal from anomalous values. An anomaly pruning technique is applied afterwards to reduce the number of false positive results. This system was deployed for operations for the Soil Moisture Active Passive (SMAP) mission in October of 2017 and was monitoring over 700 telemetry channels in near real-time at that time. We provide a more detailed description of the Telemanom method later in Section 4.1.

A major contribution alongside Telemanom is the public release of a dataset of SMAP and Mars Science Laboratory (MSL) telemetry and command information. This anonymized dataset contains 105 anomaly sequences to be detected from almost 500,000 data points across 82 channels. While this dataset is commonly cited in time series anomaly detection works, it contains shortcomings that often are not addressed. In their official code repository[2], the authors acknowledge that for certain channels, the training and test data are not scaled consistently, which means nominal test data no longer resembles the train set. Additionally, the dataset cannot be used to learn from multiple channels at once in multivariate models since the channel values come from different, independent time windows.

Convolutional LSTM with MPPCA Building upon the works of Yairi et al. (2017) and Hundman et al. (2018), Tariq et al. (2019) develop a method that combines Convolutional LSTM (ConvLSTM) neural networks with MPPCA into multivariate models. These models handle the telemetry channels of entire subsystems of spacecraft at once, relying on the success shown by ConvLSTMs in predicting multivariate outputs (Zhang et al., 2018). Dynamic thresholding and error smoothing, borrowed from the Telemanom method, are ap-

[2] https://github.com/khundman/telemanom

plied to produce intermediate anomaly scores. These are combined with anomaly probabilities calculated via MPPCA to generate final anomaly scores. For each channel, the lower-dimensional representations produced during MPPCA are reconstructed to their original data samples and the error in reconstruction is used to reveal possible channels responsible for the detected anomalies. The authors run their method on ten months of telemetry gathered from the Korea Multi-Purpose Satellite 2 (KOMPSAT-2), a satellite operated by the Korea Aerospace Research Institute (KARI).

Breakpoint Detection and Dependency Graph Analysis Boumghar et al. (2021) propose an anomaly detection approach that is supplemented by a diagnosis method for finding the cause of an anomalous event. They split multivariate telemetry into segments of similar behaviour based on the time series segmentation work of Lee et al. (2018). This is done with an autoencoder neural network that capture key features of input data in a latent space; the L2 norm of the latent representation of consecutive windows of telemetry is calculated and local maximums of the distance measures are considered as breakpoints. Not all breakpoints indicate anomalies. To further investigate these changes in behaviour, dependency graphs characterizing the telemetry before and after the breakpoint are built by applying eXtreme Gradient Boosting (XGBoost; Chen et al., 2015) and extracting the importance scores of each feature. Changes in the structures of the graphs on either side of the breakpoint can reveal clues in determining the abnormality of behaviour near that point.

Deep Generative Models An investigation performed by Ahn et al. (2020) looks at two deep generative models for anomaly detection: variational autoencoder (VAE; Kingma and Welling, 2014; D. J. Rezende et al., 2014) and GANomaly (Akcay et al., 2018). To apply such models on time series spacecraft data, the usual 2-D convolution operations are replaced with their 1-D variants. These semi-supervised anomaly detection methods are additionally augmented through Bayesian Optimization with a Gaussian process (Snoek et al., 2012) for improved hyperparameter selection. Anomalies are detected based on the generation errors of incoming input data after training on nominal data. The authors evaluate their proposed techniques on simulated sensor data and faults from an attitude and orbital control simulator developed by KARI to test their upcoming Korea Pathfinder Lunar Orbiter (KPLO) mission (Jung et al., 2019).

2.3 Deep Anomaly Detection for Time Series

In addition to the methods previously mentioned, we highlight a few different anomaly detection techniques intended for general time series signals, since they can be applied to spacecraft datasets.

Generative Adversarial Networks While generative adversarial networks (GANs; Goodfellow et al., 2014) have been wildly popular in image processing tasks, much less focus has been given to time series generation using such networks. However, their ability to generate realistic sequential data samples is leveraged by Li et al. (2019) in detecting anomalies. Their method, Multivariate Anomaly Detection with GAN (MAD-GAN) involves training an LSTM-based generator and discriminator to model the complex correlations within nominal multivariate time series data. The generator attempts to map samples from a latent space to data that follows the nominal distribution. If latent space samples corresponding to new test data are poorly reconstructed by the generator, this indicates that the test data do not follow the nominal distribution. Meanwhile, the discriminator learns to distinguish nominal samples from off-nominal ones. The reconstruction residual and discriminator output, together, form an anomaly score. Geiger et al. (2020) build upon this idea with their TadGAN framework, which incorporates techniques known to improve the stability of GAN training. We explore TadGAN in more depth in Section 5.1.

Stochastic Recurrent Neural Networks First introduced by Bayer and Osendorfer (2015), stochastic recurrent networks leverage variational inference techniques in recurrent neural networks (RNNs) to model underlying distributions of sequential data. Park et al. (2017) describe one such network, an LSTM-based variational autoencoder (LSTM-VAE), for multimodal anomaly detection in time series data. LSTM-VAE outputs an anomaly score based on the negative log-likelihood of a test sample with respect to its reconstructed distribution. As with other reconstruction-based methods in semi-supervised settings, anomalous data are indicated by higher reconstruction scores than their nominal counterparts. Following (Park et al., 2016), a dynamically varying threshold is learned from the latent space representation of nominal observations.

Su et al. (2019) propose another stochastic recurrent network which combines VAE

with gated recurrent unit (GRU; Cho, van Merrienboer, Bahdanau, et al., 2014) layers into a method called *OmniAnomaly*. In this method, not only are temporal dependencies captured in input space with the GRU, but also among latent space variables with a linear Gaussian state space model (Kitagawa and Gersch, 1996). Furthermore, planar normalizing flows (D. Rezende and Mohamed, 2015) is applied to approximate the stochastic latent space variable in order to learn a non-Gaussian posterior distribution. For determining a threshold, OmniAnomaly borrows the Peaks-Over-Threshold (POT) technique introduced in (Siffer et al., 2017), which we explain further in Section 6.1.5.

Deep Autoencoding Gaussian Mixture Model The Deep Autoencoding Gaussian Mixture Model (DAGMM; Zong et al., 2018) is an unsupervised method that consists of two networks. The first is an autoencoder that compresses high-dimensional input data $\mathbf{x}$ to a lower-dimensional space and then reconstructs it to resemble $\mathbf{x}$. Both the latent space representation and a set of features derived from the reconstruction error are fed into an estimation network. This second network facilitates learning a Gaussian Mixture Model (GMM) by predicting mixture membership of each sample, which are used to directly estimate the parameters of the GMM. By avoiding the typical alternating Expectation-Maximization (EM) algorithm for learning a GMM (Huber, 2011), joint optimization of dimensionality reduction and density estimation can be performed in an end-to-end training framework. Then the estimated likelihood of new test samples is used to predict anomalies based on a preselected threshold.

Graph Attention Akoglu et al. (2014) argue that graphs can be more robust in detecting anomalies by looking at the structure and relationships of data rather than just the raw values. Although telemetry is not inherently structured as graphs, Deng and Hooi (2021) manage to leverage graph techniques in a time series anomaly detection setting by modeling associations between sensors as graphs. In this work, embedding vectors are generated for each telemetry channel, which represent nodes in the graph. By computing the normalized dot product of each pair of nodes as measures of similarity, edges between nodes are learned and an adjacency matrix is formed. The learned structure is used to forecast the expected behaviour of each sensor with the help of a graph attention mechanism. Then the forecasted values are compared with actual observed data to yield anomaly scores at each time step.

Another method that makes use of graph attention for detecting anomalies in time series data is proposed by Zhao et al. (2020). They model not only the relationships between features of a multivariate dataset, but also the dependencies across time steps, as graphs. This work is expanded upon in Section 6.1.

Finally, a number of surveys and review papers have been published to help the interested reader navigate the state of anomaly detection research, including on time series data. Chandola et al. (2009) provide a structured and comprehensive overview of different classical techniques in multiple application domains. They identify strengths and weaknesses of each technique, but do not cover deep learning methods of recent years. Pimentel et al. (2014) mention more works in their review by considering a wider scope of methods related to anomaly detection, novelty detection, or outlier detection. Chalapathy and Chawla (2019) focus on deep anomaly detection, citing methods for multivariate time series anomaly detection that employ variations of RNNs, autoencoders, and GANs. Pang et al. (2020) extend this work by identifying main themes or challenges in anomaly detection and how deep learning techniques tackle them. Blázquez-García et al. (2021) present a taxonomy of outlier detection techniques for time series data, distinguishing between methods for univariate versus multivariate input data as well as for point versus collective outliers. Ruff et al. (2021) thoroughly review traditional shallow and newer deep learning approaches, plus provide an outlook on research opportunities that lie ahead. The reader is encouraged to peruse the survey papers mentioned here for a more extensive list of applicable methods.

Having provided an overview of anomaly detection methods relevant to spacecraft telemetry, the next step is to apply some of these methods and evaluate their performance on a real dataset. The upcoming chapter describes the setup of the experiments we will be conducting.

3

Experimental Setup

This chapter describes the framework of our experiments. Since they all follow the same formula, we first cover the elements common to each experiment before diving into specific anomaly detection methods in detail. We explore the real-world (or rather, real out-of-this-world) dataset as well as the preprocessing techniques applied before data is fed to deep learning models. We list the models and thresholding techniques we are interested in, plus the metrics to consider when evaluating such methods for anomaly detection.

3.1 Dataset and Data Preparation

Our evaluation of deep anomaly detection methods is conducted on Near-Earth Object Surveillance Satellite (NEOSSat) data provided by the Canadian Space Agency (CSA). NEOSSat is a space telescope that searches for and tracks asteroids in the inner solar system as well as objects orbiting Earth (Abbasi et al., 2019). Unlike ground-based telescopes, it is not limited by the day-night cycle, geographic location, or atmospheric weather, allowing stable imaging of multiple science targets on a daily basis. The dataset we are given is currently not available to the public; we describe its characteristics which are required to understand the experiments and discussions that follow, but leave out particularities which might be deemed sensitive.

3.1.1 NEOSSat Telemetry

NEOSSat telemetry has been collected and stored since its launch in 2013. The definition of nominal state for this spacecraft has transformed during its lifetime, notably due to two separate hardware failures and the resulting recovery actions (Abbasi et al., 2019). We choose to use recent data which is a closer representation of the present state of the spacecraft. We also wish to obtain sufficient anomalous samples to test the methods, considering that anomalies are relatively infrequent. Thus, approximately one year of housekeeping data from April 2018 to March 2019 is selected as our total dataset. Since the month of May happens to be free of reported anomalies, it is used as the train set. April is set aside as a validation set, and the remaining ten months comprise the test sets used for evaluation.

Telemetry from NEOSSat is organized into groups called *packets* based on similar components or types of channels. The number of channels in each packet ranges from approximately 16 to 61. Out of the 122 types of packets downlinked to the ground station, we use a smaller subset of 11 packets (summarized in Table 2.1) which allows us to run multiple experiments covering different anomaly detection methods. These packets are chosen because they each contain at least one channel that has been identified by satellite operators as an anomaly signature among anomalies that occurred between April 2018 and March 2019. Selecting channels and packets in this manner allows us to evaluate whether anomaly detection methods can identify anomalies from this time period when provided the same telemetry signals used by spacecraft engineers to spot them. Another benefit is that entire telemetry packets can easily be used as inputs for models that work with multivariate data, since we already know that the channels in each packet are related. In total we analyze 344 telemetry channels.

Preprocessing

Before the data is fed into our models, we apply typical preprocessing methods to standardize the different channels. Since telemetry channels are sampled at different rates, we aggregate each signal into one minute time steps by taking the average of each minute. We then scale each channel between $[-1, 1]$ based on the minimum and maximum values of the train set. After scaling, values in the test set less than -10 or greater than 10 are clipped to

these limits respectively. Since we do not know if values with extraordinarily large magnitudes are due to anomalies or sensor noise, we clip these values instead of scrubbing them so they can still be fed into anomaly detection methods. In a similar vein, missing data could be associated with known anomalies, but can also arise due to mishaps in downlinking or storing data during a satellite pass. We elect to perform simple imputation of missing values with zeros. Realistically, since each data point is timestamped, missing telemetry is trivially detected by machines without the use of sophisticated algorithms. Therefore, we also discard any anomaly prediction overlapping a window of missing data unless it is a known anomaly. This ensures that a method's performance is based on its ability to detect anomalies from available telemetry, rather than its ability to detect missing telemetry.

3.1.2 Commands

Included with this dataset are the commands executed onboard the spacecraft. Their format follows the dataset introduced in Hundman et al. (2018)'s work: one-hot encoded commands indicating if that command was executed at a particular time step. Our dataset contains 25 commands, mainly consisting of pointing and imaging operations.

3.1.3 Labels

NEOSSat anomalies are identified and documented in anomaly reports by CSA satellite operators. These reports contain the time range of the anomalies along with their signatures. Since there are so many channels for operators to analyze, often only a handful of signatures are captured. In some cases, the signatures are associated with spacecraft event logs which appear solely during error states. We choose to discard anomaly labels such as these, whose signature channels are not available at all during the training period. We are left with 56 anomalies between April 2018 and March 2019 in our dataset.

3.2 Methods of Interest

Our anomaly detection methods of interest all follow a similar formula: the first step involves training a deep neural network to output anomaly scores for every time step of input data. The model either forecasts future telemetry values or reconstructs them from

a reduced latent space. Since it learns exclusively from nominal data, it fails to accurately predict (i.e., forecast or reconstruct) telemetry when the input is off-nominal. Thus the anomaly scores are constructed from the errors in predictions. Then a threshold is calculated such that scores which exceed said threshold are classified as anomalous time steps. The predicted anomalies are compared with a set of ground truth labels to assess the performance of each method.

From the various anomaly detection methods mentioned in Sections 2.2 and 2.3, we select three to evaluate in-depth on our spacecraft dataset:

- **Telemanom** (Hundman et al., 2018) forecasts telemetry using LSTMs, and introduces an original dynamic thresholding technique.
- **TadGAN** (Geiger et al., 2020) employs GANs and applies a simple fixed threshold.
- **MTAD-GAT** (Zhao et al., 2020) makes use of graph attention layers and leverages Peaks-Over-Threshold (POT; Siffer et al., 2017) to determine thresholds.

These distinct methods, which are covered in Chapters 4, 5, and 6 respectively, have all claimed successful results on a dataset of spacecraft telemetry. As a bonus, open source official or community implementations of these methods are available publicly, which helps resolve some uncertainties arising from missing details in their papers.

3.3 Evaluation Metrics

Our primary evaluation metrics are the commonly used precision, recall, and F1 score. We compare sequences of predicted anomalies outputted by the anomaly detection methods to sequences of known anomalies from the entire anomaly set and define the following outcomes based on the methodology of (Geiger et al., 2020; Hundman et al., 2018):

- A *true positive* (TP) occurs when a known anomaly sequence overlaps any predicted sequences.
- A *false negative* (FN) occurs when a known anomaly sequence does not overlap any predicted sequences.

- A *false positive* (FP) occurs when a predicted anomaly sequence does not overlap any known sequences.

In our experiments, anomaly sequences are predicted separately for each channel. The precision of a method is computed from the total number of TP and FP results across all channels:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

The way we calculate recall requires more elaboration. Since anomaly signatures are exhibited by only a subset of channels, we do not wish to tally up FNs based on the predicted anomaly sequences of each individual channel. Instead, we consider the sequences of all channels so that a FN is counted when a known anomaly sequence does not overlap a predicted sequence of *any* channel. If there is an overlap, a TP is counted in the recall equation:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Achieving high recall scores is thus not a terribly demanding task given a large number of channels to work with. The F1 score, also stylized as F_1, shows us the balance between the precision and recall metrics as a single score in the range of $[0, 1]$:

$$\text{F1} = 2\frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

Some works calculate these metrics based on the outcome of each time step rather than overlapping sequences. We choose not to proceed this way because then the recall calculation would involve checking if a time step is predicted as anomalous by any channel (similar to a *logical OR* operation across all channels), which can evolve into all time steps being predicted as anomalous when the number of channels is large. This scenario is avoided by evaluating overlapping sequences instead, but sanity checks are needed to ensure that high precision and recall are not achieved simply by predicting one very lengthy sequence as anomalous. We do this by restricting the maximum length of a predicted anomaly to five days, since the longest true anomaly in our dataset lasts four and a half days.

Besides the metrics mentioned above, there are a few desirable characteristics we are looking for when we examine anomaly detection methods. It is just as important to be

able to determine possible root causes of an anomaly as it is to detect one. Identifying the channels responsible for a predicted anomalous sequence gives us hints to finding the cause. These anomaly signatures can then be leveraged to detect reoccurring anomalies and establish recovery procedures. Most of the methods we work with are interpretable in this way, outputting anomaly scores for each channel individually. However, we want to make sure the anomaly scores reflect meaningful statistics, such as the probability of being a true anomaly or its severity. Having reliable anomaly scores also allows us to sort predicted anomalies so that satellite operators can focus their investigation efforts on higher-scoring ones.

Since the methods we experiment with involve calculating prediction errors, either based on forecasts or reconstruction of telemetry, we are interested in whether correlations can be found between these errors and performance scores. We look at the mean absolute errors of predictions across channels in each test month, excluding anomalous time steps and their surrounding values. This ensures that the average error score we come up with represents the error in predicting only nominal time steps, which is what the models are, for the most part, trained to do.

The computational cost of each algorithm is addressed briefly in Appendix A.1 through a comparison of running times on the same hardware. Although we are analyzing spacecraft data, our intended use case for an anomaly detection method is, for the time being, solely on the ground segment. Future deployment directly onboard a spacecraft would require extensive experiments with space-grade processors, which is out of the scope of our work. As long as the running times for training and inference seem reasonable for real-world applications, we make no further analysis concerning the complexity. A description of the hardware and software resources we make use of is also provided in Appendix A.2.

3.4 Reproducibility

Due to the size of the dataset, the number of model variations we study, and the different method parameters to be tuned, we elect to perform only a single run of each of our experiments. Sources of stochasticity in our deep neural networks include random weight initialization, shuffling of training data, and dropout layers (Zhuang et al., 2021). We ac-

knowledge that multiple trials and cross-validation techniques can produce a more accurate indication of a model's performance, but the enormous running times encountered during training and evaluation prevent us from taking these approaches. Still, in an effort to address randomness in our results, we split our ten months of test data into ten separate test sets rather than treating them as a single dataset. This way we can report average F1 scores across multiple test sets for a better comparison of overall method performance.

With the common experimental setup out of the way, the next three chapters explore our methods of interest in full detail and present the results of each experiment.

4

Telemetry Forecasting with Long Short-Term Memory Networks

The use of Long Short-Term Memory networks (LSTMs) to forecast data by learning from nominal behaviour has been explored in previous anomaly detection works (Bontemps et al., 2016; Chauhan and Vig, 2015; Malhotra et al., 2015). Hundman et al. (2018), affiliated with NASA's Jet Propulsion Laboratory (JPL), are the first to target spacecraft anomalies from telemetry and commands. They additionally contribute a dynamic thresholding approach to classify anomalies from prediction residuals. We commence our experiments with their proposed method.

4.1 Technical Background

4.1.1 Recurrent Neural Network

LSTMs belong to a family of neural networks called recurrent neural networks (RNNs; Rumelhart et al., 1986b), which are used primarily for sequential data such as time series signals, written language, and audio. The key feature that distinguishes RNNs from regular feedforward networks is that the weighted connections between nodes in RNNs contain cycles, allowing a variable's historical values to influence future outputs (Graves, 2012). This is akin to storing memory in a network's internal state. RNNs also have the ability to learn from sequences of variable length, in contrast to a feedforward network which learns

parameters assigned to particular input features from fixed-size data.

Calculating gradients in an RNN can be done via algorithms like real time recurrent learning (RTRL; Robinson and Fallside, 1987) or backpropagation through time (BPTT; Werbos, 1990), both of which are extensions of the standard backpropagation algorithm (Rumelhart et al., 1986b) to sequence models. In practice, RNNs are unable to learn long-term dependencies and can only connect future predictions with information from the recent past. This is because RNNs trained with BPTT on long sequential data are particularly susceptible to *vanishing gradients*: as error signals are backpropagated across many time steps, their gradients decay to zero (Bengio et al., 1994; Hochreiter, 1991). This limits the learning as long-term dependencies are dominated by short-term dependencies in the calculation of the gradient (Hochreiter et al., 2001). The related problem of exploding gradients (Hochreiter, 1991) can also occur when the backpropagated error signals blow up exponentially in the number of layers of the network; again, this renders RNNs ineffective as it makes training highly unstable.

4.1.2 Long Short-Term Memory

LSTMs were introduced in 1997 (Hochreiter and Schmidhuber, 1997) in an attempt to solve the problems relating to long-term dependencies in vanilla RNNs. Their architecture includes different gates that control the flow of gradients through their memory units which are called *cells*. When the gates are closed, gradients pass through a cell unchanged, alleviating the vanishing gradients problem. Given the input $\mathbf{x}_t$ at the current time step t, the vector of previous hidden states $\mathbf{h}_{t-1}$, and the previous cell state c_{t-1}, the series of operations of each unit is described below:

$$f_t = \sigma\left(W_f \cdot \mathbf{x}_t + U_f \cdot \mathbf{h}_{t-1}\right), \tag{4.1a}$$

$$i_t = \sigma\left(W_i \cdot \mathbf{x}_t + U_i \cdot \mathbf{h}_{t-1}\right), \tag{4.1b}$$

$$\tilde{c}_t = \tanh(W_c \cdot \mathbf{x}_t + U_c \cdot \mathbf{h}_{t-1}), \tag{4.1c}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t, \tag{4.1d}$$

$$o_t = \sigma\left(W_o \cdot \mathbf{x}_t + U_o \cdot \mathbf{h}_{t-1}\right), \tag{4.1e}$$

$$h_t = o_t * \tanh\left(c_t\right), \tag{4.1f}$$

where W_l and U_l are, respectively, input and recurrent weights of the corresponding layer l. Bias terms are omitted for brevity. The operations are also shown in Figure 4.1.

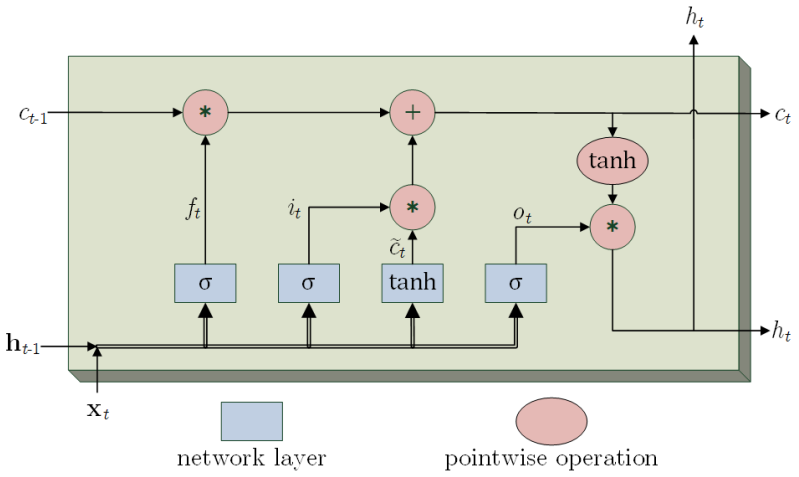

Figure 4.1: Operations involved in a single LSTM cell, given input data $\mathbf{x}_t$, the previous hidden states $\mathbf{h}_{t-1}$, and the previous cell state c_{t-1}.

The first type of gate, known as the *forget gate*, determines how much of the previous cell state is kept versus discarded (Equation 4.1a). The sigmoid function σ restricts the output f_t between 0, where the previous cell state is discarded completely, and 1, where all the previous cell state is kept. The second type is the *input gate*. The output i_t of this gate decides which information from the input $\mathbf{x}_t$ to use to update the cell state (Equation 4.1b). A hyperbolic tangent activation restricts the input values between -1 and 1 to produce candidate values $\tilde{c}_t$ with which to update the cell state (Equation 4.1c). Given the previous cell state c_{t-1}, the new cell state c_t is calculated from the results of the forget and input gates (Equation 4.1d). The final *output gate* determines how to produce the new hidden state h_t from the new cell state (Equations 4.1e-4.1f).

The addition of these gates to the recurrent network architecture has enabled LSTMs to show success in sequence-to-sequence learning tasks in areas such as natural language processing (NLP; Young et al., 2018; P. Zhou et al., 2016), speech recognition (Graves et al., 2013; Sak et al., 2014), and time series forecasting on domain-specific datasets (Cao et al., 2019; Chimmula and Zhang, 2020; Sagheer and Kotb, 2019).

4.2 Method Details

4.2.1 LSTM-based Forecasting

The method proposed by Hundman et al. (2018), called Telemanom, uses LSTMs to forecast telemetry data from past sequences of spacecraft telemetry and commands. The input consists of a time series sequence $X = \{\mathbf{x}^{t-l_s+1}, \mathbf{x}^{t-l_s+2}, \ldots, \mathbf{x}^t\}$ of length l_s for some time step t. Each $\mathbf{x}^t = [x_m^t, \mathbf{c}^t]$ contains the value x_m^t of telemetry channel m concatenated to command information $\mathbf{c}^t \in \mathbb{R}^n$ which are one-hot encoded values indicating whether the n commands were executed at time t. The output $\hat{y}_m^{t+1}$ is a prediction of the value of m at the next time step, and the loss function is simply the mean squared error (MSE) in forecasting,

$$L = \frac{1}{T}\sum_{t=1}^{T}(\hat{y}_m^t - x_m^t)^2. \tag{4.2}$$

Thus there is one model for each telemetry channel. Hundman et al. (2018) justify the univariate predictions by arguing that LSTMS struggle to predict high-dimensional outputs and that this allows the source of detected anomalies to be easily traced to individual channels.

4.2.2 Dynamic Thresholding and Mitigating False Positives

Hundman et al. (2018) propose a series of steps to find anomalous samples from the model predictions of future telemetry values. First an error score is generated at each time step as $e^t = |\hat{y}_m^t - x_m^t|$. These error scores are split into sliding windows $\mathbf{e} = [e^{t-l_e+1}, \ldots, e^{t-1}, e^t]$ of length w_e with step size w_s, and then smoothed using an exponentially-weighted moving average (EWMA; Hunter, 1986). As part of a spacecraft's nominal behaviour, telemetry values can change abruptly due to the sudden execution of time-tagged commands; such changes can be difficult to predict using LSTM-based models (Shipmon et al., 2017). This results in sharp spikes in error scores during nominal periods which need to be smoothed out. A sample of the actual and forecasted telemetry values for channel *DBG_SCI_LAST_CCD_TEMP*, along with the effect of smoothing the forecasting errors, is seen in Figure 4.2.

4.2 Method Details

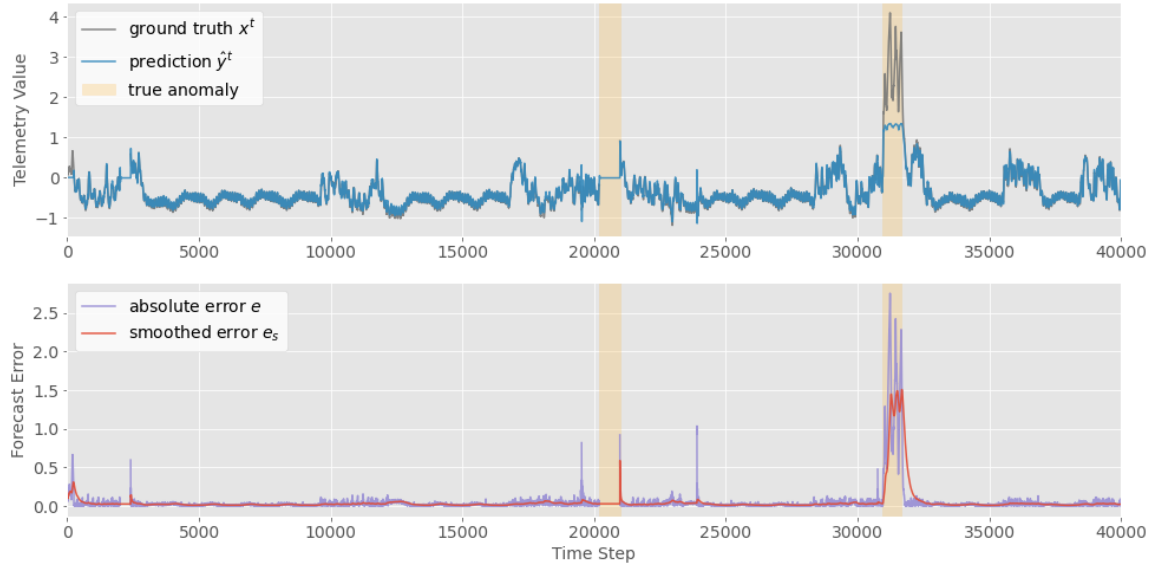

Figure 4.2: True and predicted telemetry (top), as well as error values (bottom) from channel *DBG_SCI_LAST_CCD_TEMP*.

Once the smoothed errors $\mathbf{e}_s$ are obtained, an anomaly threshold is dynamically found for each window. Candidate thresholds are given by the set $\boldsymbol{\epsilon} = \mu(\mathbf{e}_s) + \mathbf{z}\sigma(\mathbf{e}_s)$ where $\mathbf{z} = \{2.5, 3, 3.5, \ldots, 12\}$ represents the number of standard deviations $\sigma(\mathbf{e}_s)$ above the mean $\mu(\mathbf{e}_s)$. Then the threshold ε^* is determined from the candidate thresholds as

$$\varepsilon^* = \arg\max_{\varepsilon \in \boldsymbol{\epsilon}} \frac{\Delta\mu(\mathbf{e}_s) + \Delta\sigma(\mathbf{e}_s)}{|\mathbf{e}_a| + |\mathbf{E}_{seq}|^2}$$

where $\Delta\mu(\mathbf{e}_s)$ and $\Delta\sigma(\mathbf{e}_s)$ are, respectively, the percent change in mean and standard deviation of $\mathbf{e}_s$) when the values above ε are removed:

$$\Delta\mu(\mathbf{e}_s) = \frac{\mu(\mathbf{e}_s) - \mu(\{e_s \in \mathbf{e}_s | e_s < \varepsilon\})}{\mu(\mathbf{e}_s)},$$

$$\Delta\sigma(\mathbf{e}_s) = \frac{\sigma(\mathbf{e}_s) - \sigma(\{e_s \in \mathbf{e}_s | e_s < \varepsilon\})}{\sigma(\mathbf{e}_s)},$$

and $\mathbf{e}_a = \{e_s \in \mathbf{e}_s | e_s > \varepsilon\}$. Finally, $\mathbf{E}_{seq}$ is the set of continuous sequences of $e_a \in \mathbf{e}_a$. The goal is to find the threshold which produces the largest $\Delta\mu(\mathbf{e}_s)$ and $\Delta\sigma(\mathbf{e}_s)$ while

minimizing the quantity of detected anomalies $\mathbf{e}_a$ and anomalous sequences $\mathbf{E}_{seq}$ to prevent greedy behaviour. This method notably does not rely on labeled anomalous data, following the semi-supervised nature of Telemanom.

After the threshold is found for each error window, a pruning procedure is applied to reduce the number of false positive results by only retaining the most erroneous anomalies. First, the maximum error value in each error sequence is collected and sorted in descending order to form a new set $\mathbf{e}_{max} = sort(\{\max(\mathbf{e}_{seq}) \ \forall \ \mathbf{e}_{seq} \in \mathbf{E}_{seq}\})$. Then $\mathbf{e}_{max}$ is appended with the maximum smoothed error not exceeding the threshold, $\max(\{e_s \in \mathbf{e}_s | e_s \notin \mathbf{e}_a\})$. For each $e^i_{max} \in \mathbf{e}_{max}$, at index i, the percent decrease d^i between e^i_{max} and the next highest error e^{i+1}_{max} is calculated. If any such decrease as well as all subsequent decreases are less than a minimum percent p, the anomalies corresponding to those error sequences are reclassified as nominal. This pruning approach limits the number of flagged anomalies that must be verified by a human operator.

Finally, the remaining error sequences after pruning are assigned normalized anomaly scores based on their distance from the chosen threshold:

$$\tilde{s}^i = \frac{\max(\mathbf{e}^i_{seq}) - \varepsilon^*}{\mu(\mathbf{e}_s) + \sigma(\mathbf{e}_s)}. \tag{4.3}$$

4.3 Experiments

We perform experiments based on two variations of the Telemanom model:

- **LSTM-Cmd**—this is the same setup as the original Telemanom which includes spacecraft commands as covariates to influence the forecasted telemetry values;
- **LSTM-Solo**—we exclude command information from the input to observe changes in forecasting errors and ultimately in predicting anomalies.

Model Configuration

The model comprises two hidden LSTM layers, each with 80 units and dropout applied after. The output layer is a fully connected layer that produces the forecasted telemetry of

a single channel. Training occurs for 35 epochs or until the loss on the validation set no longer decreases during ten consecutive training iterations. We employ the Adam optimizer (Kingma and Ba, 2014) as a stochastic gradient descent method for minimizing our loss L with a learning rate of 10^{-3}.

Method Evaluation

Anomaly detection performance is evaluated across a range of values for the minimum percent parameter used in pruning: $p \in \{0.0, 0.1, \ldots, 0.9\}$, where $p = 0.0$ means no pruning. For computing thresholds, we use sliding windows with window sizes $w_e \in \{\frac{T}{4}, \frac{T}{3}, \frac{T}{2}\}$ and step size $w_s = 0.1 w_e$, where T is the size of the monthly test set. We compare dynamic thresholding against a fixed threshold of 4 standard deviations away from the mean of the current window of error scores, as used in (Geiger et al., 2020). A full description of the implementation as well as training and evaluation configurations is presented in Appendix A.3.

4.4 Results

The overall results of the two Telemanom experiments across all test months are presented in Table 4.1. Dynamic thresholding clearly achieves higher F1 scores than a fixed threshold, due to a significant improvement in recall.

	Fixed Threshold			Dynamic Threshold		
Experiment	Precision	Recall	F1	Precision	Recall	F1
LSTM-Cmd	0.56(0.30)	0.75(0.23)	**0.56**(0.29)	0.58(0.31)	0.89(0.17)	**0.64**(0.27)
LSTM-Solo	0.56(0.30)	0.71(0.20)	0.55(0.28)	0.55(0.28)	0.91(0.17)	0.63(0.31)

Table 4.1: Mean(standard deviation) of monthly results of Telemanom experiments with fixed and dynamic thresholds.

These overall results are achieved after exploring a range of values for w_e as well as p. Figure 4.3 shows the impact of pruning on both experiments after applying a dynamic threshold. It can be seen that, up to a certain level of pruning, the precision improves significantly while the recall is negatively effected only to a minor extent. Since the level at

which the highest overall F1 score is found seemingly varies between methods, we choose to report this highest score in our results here and going forward. This is discussed more in Section 7.1.2.

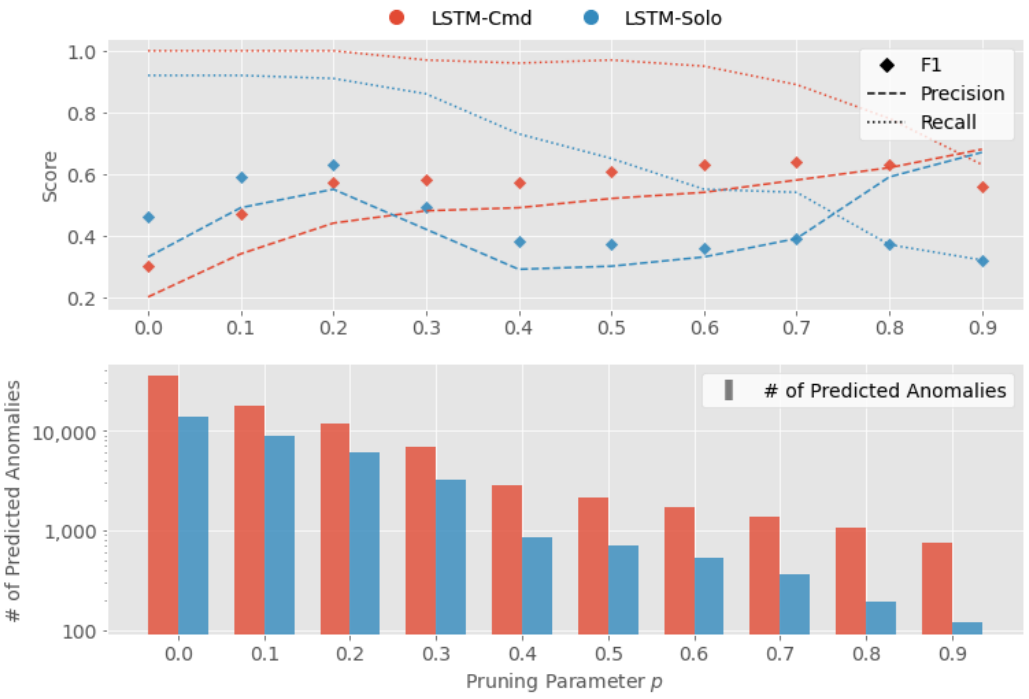

Figure 4.3: Impact of the anomaly pruning parameter p after dynamic thresholding on mean precision, recall, and F1 scores (top); and on the number of predicted anomalies across all channels (bottom).

The mean absolute error (MAE) in forecasted values is slightly increased when commands are included in input data (0.158) compared to no commands (0.136). This indicates that past command information may not provide any benefit in predicting future telemetry values in this setup. We would expect that a model which more accurately predicts nominal values would better detect off-nominal samples; however, there is so far no clear correlation between the error and F1 score, and a tiny increase in anomaly detection performance is achieved overall with commands as covariates.

4.5 Method Summary

LSTMs have been around for a long time, and using them for time series forecasting is nothing new. While the dynamic thresholding and false positive reduction techniques proposed by Hundman et al. (2018) are effective, we look to more recent technologies that can be applied to generate anomaly scores.

5

Time Series Anomaly Detection using Generative Adversarial Networks

Since generative adversarial networks (GANs) entered the spotlight of the deep learning stage in 2014 (Goodfellow et al., 2014), they have not left. Their success in image-related tasks such as image generation, video prediction, and even anomaly detection in images has led to works exploring the effectiveness of GANs on time series data. One product of this effort is a method called *TadGAN*, proposed by Geiger et al. (2020), which claims better anomaly detection performance over Telemanom on the SMAP and MSL datasets. We highlight their approach in this chapter.

5.1 Technical Background

This section covers the basic formulation of GANs and three considerable improvements developed since its inception.

5.1.1 Generative Adversarial Network

The vanilla GAN architecture involves two networks: a *generator* $\mathcal{G}$ and a *discriminator* $\mathcal{D}$. Given random noise vectors $\mathbf{z}$, the generator aims to learn the distribution of some data $\mathbf{x}$ to produce fake samples $\mathcal{G}(\mathbf{z})$ that match this distribution. The discriminator has the goal of differentiating between real and fake data, and outputs a probability that some input is a real

sample. During training, $\mathcal{D}$ tries to maximize this output on real data $\mathbf{x}$ while minimizing the output given fake data $\mathcal{G}(\mathbf{z})$. At the same time, $\mathcal{G}$ attempts to minimize the probability $1 - \mathcal{D}(\mathcal{G}(\mathbf{z}))$ that $\mathcal{D}$ recognizes the generated data as fake. Combining these objectives gives us the overall GAN loss function, described as a minimax game with a value function $V(\mathcal{G}, \mathcal{D})$:

$$\min_G \max_D V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{\mathbf{x}\sim p_r}[\log \mathcal{D}(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z})))] \tag{5.1}$$

$$= \mathbb{E}_{\mathbf{x}\sim p_r}[\log \mathcal{D}(\mathbf{x})] + \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[\log(1 - \mathcal{D}(\tilde{\mathbf{x}}))], \tag{5.2}$$

where p_r is the real data distribution over $\mathbf{x}$ and p_z is a prior distribution of the noise $\mathbf{z}$. By implicitly defining the model distribution of the generated samples $\tilde{\mathbf{x}} = \mathcal{G}(\mathbf{z})$ as p_g, we obtain Equation 5.2.

In practice, training occurs by alternating multiple iterations of updates to $\mathcal{D}$ with updates to $\mathcal{G}$, to avoid overfitting on finite datasets. Goodfellow et al. (2014) show that for a fixed $\mathcal{G}$, the optimal $\mathcal{D}$ is found at $\mathcal{D}^*(\mathbf{x}) = p_r(\mathbf{x})/(p_r(\mathbf{x}) + p_g(\mathbf{x}))$, at which point the value function can be reformulated as

$$\begin{aligned}
V(\mathcal{G}, \mathcal{D}^*) &= \mathbb{E}_{\mathbf{x}\sim p_r}[\log \mathcal{D}^*(\mathbf{x})] + \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[\log(1 - \mathcal{D}^*(\tilde{\mathbf{x}}))] \\
&= \mathbb{E}_{\mathbf{x}\sim p_r}\left[\log \frac{p_r(\mathbf{x})}{p_r(\mathbf{x}) + p_g(\mathbf{x})}\right] + \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}\left[\log \frac{p_g(\tilde{\mathbf{x}})}{p_r(\tilde{\mathbf{x}}) + p_g(\tilde{\mathbf{x}})}\right] \\
&= -\log(4) + \mathbb{E}_{\mathbf{x}\sim p_r}\left[\log \frac{p_r(\mathbf{x})}{(p_r(\mathbf{x}) + p_g(\mathbf{x}))/2}\right] \\
&\qquad + \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}\left[\log \frac{p_g(\tilde{\mathbf{x}})}{(p_r(\tilde{\mathbf{x}}) + p_g(\tilde{\mathbf{x}}))/2}\right] \\
&= -\log(4) + KL\left(p_r \middle|\middle| \frac{p_r + p_g}{2}\right) + KL\left(p_g \middle|\middle| \frac{p_r + p_g}{2}\right) \\
&= -\log(4) + 2 \cdot JS(p_r||p_g),
\end{aligned} \tag{5.3}$$

where KL and JS are the Kullback-Leibler and Jensen-Shannon divergences, respectively. Thus $V(\mathcal{G}, \mathcal{D}^*)$ represents the similarity of p_g and the data distribution p_r through the JS divergence (Equation 5.3). If the optimal discriminator is achieved at every round of updates to $\mathcal{D}$, then optimizing $\mathcal{G}$ leads p_g to converge to p_r.

5.1.2 Wasserstein-GAN

One of the major disadvantages in working with GANs is that the training process is susceptible to problems like instability, when the generator and discriminator oscillate rather than converge; and mode collapse, when the generator fails to output samples with sufficient diversity to model p_r. The convergence theory of GANs is an active area of research that has seen significant progress over the years. One idea from Arjovsky et al. (2017) is to replace the JS divergence with a loss based on the Wasserstein-1 distance, also known as Earth-Mover's (EM) distance. This distance is smooth and differentiable even for disjoint distributions, leading to improved stability in training. Through the Kantorovich-Rubinstein duality (Villani, 2009), the Wasserstein-1 distance is expressed as

$$W(p_r||p_g) = \sup_{||f||_L \leqslant 1} \mathbb{E}_{\mathbf{x}\sim p_r}[f(\mathbf{x})] - \mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[f(\tilde{\mathbf{x}})],$$

where the $\sup$ term represents the supremum over all 1-Lipschitz functions. If the function f, parameterized by w, comes from a family of K-Lipschitz functions for some K, the distance can be measured by

$$\max_{w\in\mathcal{W}} \mathbb{E}_{\mathbf{x}\sim p_r}[f_w(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[f_w(\mathcal{G}(\mathbf{z}))]. \tag{5.4}$$

In this new form of GAN, called *Wasserstein-GAN* (WGAN), the discriminator is renamed as *critic* and aims to learn an optimal f_w to be able to measure the Wasserstein distance between distributions p_r and p_g in the form of Equation 5.4. The generator then minimizes this distance to produce samples as close to the real distribution as possible.

5.1.3 Wasserstein-GAN with Gradient Penalty

For the function f_w to be K-Lipschitz continuous for some K, the critic weights w must lie in a compact space $\mathcal{W}$. Although the authors of WGAN suggest a simple trick of clipping w to constants $[-c, c]$ after each gradient update to enforce the Lipschitz constraint, they acknowledge the opportunity for better methods to be proposed by the research community. One such improvement, which does not rely on careful tuning of hyperparameter c, is Wasserstein-GAN with gradient penalty (WGAN-GP), proposed by Gulrajani et al. (2017).

They prove that the ideal critic function has gradient norm of 1 almost everywhere under distributions p_r and p_g. Based on this concept, weight clipping is replaced with a term that penalizes the model as the gradient norm moves away from 1. Adding the gradient penalty to the original critic loss from Equation 5.4 gives us the new critic loss

$$L_{\mathcal{C}} = \underbrace{\mathbb{E}_{\tilde{\mathbf{x}}\sim p_g}[\mathcal{C}(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x}\sim p_r}[\mathcal{C}(\mathbf{x})]}_{\text{original critic loss}} + \underbrace{\lambda \mathbb{E}_{\bar{\mathbf{x}}\sim p_{\bar{\mathbf{x}}}}[(\|\nabla_{\bar{\mathbf{x}}}\mathcal{C}(\bar{\mathbf{x}})\|_2 - 1)^2]}_{\text{gradient penalty}}, \tag{5.5}$$

where $\mathcal{C}$ denotes the critic and λ is a coefficient to weight the new penalty term. Due to the intractability of enforcing the unit gradient norm constraint everywhere, a soft version of the constraint is applied instead to random samples $\bar{\mathbf{x}} \sim p_{\bar{\mathbf{x}}}$, where $p_{\bar{\mathbf{x}}}$ is implicitly defined from uniformly sampling from straight lines between pairs of points in p_r and p_g.

5.1.4 Cycle Consistency Loss

In tasks where the goal is to generate data from a target domain Y that is different than that of the input domain X, there can be a large space of mapping functions $\mathcal{G} : X \rightarrow Y$ to search through. To reduce this search space, Zhu et al. (2017) introduce another mapping function $\mathcal{F} : Y \rightarrow X$ and argue that the mapping functions should be able to bring samples $\mathbf{x} \in X$ and $\mathbf{y} \in Y$ back to their original data in a cycle:

$$\mathbf{x} \rightarrow \mathcal{G}(\mathbf{x}) \rightarrow \mathcal{F}(\mathcal{G}(\mathbf{x})) \approx \mathbf{x},$$

$$\mathbf{y} \rightarrow \mathcal{F}(\mathbf{y}) \rightarrow \mathcal{G}(\mathcal{F}(\mathbf{y})) \approx \mathbf{y}.$$

To learn this behaviour, a cycle consistency loss L_{cyc} (T. Zhou et al., 2016) is included in the overall objective:

$$L_{cyc}(\mathcal{G}, \mathcal{F}) = \underbrace{\mathbb{E}_{\mathbf{x}\sim p_r(\mathbf{x})}[\|\mathcal{F}(\mathcal{G}(\mathbf{x})) - \mathbf{x}\|_1]}_{\text{forward cycle loss}} + \underbrace{\mathbb{E}_{\mathbf{y}\sim p_r(\mathbf{y})}[\|\mathcal{G}(\mathcal{F}(\mathbf{y})) - \mathbf{y}\|_1]}_{\text{backward cycle loss}}.$$

This method is referred to as *CycleGAN*.

5.2 Method Details

5.2.1 Adversarial Learning for Time Series Reconstruction

Along with forecasting-based anomaly detection, reconstruction-based models are some of the most common methods used in semi-supervised settings. The idea is to train a model, such as an autoencoder (Rumelhart et al., 1986a), to encode input data in a low-dimensional latent space and then map the latent data back into a reconstruction of the original input. An error score is calculated as the difference between the input and its reconstruction. Only nominal data is used during training so that when the model encounters off-nominal test samples, it is not able to accurately reconstruct the input, resulting in higher error scores.

TadGAN combines a time series reconstruction model with adversarial learning. For an input data domain X and a latent domain Z, the reconstruction model is made up of an LSTM-based encoder $\mathcal{E} : X \to Z$ and an LSTM-based generator $\mathcal{G} : Z \to X$. Given an input sample $\mathbf{x} \in X$, its reconstruction $\hat{\mathbf{x}}$ is expressed as

$$\mathbf{x} \to \mathcal{E}(\mathbf{x}) \to \mathcal{G}(\mathcal{E}(\mathbf{x})) = \hat{\mathbf{x}} \approx \mathbf{x}.$$

Additionally, there are two critics involved; $\mathcal{C}_x$ learns to distinguish between real and reconstructed data, while $\mathcal{C}_z$ differentiates between random latent samples $\mathbf{z} \sim p_{\mathbf{z}} = \mathcal{N}(0, 1)$ and encoded samples $\mathcal{E}(\mathbf{x})$. Figure 5.1 depicts this model architecture.

The overall objective of TadGAN borrows key elements from Wasserstein-GAN (Arjovsky et al., 2017), WGAN-GP (Gulrajani et al., 2017), and CycleGAN (Zhu et al., 2017):

$$\min_{\{\mathcal{E},\mathcal{G}\}} \max_{\{\mathcal{C}_x \in \mathbf{C_x}, \mathcal{C}_z \in \mathbf{C_z}\}} V_X(\mathcal{C}_x, \mathcal{G}) + V_Z(\mathcal{C}_z, \mathcal{E}) + V_{L2}(\mathcal{E}, \mathcal{G})$$

with

$$\begin{aligned}
V_X(\mathcal{C}_x, \mathcal{G}) &= \mathbb{E}_{\mathbf{x}\sim p_r}[\mathcal{C}_x(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_z}[\mathcal{C}_x(\mathcal{G}(\mathbf{z}))] + \lambda \mathbb{E}_{\bar{\mathbf{x}}\sim p_{\bar{\mathbf{x}}}}[(\|\nabla_{\bar{\mathbf{x}}} \mathcal{C}_x(\bar{\mathbf{x}})\|_2 - 1)^2], \\
V_Z(\mathcal{C}_z, \mathcal{E}) &= \mathbb{E}_{\mathbf{z}\sim p_z}[\mathcal{C}_z(\mathbf{z})] - \mathbb{E}_{\mathbf{x}\sim p_r}[\mathcal{C}_z(\mathcal{E}(\mathbf{x}))] + \lambda \mathbb{E}_{\bar{\mathbf{z}}\sim p_{\bar{\mathbf{z}}}}[(\|\nabla_{\bar{\mathbf{z}}} \mathcal{C}_z(\bar{\mathbf{z}})\|_2 - 1)^2], \\
V_{L2}(\mathcal{E}, \mathcal{G}) &= \gamma \mathbb{E}_{\mathbf{x}\sim p_r}[\|\mathbf{x} - \mathcal{G}(\mathcal{E}(\mathbf{x}))\|_2].
\end{aligned}$$

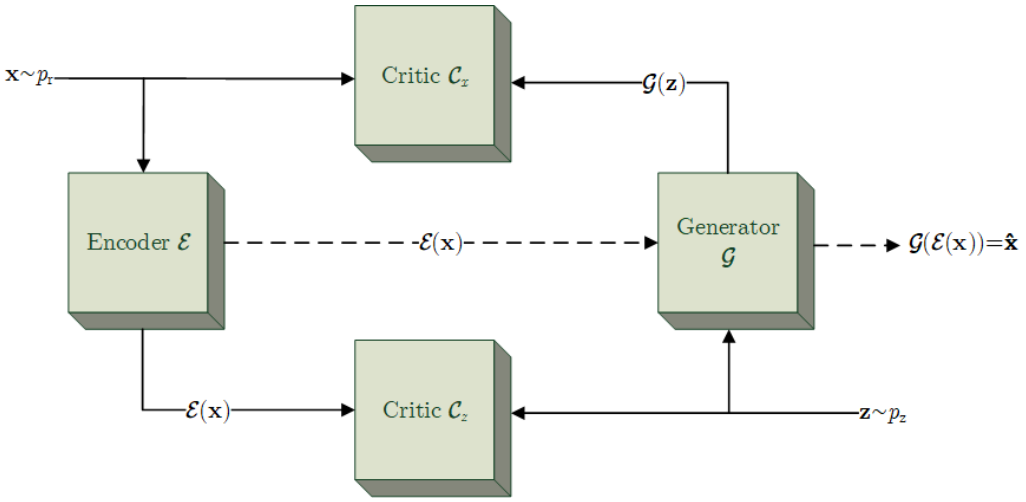

Figure 5.1: TadGAN model architecture.

Here, $\mathbf{C_x}$ and $\mathbf{C_z}$ are sets of 1-Lipschitz continuous functions. As before in Equation 5.5, $p_{\tilde{\mathbf{x}}}$ is obtained by uniformly sampling along straight lines between points in p_r and p_g. Likewise, $p_{\tilde{\mathbf{z}}}$ comes from uniformly sampling between points in p_z and p_e, where p_e is the implicitly defined model distribution of the encoded samples $\tilde{\mathbf{z}} = \mathcal{E}(\mathbf{x})$. The forward consistency loss is weighted by γ and calculated using the L2 norm instead of L1 to accentuate anomalous points; meanwhile the backward consistency loss is left out of TadGAN because it was not found to improve the performance of the method.

5.2.2 Combining GAN Outputs into Anomaly Scores

TadGAN leverages the scores produced by $\mathcal{C}_x$ in distinguishing between real and reconstructed samples as well as the reconstruction error to output anomaly scores for each time step. The authors experiment with three ways of computing this reconstruction error e^t. Let $\mathbf{x}^t = \{x^{t-l+1}, x^{t-l+2}, \ldots, x^{t+l}\}$ be the true time series input and $\hat{\mathbf{x}}^t = \{\hat{x}^{t-l+1}, \hat{x}^{t-l+2}, \ldots, \hat{x}^{t+l}\}$ be the reconstructed sequence at time t, both of length $L_s = 2l$. Then,

- *Point-wise difference* is the absolute difference at each time step: $e^t = |x^t - \hat{x}^t|$;
- *Area difference* is the average difference between the areas under two curves, and is effective at identifying regions where small differences persist over a long period of

time:

$$e^t = \frac{1}{L_s}\left|\int_{t-l+1}^{t+l} x^t - \hat{x}^t dx\right|;$$

- *Dynamic time warping* (DTW; Sakoe and Chiba, 1978; Vintsyuk, 1972) measures the similarity between local regions of two time sequences which can be out of step or of varying lengths (Berndt and Clifford, 1994). Given a matrix $\mathbf{W} \in \mathbb{R}^{L_s \times L_s}$ where each element $w_{i,j}$ is the Euclidean distance measure between x^i and $\hat{x}^j$, warp paths are defined as $\mathcal{W} = \{w^1, w^2, \ldots, w^K\}$ containing K distance measures. The warp paths are subject to constraints on continuity, such that $\{1, 2, \ldots, L_s\}$ is contained in both the set of indices i and in the set of indices j that are being measured by $w^k_{i,j}\ \forall\ w^k_{i,j} \in \mathcal{W}$; and on monotonicity, where the indices i and j of measure $w^k_{i,j}$ must be nondecreasing as k increases in $[1, K]$. The goal is to use the minimum distance between two time series as the reconstruction error, calculated as the minimum warp path among the set of valid warp paths $\boldsymbol{\mathcal{W}}$:

$$e^t = \min_{\mathcal{W} \in \boldsymbol{\mathcal{W}}} \left[\frac{1}{K}\sqrt{\sum_{k=1}^{K} w^k}\right].$$

To normalize and prepare the reconstruction error and critic scores to be combined, their respective z-scores Z_{RE} and Z_{C_x} are first calculated from their means and standard deviations. The anomaly score s^t is then produced either by taking their sum, weighted by α, as in Equation 5.6, or their product as in Equation 5.7:

$$s^t = \alpha Z^t_{RE} + (1-\alpha) Z^t_{C_x}, \tag{5.6}$$

$$s^t = Z^t_{RE} \cdot Z^t_{C_x}. \tag{5.7}$$

According to the ablation studies performed by Geiger et al. (2020), multiplication is the better option. The resulting anomaly scores at each time step are split into windows and a fixed threshold of 4 standard deviations away from the mean of the window is applied to identify anomalous time steps. Sequences of predicted anomalies are pruned and assigned a normalized anomaly score $\tilde{s}^t$ as is done in Telemanom.

5.3 Experiments

The experiments we perform in this chapter are with the following models:

- **GAN-Cmd**—we follow the TadGAN model with inputs as proposed in Telemanom, using a single telemetry channel and command information to make predictions on the same channel;
- **Recon-Cmd**—we remove the adversarial network to observe the performance of a reconstruction-only anomaly detection model and characterize the impact of adversarial learning;
- **GAN-Multi**—we extend the TadGAN model to learn from all of the $N > 1$ channels in a packet together and output multivariate predictions of these channels.

Model Configuration

The encoder $\mathcal{E}$ contains one bidirectional LSTM layer (Schuster and Paliwal, 1997) and a fully connected output layer. Generator $\mathcal{G}$ consists of a fully connected layer, a bidirectional LSTM layer with dropout, an upsampling step, another bidirectional LSTM with dropout, and finally a fully connected output layer with hyperbolic tangent activation. The architecture of Critic $\mathcal{C}_x$ includes four layers of 1-dimensional convolution[1] with leaky rectified linear unit (LeakyReLU; Maas et al., 2013) activation and dropout, followed by a fully connected output. The Critic $\mathcal{C}_z$ is made up of two fully connected layers, each with LeakyReLU and dropout, plus a final fully connected layer.

We run 2000 training loops with 5 critic iterations per loop using the Adam optimizer (Kingma and Ba, 2014). The input sequence is of length $L_s = 100$ and the dimensionality of the latent space is N*20, based on the number of channels N. We explore different combinations of encoder and generator hidden units for the multi-channel models since N varies for each packet, but stick to 100 and 64 units respectively for the single channel models. Evidently, the critic configurations do not apply to our reconstruction-only experiment. Full details of our TadGAN setup related to implementation and model parameters are found in Appendix A.4.

[1]The 1-dimensional convolution operation is elaborated in Section 6.2.1.

Method Evaluation

Following the experiments of Geiger et al. (2020), we evaluate different methods—point-wise difference, area difference, and DTW—for calculating reconstruction errors and then multiply them with the outputs of Critic $\mathcal{C}_x$ to produce anomaly scores. We also evaluate the anomaly detection performance when the critic score is excluded from the anomaly score, to observe its effect. Along with the fixed thresholding used in TadGAN, we also apply dynamic thresholding introduced in Telemanom (Hundman et al., 2018) on sliding windows across anomaly scores, prior to pruning.

5.4 Results

The overall results of the TadGAN experiments are presented in Table 5.1. Once again, there is a noticeable improvement in recall and overall F1 score when using a dynamic threshold over a fixed one. The difference in performance across the three experiments is minor, with GAN-Multi slightly ahead of the others.

	Fixed Threshold			Dynamic Threshold		
Experiment	Precision	Recall	F1	Precision	Recall	F1
GAN-Cmd	0.55(0.37)	0.66(0.27)	0.53(0.34)	0.60(0.36)	0.98(0.05)	0.66(0.37)
Recon-Cmd	0.52(0.34)	0.69(0.20)	0.52(0.33)	0.59(0.36)	0.88(0.15)	0.64(0.34)
GAN-Multi	0.60(0.40)	0.58(0.34)	**0.54**(0.35)	0.58(0.34)	1.00(0.00)	**0.67**(0.33)

Table 5.1: Mean(standard deviation) of monthly results of TadGAN experiments with fixed and dynamic thresholds.

According to the creators of TadGAN, the outputs of the critic networks are supposed to be lower for more anomalous samples, but taking a closer look at channels *PCDU_WHL_X_I* and *PCDU_WHL_Y_I* in Figure 5.2 reveals that this is not always the case. These channels have similar training and test data but the critic scores for these two signals appear like mirrored versions of each other. Fortunately, this method uses Z-scores instead of using raw critic scores directly, so any scores that deviate significantly from the mean can be interpreted as possible anomalous time steps. In spite of this workaround, it turns out the best results are achieved when only the reconstruction error is used, as seen in Table 5.2.

5.4 Results

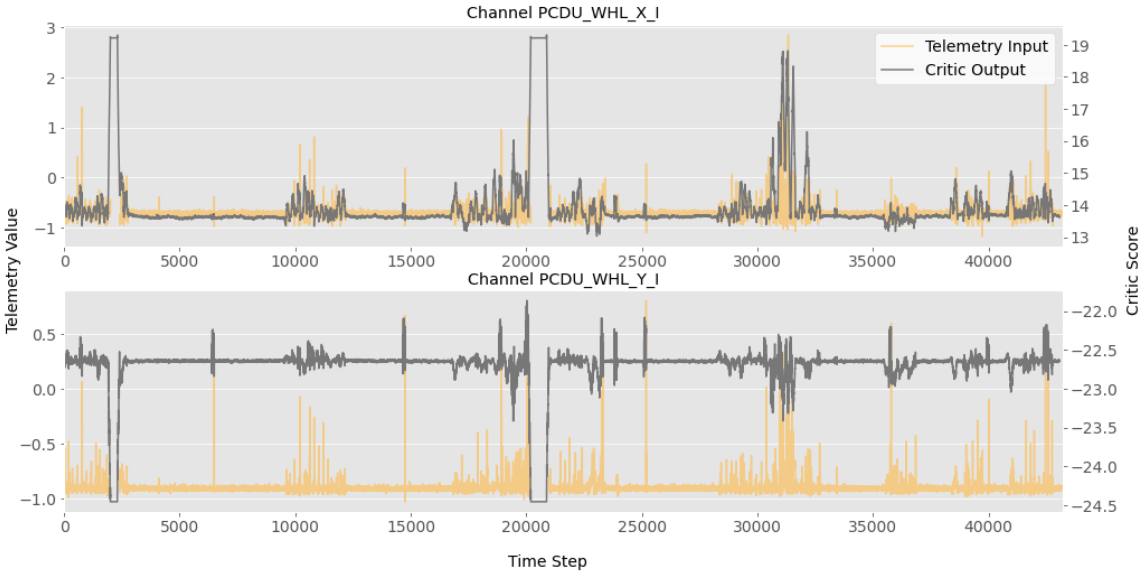

Figure 5.2: Critic outputs of two similar channels, *PCDU_WHL_X_I* and *PCDU_WHL_Y_I*, along with their telemetry inputs.

Compared to the point-wise or area methods for computing reconstruction differences, DTW is higher in both intuitive and computational complexity. Yet it offers no noticeable improvements (Table 5.2). DTW can be more effective when comparing sequences of different lengths or shifted out of step, but in this case the original input and its reconstruction are the same length and already aligned.

Variation	GAN-Cmd	Recon-Cmd	GAN-Multi
Point × Critic	0.60(0.37)	-	0.52(0.33)
Area × Critic	0.58(0.35)	-	0.52(0.33)
DTW × Critic	0.61(0.37)	-	0.52(0.35)
Point	**0.66**(0.38)	**0.64**(0.36)	**0.67**(0.36)
Area	**0.66**(0.37)	0.63(0.37)	0.65(0.37)
DTW	0.65(0.37)	**0.64**(0.37)	0.65(0.37)

Table 5.2: F1 scores of variations of computing the reconstruction error (point-wise, area, and DTW) and anomaly score formations (with and without the critic output). These scores are based on dynamic thresholding with the same error window size $w_e = 0.33$.

The errors in reconstructing time series data are generally greater with GAN-Cmd

(0.279) and GAN-Multi (0.329) than with the reconstruction-only model (0.249); this is expected since adversarial learning involves balancing opposing objectives rather than optimizing a sole objective. Despite the differences in reconstruction errors, the three experiments yield similar performance scores, hinting once more that there is no clear correlation between the errors and the model's ability to detect anomalies. The difference in reconstruction abilities can be observed in channels such as *DBG_ST_CCD_LAST_TEMP* in Figure 5.3.

5.5 Method Summary

GANs have been extensively researched for many years and have shown promise in a range of problems, including the anomaly detection experiments we are running. Yet they are still difficult to train in certain settings, and may not show clear signs of converging. Additionally, we have seen in Figure 5.2 that the behaviour exhibited by the critic networks can be unpredictable, assigning very different scores to seemingly similar data. These uncertainties can lead to resistance from satellite operators in adopting such methods.

In our experiments with TadGAN, learning from and predicting entire packets of related telemetry yields minimal improvements in anomaly detection over predicting individual channels at a time (although it does improve the total training duration, as seen in Appendix A.1). Either the associations among telemetry channels offer no clues when detecting anomalies, or we are not using the appropriate model to leverage those associations. In the next chapter we look at a method that employs graph attention (Veličković et al., 2018) to learn the connections between channels and improve its multivariate time series predictions.

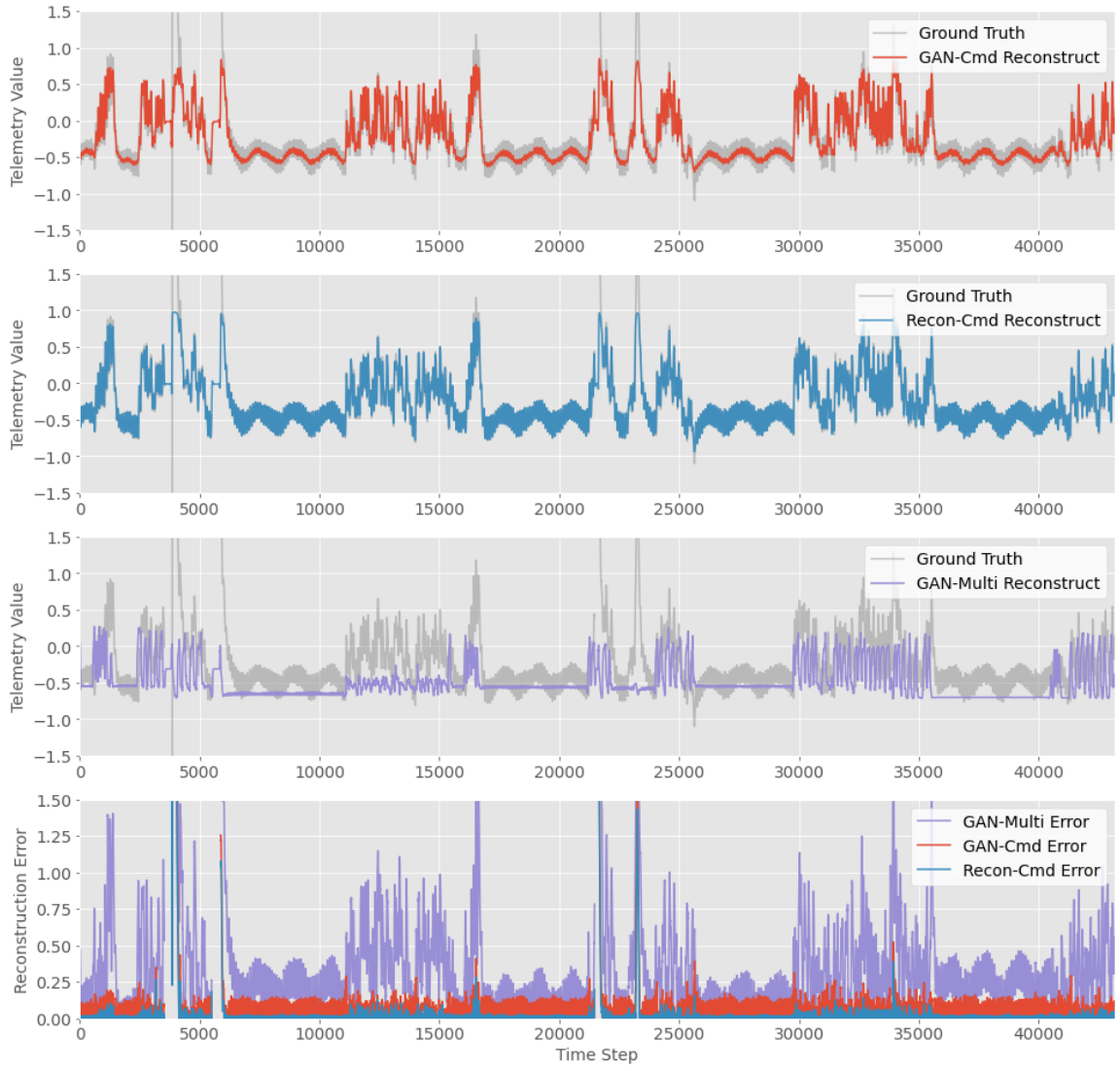

Figure 5.3: Reconstructed telemetry values (top three plots) and reconstruction errors (bottom) of GAN-Cmd, Recon-Cmd, and GAN-Multi experiments on the *DBG_ST_CCD_LAST_TEMP* channel. Despite the differences in reconstruction errors, the three experiments result in similar F1 scores.

6

Multivariate Time-Series Anomaly Detection via Graph Attention Network

In the previous chapter, we reconstructed time series data as part of the anomaly detection process. We saw that the multivariate reconstructions yielded much higher reconstruction errors than the single channel outputs and did not resemble the original samples at all (Figure 5.3). As pointed out by Hundman et al. (2018), LSTMs struggle to predict high-dimensional outputs. What kind of model might be better suited to deal with multiple telemetry channels at once?

Inspired by the success of convolutional neural networks (CNNs) on high-dimensional image-based tasks, graph neural networks (GNNs; Gori et al., 2005; Scarselli et al., 2009) and related methods have emerged to handle data that cannot be represented in a grid-like structure but can be represented as graphs. Of particular interest is the graph attention network (GAT; Veličković et al., 2018), which employs self-attention mechanisms previously shown to excel in language understanding (Cheng et al., 2016; Vaswani et al., 2017). GAT layers can model the relationships among nodes in a graph, and Zhao et al. (2020) utilize this idea to capture correlations between telemetry channels as well as across time steps. Their method, called Multivariate Time series Anomaly Detection via Graph Attention Network (MTAD-GAT), combines forecasting- and reconstruction-based anomaly detection and is the subject of our last set of experiments.

6.1 Technical Background

MTAD-GAT combines graph neural network techniques, an RNN advancement known as gated recurrent unit (GRU), and a thresholding technique called Peaks-Over-Threshold (POT).

6.1.1 Graph Foundations

A graph consists of a set of *nodes* connected by *edges* which define some sort of relationship between pairs of nodes. These nodes, which contain one or more features describing them, may be represented as low-dimensional *node embeddings* that capture both structural plus node information to be as used as inputs in downstream machine learning tasks (Hamilton et al., 2017). Another way to represent the structure of the graph is in the form of an adjacency matrix where the (i, j)-th element indicates the presence or absence of an edge between nodes i and j. A graph that contains an undirected edge between every pair of nodes is called a complete graph.

6.1.2 Graph Convolution

Variations of graph convolution networks (GCN) have been proposed (Bruna et al., 2014; Hamilton et al., 2017; Kipf and Welling, 2017) to aggregate and extract information, similarly to CNNs but on arbitrarily structured graph data. Given a set of node features $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N\}, \mathbf{h}_i \in \mathbb{R}^F$, where N is the number of nodes in the graph and F is the number of features in every node, a graph convolution layer first applies a linear transformation to obtain high-level representations of every node. This transformation $\mathbf{g}_i = \mathbf{W}\mathbf{h}_i$ is parameterized by a weight matrix $\mathbf{W}$ shared across nodes. Then a graph convolution operation is performed which generally consists of an aggregation of features over some neighbourhood $\mathcal{N}_i$ of node i to compute a new set of node features,

$$\mathbf{H}' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\mathbf{g}_j\right), \tag{6.1}$$

where σ is an activation function and α_{ij} represents the importance of node j to node i.

6.1.3 Graph Attention Network

While α_{ij} can be explicitly defined based on the structure of the graph, Veličković et al. (2018) propose a GAT which uses *self-attention* to learn α_{ij}. Stepping away briefly from the graph domain, the concept of attention was first introduced by Bahdanau et al. (2014) to overcome a critical limitation in neural translation models such as *seq2seq* (Sutskever et al., 2014). This model uses an RNN encoder-decoder architecture to read an input sequence, form a compressed context vector of fixed length, and then predict the next output of the sequence given the context vector and its previously generated outputs. A problem with this kind of model is that its predictive performance deteriorates when dealing with long input sequences (Cho, van Merrienboer, Bahdanau, et al., 2014) because then the context vector fails to adequately express key information pertaining to earlier parts of the input. An attention mechanism aims to solve this by assigning weights which signify the importance of inputs near each position j to outputs at position i. A new context vector is defined based on these weights that expresses the most relevant parts of the input sequence in order to make the next prediction.

A GAT leverages this solution in the graph convolution framework. The unnormalized attention scores e_{ij} of each pair of nodes i and j are calculated using a single layer neural network parameterized by $\mathbf{a}$:

$$e_{ij} = \text{LeakyReLU}(\mathbf{a}[\mathbf{g}_i, \mathbf{g}_j]),$$

where $[\cdot, \cdot]$ is the concatenation operation. Any choice of attention mechanism can be substituted to calculate e_{ij}. Then a softmax function normalizes scores across the neighbourhood of node i to produce the α_{ij} term from Equation 6.1 above:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}.$$

This offers an end-to-end learning approach for capturing meaningful information across neighbours of nodes and producing new features $\mathbf{H}$ to be used in graph-based tasks. A method for improving training stability in GAT, called multi-head attention (Vaswani et al.,

2017), combines the results of multiple independent attention mechanisms for each node; however, this technique is not used as part of MTAD-GAT.

6.1.4 Gated Recurrent Unit

Cho, van Merrienboer, Gulcehre, et al. (2014) aim to improve on LSTMs with their invention of a simpler RNN hidden unit called gated recurrent unit (GRU). Figure 6.1 depicts the new process to update the hidden state. Given the input $\mathbf{x}_t$ at time step t, the vector of previous hidden states $\mathbf{h}_{t-1}$, and weights W_l and U_l corresponding to layer l, the unit update equations are as follows:

$$r_t = \sigma(W_r \cdot \mathbf{x}_t + U_r \cdot \mathbf{h}_{t-1}), \tag{6.2a}$$

$$\tilde{h}_t = \tanh(W_h \cdot \mathbf{x}_t + U_h \cdot (\mathbf{r}_t \odot \mathbf{h}_{t-1})), \tag{6.2b}$$

$$z_t = \sigma(W_z \cdot \mathbf{x}_t + U_z \cdot \mathbf{h}_{t-1}), \tag{6.2c}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t. \tag{6.2d}$$

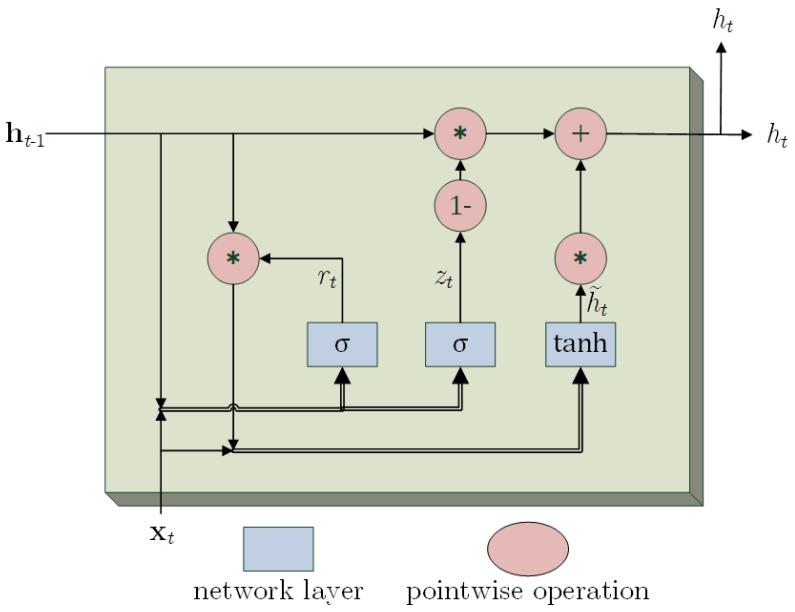

Figure 6.1: Operations involved in a single GRU, given input data $\mathbf{x}_t$ and the previous hidden states $\mathbf{h}_{t-1}$.

Two new types of gates are defined, with the first being a *reset gate* $r \in [0, 1]$ which determines whether to forget the previously computed hidden state (Equation 6.2a). A value of 0 effectively acts as if resetting the unit to read the beginning of an input sequence. Equation 6.2b computes the candidate values $\tilde{h}_t$ with which to update the hidden state, where $\odot$ denotes element-wise multiplication. The *update gate*, which is the other type of gate, outputs coefficients z_t used in the convex combination of the previous and candidate hidden states (Equations 6.2c-6.2d). In other words, z_t controls how much the previous hidden state is updated with candidate values to produce the new hidden state h_t.

Like LSTM units, GRUs improve on the traditional RNN model through their ability to retain important features of an input sequence from many time steps in the past. Their use of gating units allows error signals to more easily be backpropagated without encountering the problem of vanishing gradients (Bengio et al., 1994; Hochreiter, 1991). Based on empirical evaluations done by Chung et al. (2014), there is no clear consensus on which type of hidden unit performs better in sequence modeling tasks.

6.1.5 Peaks-Over-Threshold

MTAD-GAT uses a thresholding approach called Peaks-Over-Threshold (POT), originating from Siffer et al. (2017). The basis of POT is in Extreme Value Theory (EVT; Beirlant et al., 2004), in particular its second theorem, known as the Pickands-Balkema-de Haan theorem (Balkema and de Haan, 1974). This states that the tail $\bar{F}_u(x)$ of a probability distribution can be well approximated by a generalized Pareto distribution (GPD) without making any assumptions on the underlying distribution as a whole. Given a sequence of data points $\mathbf{x}$ and some level u, *peaks* are defined as $\mathbf{y} = \mathbf{x} - u$, for $\mathbf{x} > u$. In other words, peaks are the excess values above the level (we use this term instead of threshold for u to avoid confusion with the anomaly score threshold). As u approaches the upper bound U of the underlying distribution, the peaks follow a GPD parameterized by a shape γ and scale σ:

$$\bar{F}_u(x) = P(\mathbf{x} - u > x \mid \mathbf{x} > u) \rightarrow \left(1 + \frac{\gamma x}{\sigma}\right)^{-1/\gamma}, \text{ as } u \rightarrow U.$$

The γ and σ values are computed using maximum likelihood estimation (MLE), where the goal is to maximize the log-likelihood $\mathcal{L}(\gamma, \sigma)$ of peaks $\mathbf{y} = \{y_1, y_2, \ldots, y_{N_u}\}$ given

those parameters:

$$\mathcal{L}(\gamma, \sigma) = -N_u \log \sigma - \left(1 + \frac{1}{\gamma}\right) \sum_{i=1}^{N_u} \log\left(1 + \frac{\gamma}{\sigma} y_i\right). \tag{6.3}$$

Optimizing $\mathcal{L}(\gamma, \sigma)$ involves searching for solutions to $\nabla\mathcal{L}(\gamma, \sigma) = 0$. Davison (1984) observes that the bivariate problem can be reduced to a single variable equation through a reparameterization of (γ, σ) to (θ), where $\theta = \gamma/\sigma$. Making the following substitutions,

$$\sigma = \frac{\gamma}{\theta}, \tag{6.4}$$

$$\gamma = -\frac{1}{N_u} \sum_{i=1}^{N_u} \log(1 - \theta y_i), \tag{6.5}$$

results in a log-likelihood parameterized only by θ:

$$\mathcal{L}(\theta) = -N_u - \sum_{i=1}^{N_u} \log(1 - \theta y_i) - N_u \log\left(-\frac{1}{N_u} \sum_{i=1}^{N_u} \frac{\log(1 - \theta y_i)}{\theta}\right).$$

Grimshaw (1993) takes advantage of this reparameterization and formulates the optimization problem as solving $\frac{d}{d\theta}\mathcal{L}(\theta) = 0$. This is equivalent to finding the set of zeros, denoted $\boldsymbol{\theta}^0$, of

$$h(\theta) = \left(1 + \frac{1}{N_u} \sum_{i=1}^{N_u} \log(1 - \theta y_i)\right) \left(\frac{1}{N_u} \sum_{i=1}^{N_u} \frac{1}{1 - \theta y_i}\right) - 1.$$

Grimshaw (1993) proves that certain properties of $h(\theta)$ help constrain the search space of zeros. Once found, converting each $\theta^0 \in \boldsymbol{\theta}^0$ back to GPD parameters (Equations 6.4-6.5) creates a set of candidate parameters $\mathbf{C}$ for producing the maximum of the GPD log-likelihood. An additional candidate consisting of boundary values at $(\gamma = 1, \sigma = \max(\mathbf{y}))$ is appended to $\mathbf{C}$. Then identifying the optimal parameters $\hat{\gamma}$ and $\hat{\sigma}$ is a matter of evaluating Equation 6.3 with each $(\gamma_c, \sigma_c) \in \mathbf{C}$.

After obtaining parameters $\hat{\gamma}$ and $\hat{\sigma}$ of the fitted GPD, the probability of extreme observations can be evaluated. By setting some probability q (also known as the risk), it is then possible to calculate a threshold ε_q such that $P(\mathbf{x} > \varepsilon_q) < q$. This threshold is used as the

anomaly score threshold and is expressed as

$$\varepsilon_q = u + \frac{\hat{\sigma}}{\hat{\gamma}} \left(\left(\frac{qn}{N_u} \right)^{-\hat{\gamma}} - 1 \right),$$

where n is the length of data points $\mathbf{x}$ and N_u is the number of peaks.

6.2 Method Details

6.2.1 Feature Extraction

The first step of the MTAD-GAT model involves a 1-dimensional CNN to extract high-level features from windows of input data. Given an input $\mathbf{X} \in \mathbb{R}^{T \times N}$ containing N telemetry channels over a window of T time steps, the convolution layer produces $\mathbf{H} \in \mathbb{R}^{T \times F}$ where each column $\mathbf{h}_f \in \mathbf{H}$ is expressed as

$$\mathbf{h}_f = \sum_{n=1}^{N} \mathbf{w}_{f,n} \star \mathbf{x}_n.$$

Here, $\star$ denotes the cross-correlation operator and $\mathbf{w}_{f,n}$ is the vector of learnable weights, or kernel, corresponding to output channel f and input channel n. The input is padded to return sequences of length T, and the number of output channels F matches the number of input channels N.

6.2.2 Graph Attention for Temporal and Feature-based Dependencies

MTAD-GAT makes use of two separate GAT layers, shown in Figure 6.2, to simultaneously learn temporal and feature-based dependencies. One layer—GAT$_{time}$—operates on a complete graph where each node $\mathbf{h}_t \in \mathbb{R}^F$ consists of the vector of telemetry values at time t. This allows relationships between time steps to be learned and produces $\mathbf{H}'_{time} \in \mathbb{R}^{T \times F}$. The other layer, GAT$_{feat}$, identifies correlations between telemetry channels by operating on another complete graph where each node $\mathbf{h}_f \in \mathbb{R}^T$ contains the telemetry values of channel f from the same window of time. The output of this second layer is $\mathbf{H}'_{feat} \in \mathbb{R}^{F \times T}$, where each row is a T-dimensional vector representing the output of each node. The GAT

layer outputs are concatenated along with the output of the previous 1-dimensional convolution, forming $\tilde{\mathbf{X}} = [\mathbf{H}, \mathbf{H}'^{\mathsf{T}}_{feat}, \mathbf{H}'_{time}] \in \mathbb{R}^{T \times 3F}$.

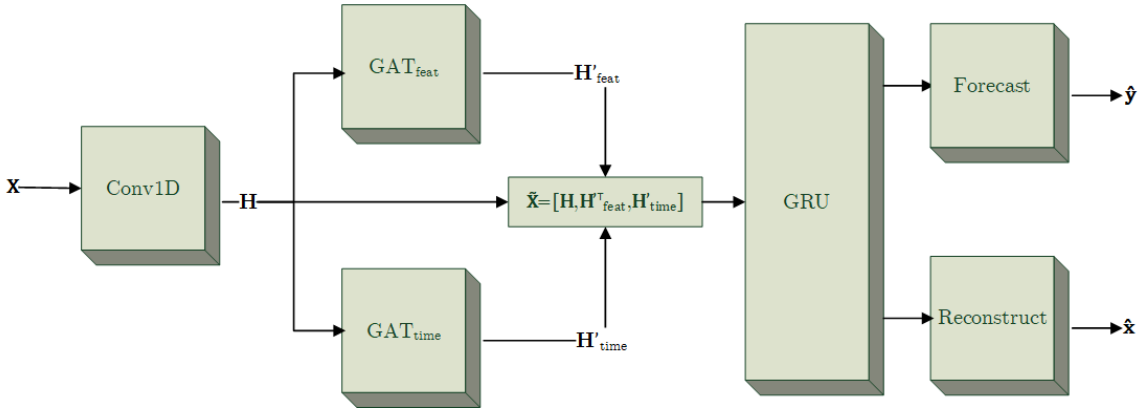

Figure 6.2: MTAD-GAT model architecture.

From here, GRU-based layers (Cho, van Merrienboer, Bahdanau, et al., 2014) produce a forecast $\hat{\mathbf{y}}^{t+1} \in \mathbb{R}^N$ of future time series values in addition to a reconstruction $\hat{\mathbf{x}}^t \in \mathbb{R}^N$ of the current window of telemetry in order to leverage both types of predictions in detecting anomalous time steps. While the authors of MTAD-GAT propose using a Variational Autoencoder (VAE; Kingma and Welling, 2014; D. J. Rezende et al., 2014) to yield a probabilistic reconstruction score, we find that not enough details are provided to implement this component and instead opt for a standard autoencoder. This idea is borrowed from the creator of an unofficial implementation of MTAD-GAT[1], who cites instability and difficulty in training VAEs. The overall loss thus combines the forecasting-based and reconstruction-based errors, calculated across all channels being predicted:

$$\mathbf{e}^t_{forecast} = \text{RMSE}(\mathbf{x}^t, \hat{\mathbf{y}}^t),$$

$$\mathbf{e}^t_{reconstruct} = \text{RMSE}(\mathbf{x}^t, \hat{\mathbf{x}}^t),$$

$$L = \mathbf{e}^t_{forecast} + \mathbf{e}^t_{reconstruct},$$

where RMSE is the root mean square error. Lastly, the anomaly scores for each channel n at each step t in the test set are described by a similar combination of forecast and

[1] https://github.com/ML4ITS/mtad-gat-pytorch

reconstruction errors: $s_n^t = e_{n,forecast}^t + e_{n,reconstruct}^t$.

6.2.3 Streaming Peaks-Over-Threshold with Drift

Instead of using the raw anomaly scores $\mathbf{s}_n$ to calculate the POT threshold ε_q, we use relative differences $\delta_n^t = s_n^t - m^t$ where m^t is a moving average of the last d scores,

$$m^t = \frac{1}{d}\sum_{k=1}^{d} s_n^{t-k}.$$

This alternate form of POT takes into account any drift of non-stationary signals by modeling changes from the average local behaviour represented as m^t. Additionally, we apply POT in a streaming fashion, allowing ε_q to be continuously updated with new information. First, an initial value for ε_q is calculated with relative anomaly scores from nominal training data, as a calibration step. Then δ_n^t of test data is evaluated sequentially in which one of the following three cases can occur:

- If δ_n^t exceeds ε_q, s_n^t is added to the set of predicted anomalies $\mathcal{A}$;
- If δ_n^t exceeds the level u but not ε_q, it is added to the set of peaks $\mathbf{y}$ and ε_q is updated based on new estimates of GPD parameters $\hat{\gamma}$ and $\hat{\sigma}$;
- If δ_n^t does not exceed u nor ε_q, nothing changes.

Sequences of anomalies in $\mathcal{A}$ are pruned and given a normalized anomaly score, following the same process as Telemanom and TadGAN. To illustrate the differences between POT, dynamic, and fixed thresholds, Figure 6.3 shows an example of these thresholds when calculated on smoothed error values of a continuous channel.

6.3 Experiments

We initially perform three variations of experiments in this chapter:

- **GAT-Multi**—following the original MTAD-GAT method, we attempt multivariate forecasting and reconstruction of multiple channels at once by training one model for each packet of telemetry channels;

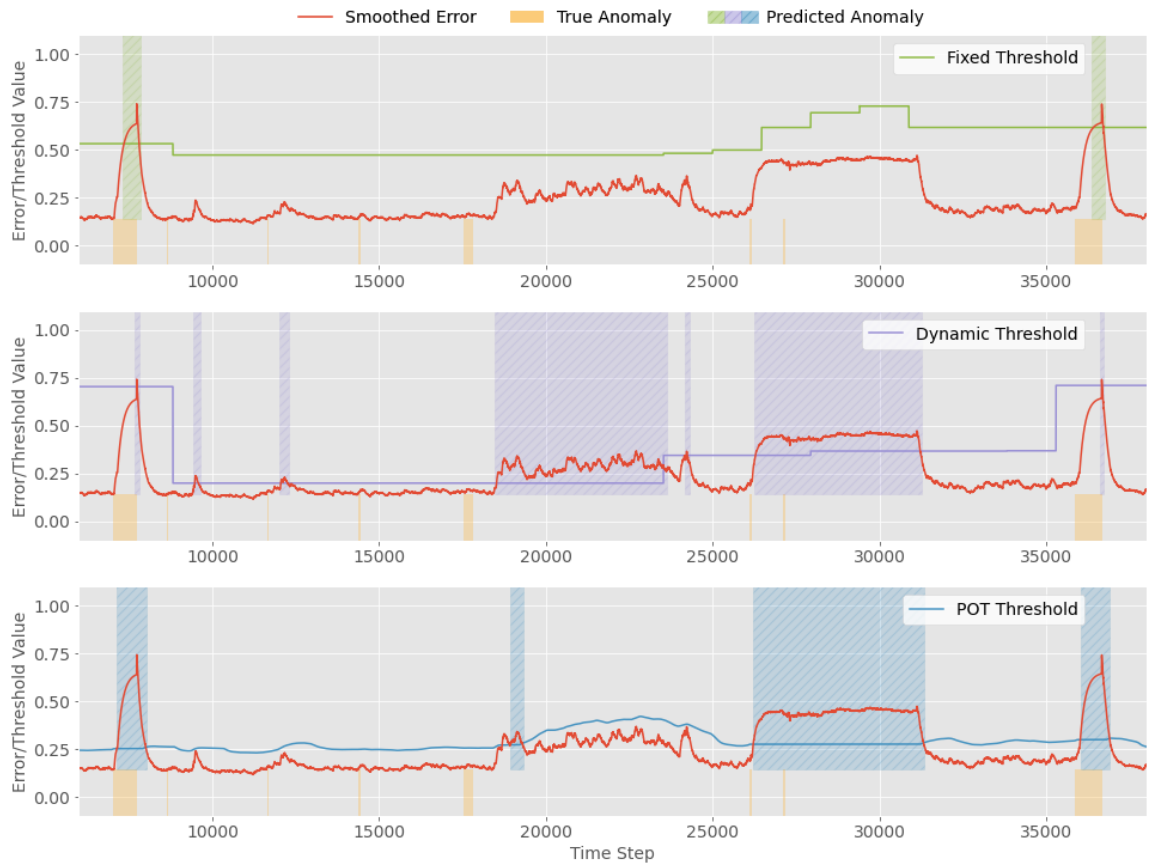

Figure 6.3: Fixed, dynamic, and POT thresholds calculated on smoothed errors for channel *PCDU_GPS_I*. Predicted and true anomaly sequences are highlighted in the background. Anomalies are predicted wherever the error exceeds the threshold.

- **GAT-Single**—we train models on packets of telemetry channels to produce predictions for a single channel at a time, in response to the high prediction errors seen in experiments with multivariate outputs;
- **GRU-Single**—we then remove the GAT layers from the model and perform forecasting and reconstruction of a single channel based on GRU layers only.

Model Configuration

The MTAD-GAT model begins with a 1-dimensional convolution with a kernel size of 3 and ReLU activation. This is followed by simultaneous GAT layers that produce $\mathbf{H}'_{feat}$ and

$\mathbf{H}'_{time}$ as defined in Section 6.2.2. The next layer is a GRU RNN that is shared by both the forecasting and reconstruction sides of the model. Forecasting is then achieved with two hidden fully connected layers with ReLU and dropout, plus another fully connected output layer. At the same time, the decoder half of the reconstruction network consists of a GRU-based layer in addition to a fully connected output layer. We try different settings for the number of units—80, 150, and 300—in the GRU layers as well as the hidden fully connected layers for each packet type.

Each model is trained for up to 35 epochs with early stopping, as in the Telemanom experiments. We employ the Adam optimizer using a learning rate of 10^{-3}.

Method Evaluation

For evaluation with POT, we try different values of risk $q \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ and initial level $u \in \{0.97, 0.99, 0.999\}$, followed by pruning. We also evaluate model performance with dynamic thresholds for comparison. Additional specifics on implementation as well as training and evaluation configurations are found in Appendix A.5.

6.4 Results

The results of the initial experiments in this chapter are presented in Table 6.1. We observe right away that the overall performance using POT thresholding is better than with dynamic thresholding. We also note that the single channel output models achieve similar F1 scores whether they incorporate GAT layers or not.

	Dynamic Threshold			POT Threshold		
Experiment	Precision	Recall	F1	Precision	Recall	F1
GAT-Multi	0.49(0.29)	0.92(0.20)	**0.60**(0.29)	0.63(0.30)	0.94(0.11)	0.70(0.27)
GAT-Single	0.57(0.36)	0.91(0.11)	0.59(0.36)	0.76(0.26)	0.97(0.10)	0.81(0.26)
GRU-Single	0.52(0.31)	0.81(0.21)	0.58(0.26)	0.83(0.26)	0.92(0.17)	**0.82**(0.25)

Table 6.1: Mean(standard deviation) of monthly results of MTAD-GAT experiments with dynamic and POT thresholds.

Like in dynamic thresholding, pruning plays an important role in the performance of

POT. As the amount of pruning increases, the precision score elevates at a sharp rate while the recall is largely unaffected until p exceeds 0.5 (Figure 6.4).

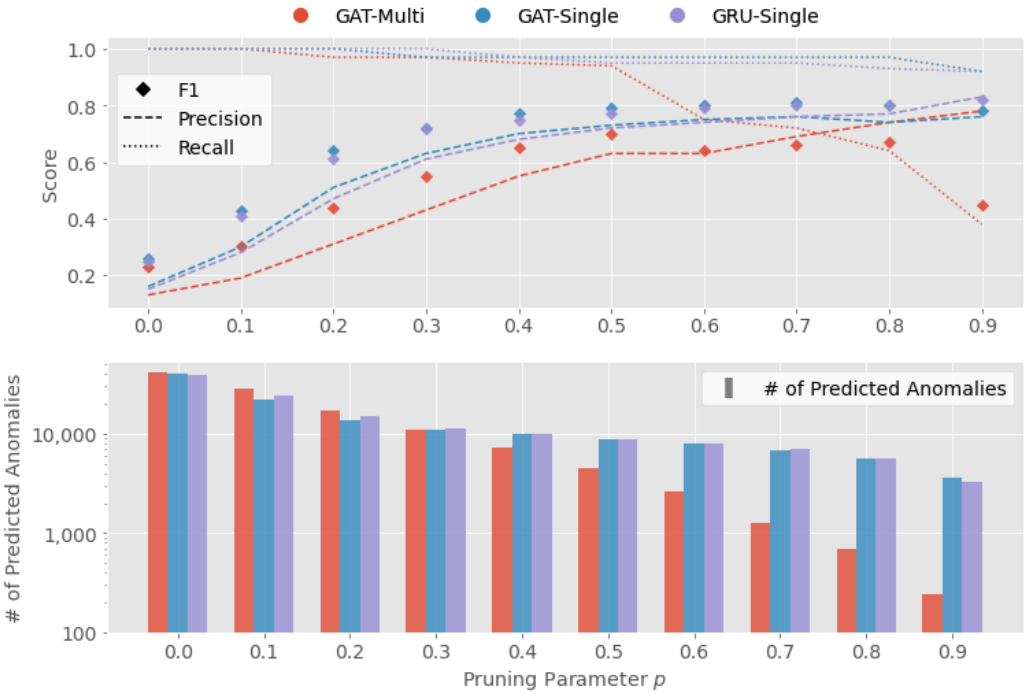

Figure 6.4: Impact of the anomaly pruning parameter p after POT thresholding on mean precision, recall, and F1 scores (top); and on the number of predicted anomalies across all channels (bottom).

6.4.1 Ablation Study

The two latest model variations we analyze, GAT-Single and GRU-Single, achieve the highest overall F1 scores we have seen so far. To characterize the impact of the pieces that make up the GRU-Single model, we conduct additional ablation experiments:

- **GRU-Forecast**—we decompose the GRU-Single experiment to perform only forecasting of a single channel based on GRU layers;
- **GRU-Recon**—this is similar to GRU-Forecast but we use reconstruction errors in training and evaluating single channels, instead of forecasting errors;

- **GRU-Solo**—this model takes as input a single channel with no other covariates to perform forecasting and reconstruction of that same channel;
- **GRU-NoConv**—we remove the initial 1-dimensional convolution layer from the GRU-Single experiment's model.

The results of these four experiments are summarized in Table 6.2. The effect of withholding components of the GRU-Single model appears to be small when either dynamic thresholds or POT thresholds are applied. In some cases the mean F1 score actually improves compared to GRU-Single.

	Dynamic Threshold			POT Threshold		
Experiment	Precision	Recall	F1	Precision	Recall	F1
GRU-Forecast	0.49(0.32)	0.94(0.11)	0.56(0.35)	0.80(0.25)	0.91(0.13)	0.81(0.23)
GRU-Recon	0.58(0.30)	0.85(0.20)	**0.62**(0.31)	0.83(0.28)	0.83(0.31)	**0.83**(0.29)
GRU-Solo	0.59(0.32)	0.71(0.20)	0.59(0.22)	0.79(0.27)	0.81(0.30)	0.79(0.27)
GRU-NoConv	0.54(0.35)	0.83(0.19)	0.56(0.35)	0.80(0.28)	0.80(0.31)	0.79(0.27)

Table 6.2: Mean(standard deviation) of monthly results of MTAD-GAT ablation experiments with dynamic and POT thresholds.

6.5 Method Summary

It turns out that while graph attention is a promising addition to the anomaly detection model, it does not appear to be responsible for the high F1 scores achieved in our experiments of this chapter. Despite this, the graph-based methods provide the advantage of revealing how other channels are associated to the one being predicted, in the form of attention scores. We will see in Section 7.3 that these scores could be helpful in diagnosing predicted anomalies.

In this latest round of experiments, POT thresholding shows consistent improvements over dynamic thresholding. This encourages us to revisit the methods we have previously tested and evaluate their performance using POT thresholds. An overall comparison of our anomaly detection methods using all types of thresholds is presented in our penultimate chapter, along with analysis of the experimental results.

7 Discussion

Having conducted various experiments with the anomaly detection methods of interest, it is now time to analyze the results. We first focus on the overall F1 score to get a sense of how the different prediction models and thresholding techniques contribute to each method's performance. Recognizing that this metric does not tell the whole anomaly detection story, we likewise discuss the true positive and false positive predictions as well as the available information for interpreting predicted anomalies.

7.1 Comparison of Methods

To compare the different combinations of models and thresholding techniques, the F1 scores of our main experiments are gathered in Table 7.1. Precision and recall scores are omitted from this table for clarity and can be found in the extended table in Appendix B.1, along with the ablation results.

7.1.1 Model Performance

Generally, we see a trend of increasing scores over the progression of our experiments; the top GAT and GRU models perform better than the best GAN-based model which beats the LSTM methods. The ablation study in Section 6.4.1 is an attempt to decompose the top scoring model and identify a possible cause for its high performance. Surprisingly, there is little change in average F1 scores among the models which employ GRU layers, except

	Threshold		
Experiment	Fixed	Dynamic	POT
LSTM-Cmd	0.56(0.29)	**0.64**(0.27)	0.57(0.29)
LSTM-Solo	0.55(0.28)	**0.63**(0.31)	0.60(0.35)
GAN-Cmd	0.53(0.34)	0.66(0.37)	**0.76**(0.25)
Recon-Cmd	0.52(0.33)	0.64(0.34)	**0.69**(0.25)
GAN-Multi	0.54(0.35)	**0.67**(0.33)	0.65(0.24)
GAT-Multi	0.48(0.28)	0.60(0.29)	**0.70**(0.27)
GAT-Single	0.54(0.27)	0.59(0.36)	**0.81**(0.26)
GRU-Single	0.55(0.28)	0.58(0.26)	**0.82**(0.25)

Table 7.1: Mean(standard deviation) F1 scores of all variations of anomaly detection models across all test months, evaluated with fixed, dynamic, and POT thresholds. See Table 8.1 for a recapitulation of experiments.

for GAT-Multi. This leads us to wonder if it is the GRUs driving the high F1 scores. Some of our GRU-based experiments are similar in setup to the LSTM-based models, and we do not expect to see this much of a difference in their results. Studies by Chung et al. (2014), Jozefowicz et al. (2015) and Cahuantzi et al. (2021) conclude empirically that neither GRUs nor LSTM units clearly outperform the other on a variety of sequence modeling tasks. From the prediction errors listed in Table 7.2, however, we see that GAT-Single and GRU-Single do in fact generate the lowest errors in our setting. Their superior anomaly detection scores may be attributed to this increased accuracy in forecasting and reconstructing time series values.

	LSTM-Cmd	LSTM-Solo	GAN-Cmd	Recon-Cmd	GAN-Multi	GAT-Multi	GAT-Single	GRU-Single
Mean Error	0.158	0.136	0.279	0.249	0.329	0.160	**0.120**	**0.123**
Standard Dev.	0.030	0.029	0.016	0.019	0.018	0.026	0.019	0.019

Table 7.2: Prediction errors (forecasting- and reconstruction-based) of each model averaged across all test months.

We expect to notice such correlation between prediction error and anomaly detection performance in other experiments too since the premise of our models is to accurately generate nominal spacecraft telemetry to detect large deviations from this predicted behaviour. Despite our initial observations in Sections 4.4 and 5.4 claiming otherwise, these correlations are now apparent when considering the POT threshold scores. Examples of this are seen in the models that output multiple channels at once, GAN-Multi and GAT-Multi; they have higher errors and lower F1 scores than their counterparts which produce univariate predictions. Deep learning models are known to be able to handle input and output data that are high-dimensional—we need only turn to image generation tasks to see evidence of this. Multivariate time series prediction, while interesting from a research perspective, is less relevant for satellite operators if the single channel models perform better. For our task, developing models that further reduce the prediction errors may be an appropriate challenge to focus on, but developing models to predict multiple features over many time steps may be less so.

In the LSTM experiments, including one-hot encoded command information as covariates adds little to the anomaly detection performance when applying a fixed or dynamic threshold, and even lowers it with a POT threshold. As noted earlier, LSTMs have difficulty predicting abrupt changes in telemetry. We leverage past commands and telemetry as multivariate inputs to forecast future telemetry values, but we could also use future command information to improve our predictions. This information is available because spacecraft command sequences are nominally planned well in advance of their execution. To incorporate future commands into our model input, the command covariates need to be shifted in time by 1 step, so that at each time step our model takes the telemetry values up to time t and the commands up to time $t + 1$ to forecast the telemetry at time $t + 1$ (Herzen et al., 2021). This allows the model to learn the relationship between future commands and future telemetry, rather than only the relationship between past commands and future telemetry. Additionally, the creators of Telemanom suggest using more granular command information to increase prediction performance. Including more details in the input data, such as specific parameters (i.e., arguments) associated with each command, may improve the prediction errors as well as the method's ability to detect the contextual class of anomalies.

7.1.2 Thresholds and Pruning

Although the POT technique does not always perform better than a dynamic threshold, the highest F1 scores we see from our experiments come from POT thresholding. We observe in the extended results of Table B.1 that, for LSTM-Cmd and LSTM-Solo, the diminished F1 scores with POT stem from low recall. Both dynamic and POT thresholding perform significantly better than with a simple fixed threshold, except for the LSTM experiments where the scores with POT are closer to the fixed threshold results. By looking at the variance in results achieved with different parameter settings, presented in Appendix B.2, we cannot conclude that either dynamic or POT thresholding is less sensitive to its parameters than the other. Overall, POT yields a smaller spread of F1 scores across the ten months of test data.

Dynamic and POT thresholding achieve high scores only in conjunction with the anomaly pruning process. Before pruning, both types of thresholds attain recall scores close to 1 and abysmal precision, resulting in overall terrible F1 scores. We see examples of this in Figures 4.3 and 6.4, where both types of thresholds at $p = 0.0$ produce tens of thousands of predicted sequences (over all the channels). Once pruning is introduced, the dramatic improvement to precision scores outweighs the minor reduction in recall scores up to a certain level of p. Pruning is effective because it leaves behind only the predicted anomalies with the highest anomaly scores for each channel, and for the purposes of calculating recall, only one channel needs to successfully detect an anomaly for it to be considered a true positive. However, pruning cannot be applied alone, as candidates for pruning must first be found and associated with appropriate anomaly scores. This is why the choice of threshold is important; it sets the groundwork for anomaly pruning to be successful.

In our experiments we identify the best scores achieved over a range of parameters related to thresholding and pruning, by exploiting ground truth labels. This is done in order to analyze the top possible results of each method and to investigate their potential for detecting anomalies. We also find this to be the fairest manner for comparing methods; after witnessing the F1 scores fluctuate considerably over different p levels, it seems that comparing all of the results with the same fixed p may not have much meaning. In a real setting, satellite operators may be able to get a sense of the appropriate value to set for the

pruning parameter after sufficient anomaly labels are collected. This level can be readjusted over time as new information comes in or based on alternative weightings of precision and recall instead of a balanced F-measure. If p is unable to be set with certainty then thresholds can still be established to produce candidate anomalies with corresponding anomaly scores. We will see shortly in Section 7.3 whether anomaly detection can reasonably be done in the absence of pruning.

7.1.3 Variance in Results

Much of our discussion is on the mean F1 scores of each method, but Table 7.1 also shows the standard deviations over the ten months of test data. Aside from the LSTM models, the POT scores show less variance than the other two threshold types. Overall, the standard deviation values are relatively large, prompting a closer look at each test month.

Results by Month

In Figure 7.1, the dynamic threshold results show incredibly low scores in September, January, and March for most methods. No model consistently comes out on top in every month. Meanwhile, with the POT threshold all methods show poor results in March. The LSTM models have the lowest scores in every month of the POT results while GAT-Single and GRU-Single achieve close to the highest scores. Aside from the LSTM experiments, there almost looks to be a downward trend in F1 scores over the test months for POT results, even without considering March. If such a downward trend existed, it might be due to the evolving state of the satellite over close to a year of operations.

The poor F1 scores in September, January, and March are not due to recall levels, but rather stem from low precision. This is evident in the plots of precision and recall of each method provided in Appendix B.5. In some cases, a method with relatively high precision in a month can still produce more false positives than a low-precision method, if it also is able to detect abundantly more true anomalies. An example of this occurs in September when comparing the POT results for GAT-Single and LSTM-Cmd: 54 out of the 237 anomalies predicted by GAT-Single are false positives, resulting in a precision of 0.77; LSTM-Cmd has a precision of 0.06 and while detecting only 16 false positives. The reason this is possible is because we count true positives (and false positives) for each channel, so

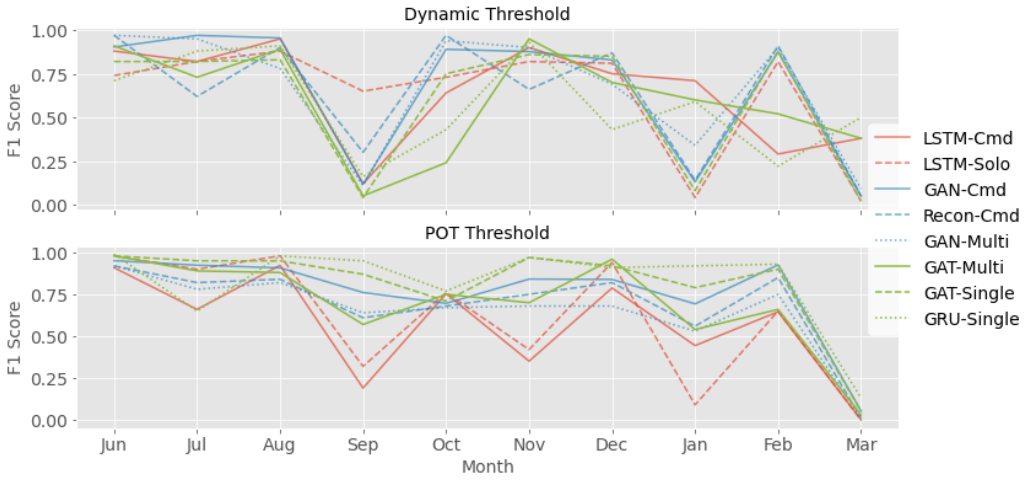

Figure 7.1: F1 scores of every method for each test month.

multiple true positive results detected by a method could all be the same anomaly. While we think this is still a fair way to compare methods, it is worth bringing up because of the burden placed on spacecraft operators when copious amounts of false positives are raised. Section 7.3 offers an alternative perspective of true versus false predictions, where overlapping anomalies predicted across multiple channels are counted together.

Results by Packet ID

Taking a glance at the results grouped by packet ID in Figure 7.2, we notice the precision scores drop for packet *ST_TLM* compared to the rest. This packet contains a mix of discrete and continuous channels for attitude determination, and it is unclear why it produces lower scores. We show precision instead of F1 scores in this figure because each packet type only detects a certain set of anomalies, so taking recall out of the equation allows for a fairer assessment of each packet.

7.2 Predicted Anomalies

Figure 7.3 displays the true anomalies detected by each method using both dynamic and POT thresholds. The anomaly with ID *29-41* is not detected at all with POT, even though it is a reoccurrence of a previously detected case. The LSTM models with POT have the

7.2 Predicted Anomalies

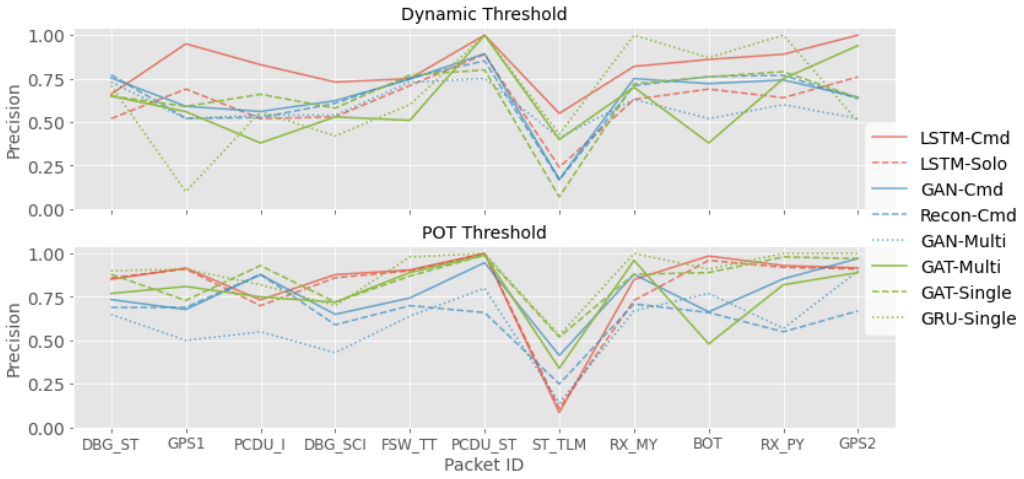

Figure 7.2: Precision scores of every method, grouped by Packet ID.

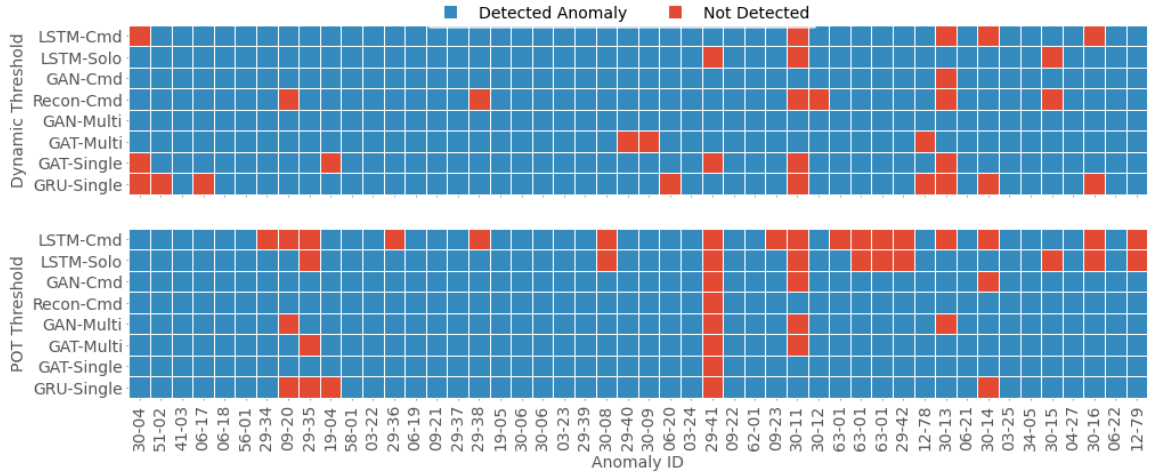

Figure 7.3: Detected and undetected anomalies by each method.

largest number of false negative results, many of which land in the month of November. This helps explain the lower POT scores with LSTM compared to the other models. The GRU-Single experiment with dynamic thresholds also contains many false negatives. We note that this figure does not exactly convey which types of anomalies can be detected by each method, since they are all detected when the pruning amount p is set to 0.0. Rather, this figure shows the results based on the highest F1 score, to give us clues as to why methods perform better than others and where to begin investigating the reasons behind

these differences.

False Positive Analysis

One of the top performing models, GAT-Single, yields a much higher recall than precision. We investigate the predicted anomaly sequences of this method by selecting ten false positive results with high anomaly scores to submit to the NEOSSat satellite operations team for verification. We also provide the telemetry channels tied to these high scores to help in the analysis, in the same manner that a predicted anomaly might be presented in real-time operations. The operators are able to come up with explanations for six of the ten cases. Two are due to actions taken to recover from prior anomalies; these types of false alarms can easily be filtered out since the actions are planned procedures that cause the spacecraft to exhibit different but known behaviour. Four false positive cases coincide with GPS receivers resetting sporadically, which are not part of nominal operations but do not significantly impact the spacecraft and thus are not reported as anomalies. This reveals that anomaly scores may well indicate the presence of anomalous behaviour, since components resetting by themselves is clearly anomalous, but the severity of the anomaly may be poorly reflected by the scores. No causes are associated with the remaining four false positives. While we do not unearth any new anomalies through this investigation, the results are still encouraging now that we are able to back up some of our falsely predicted sequences with reasonable explanations. The date range of each case and their verification outcomes are summarized in Appendix B.3.

Detecting Collective Anomalies

In the evaluation phases of the anomaly detection methods, we set thresholds and predict anomalies for individual channels at a time, which grants traceability of predicted anomalies down to the channel level (Hundman et al., 2018). Doing it this way may hinder the ability of these methods to detect collective anomalies which are only anomalous across multiple channels and not when considered separately. However, Figure 7.3 shows that just about every single true anomaly can be detected by our methods. It seems that the time range of our dataset may not contain any actual instances of collective anomalies; it is also imaginable that these cases went unnoticed and were not reported by satellite operators.

These are exactly the sort of anomalies that are less obvious to detect by humans, where machine learning models might prove their worth. Since we do not have the appropriate dataset for evaluating purely collective anomalies, we do not investigate them further.

7.3 Interpretability

We rely on meaningful anomaly scores, of course, to calculate a threshold that sets apart nominal and anomalous behaviours. These scores also provide some measure of anomalousness for predicted anomalies, which adds colour to the otherwise black and white classification achieved by solely setting a threshold. This means operations engineers have an alternative way to investigate anomalies, by prioritizing their efforts according to the top scoring anomalies. One way for us to analyze the significance of anomaly scores is based on this idea, by sorting predicted anomalies by their normalized score to determine if true anomalies appear near the front.

We gather overlapping predicted anomaly sequences across all channels and retain the maximum anomaly scores of those sequences to produce plots like in Figure 7.4, which illustrates the distribution of true positive and false positive results among the top 100 sequences (plots for the rest of the methods are found in Appendix B.4). In LSTM-Cmd, GAN-Cmd, and GRU-Single without pruning, the true positive cases are spread out among the sorted anomaly scores with false positives interspersed between. In GRU-Single with pruning, the truly detected anomalies are more clustered near the top scores. The effect of pruning is also apparent when comparing the two GRU-Single plots. While the number of detected anomalies based on the top anomaly scores is promising, the fact that many false positives still exist among them may be a deterrent for spacecraft engineers. Additionally, we observe greater variance (i.e., a more noticeable slope) in anomaly scores with POT thresholding, indicating that the assigned scores are different enough to be sorted. This is in contrast to the flatter scores from dynamic thresholding which appear so close in value that assigning priority to predicted anomalies may not be straightforward. We note that comparing scores across different channels like this still requires the calculation of a threshold ε^* since it is used in Equation 4.3.

Ultimately, the goal of interpretability in the methods we are examining is to provide

7.3 Interpretability

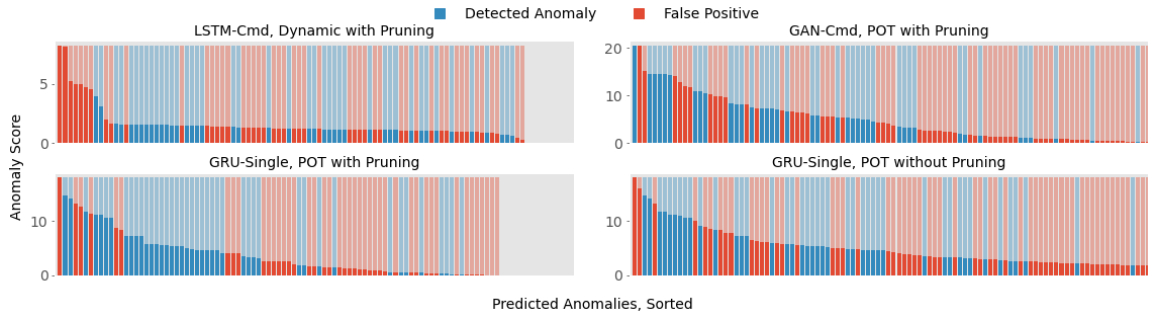

Figure 7.4: Top 100 predicted anomaly sequences sorted by anomaly score with true positive and false positive labels, for selected experiments.

spacecraft operators with the information needed to investigate, confirm, and explain a predicted anomaly. Although these methods point out the telemetry channels that are producing large anomaly scores, this only gives a starting point for an investigation by the operators. The anomalies, if they truly are anomalies, may be rooted in a different subsystem and difficult to trace.

Inspired by Zhao et al. (2020), we investigate if graph attention scores can provide supplementary information for diagnosing anomalies. The feature-based graph attention layers used in the GAT-Multi and GAT-Single models identify other channels that are relevant to the ones that are being predicted. We can observe the trend of attention scores at nominal time steps and compare them with scores calculated during predicted anomalies. As an example, Figure 7.5 displays attention scores α_{ij} for channel $i = 4$ from packet *DBG_SCI* calculated over one month, which includes three periods of true anomalies. The increase or drop in scores during these periods indicates how the anomalies affect the correlation of channels i and j; this can be observed in channels $j = 1, 6, 9,$ and 18, to name a few. This kind of information may help in figuring out the root cause of an anomaly.

Furthermore, we notice that the three anomalous periods show distinct visual patterns in the attention scores presented in Figure 7.5. Could these patterns be considered signatures of their respective anomalies, used to detect and classify reoccurrences? We briefly explore this idea by comparing Anomaly 06-16 with four of its reoccurrences, shown in Figure 7.6. There is arguably some resemblance across all five instances of this same anomaly type. In particular, Reoccurrences 1 and 2 exhibit very similar-looking patterns, as do Reoccur-

7.3 Interpretability

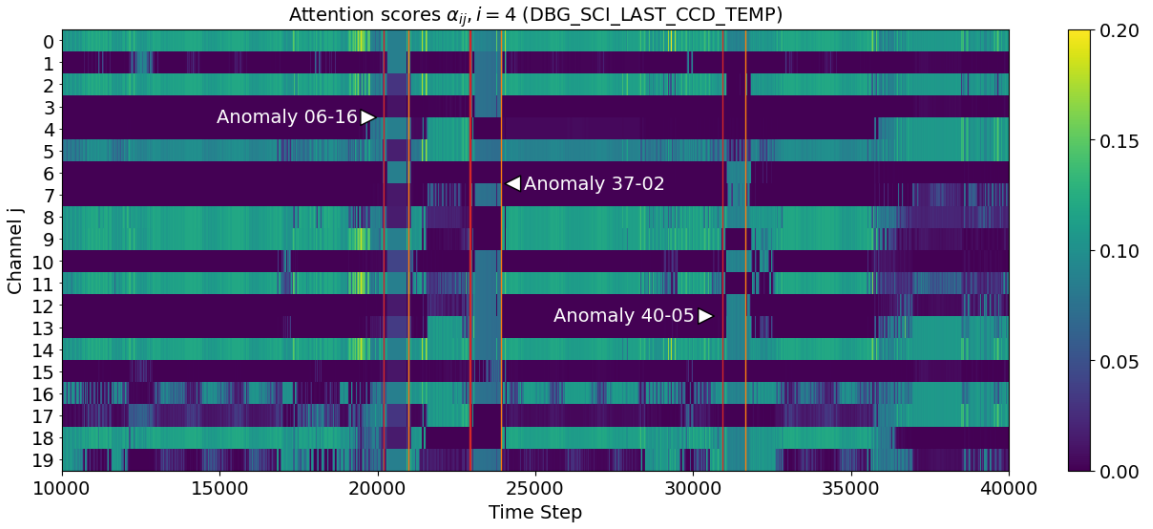

Figure 7.5: Progression over time of attention scores α_{ij} for channel $i = 4$, *DBG_SCI_LAST_CCD_TEMP*, produced by the GAT-Single model. Each channel j is a channel within packet *DBG_SCI*. Red and orange lines indicate the start and end of three true anomalies.

rences 3 and 4. This is only one example of a reoccurring anomaly based on the attention scores of a single channel, but such patterns may provide additional clues for the purposes of identifying anomalies.

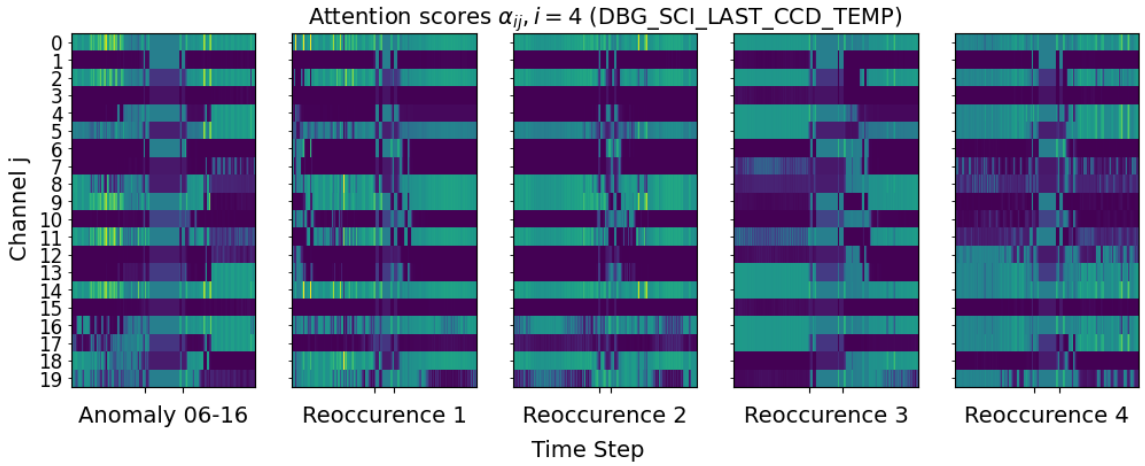

Figure 7.6: Attention scores α_{ij} for Anomaly 06-16 and its next four reoccurrences. The x-axis ticks are the start and end of each anomaly instance. This figure shares the same colour bar as Figure 7.5.

8

Conclusion and Future Work

In this work we have explored methods for detecting spacecraft anomalies from telemetry. Our focus has been on deep learning approaches, driven by the size and complexity of the data. Additionally, due to the limited availability of anomaly labels in operational settings, we favour semi-supervised rather than supervised techniques. We choose to apply anomaly score prediction models and thresholding techniques proposed by Hundman et al. (2018), Geiger et al. (2020), Zhao et al. (2020), and Siffer et al. (2017). Our model variations, listed in Table 8.1, cover different ways to construct the input data (e.g., incorporating command information and learning form multiple channels together), types of outputs (e.g., univariate and multivariate predictions), and types of prediction models (e.g., forecasting- and reconstruction-based approaches). These models leverage trending concepts in deep learning research like GANs, graph networks, and attention mechanisms; if we follow this trend, there are plenty of opportunities for novel anomaly detection methods as new, effective technologies emerge in related domains.

Through our experiments we have determined that models built with gated recurrent units (GRUs; Cho, van Merrienboer, Bahdanau, et al., 2014) achieve the best overall F1 scores when combined with a thresholding technique known as Peaks-Over-Threshold (POT; Siffer et al., 2017) and an anomaly pruning process developed by Hundman et al. (2018). These models perform well when using multiple channels as inputs to make predictions for a single channel at a time. However, the type of training loss—forecasting error, reconstruction error, or a combination of both—does not appear to be a major factor.

Experiment	Network Description	Input	Output
LSTM-Cmd	LSTM forecasting	1 channel,commands	1 channel
LSTM-Solo	LSTM forecasting	1 channel	1 channel
GAN-Cmd	GAN,LSTM reconstruction	1 channel,commands	1 channel
Recon-Cmd	LSTM reconstruction	1 channel,commands	1 channel
GAN-Multi	GAN,LSTM reconstruction	multi-channel	multi-channel
GAT-Multi	GAT,GRU forecast,reconstruct	multi-channel	multi-channel
GAT-Single	GAT,GRU forecast,reconstruct	multi-channel	1 channel
GRU-Single	GRU forecast,reconstruct	multi-channel	1 channel
GRU-Forecast	GRU forecasting	multi-channel	1 channel
GRU-Recon	GRU reconstruction	multi-channel	1 channel
GRU-Solo	GRU forecast,reconstruct	1 channel	1 channel
GRU-NoConv	No initial convolution layer	multi-channel	1 channel

Table 8.1: Variations of deep learning models explored in our experiments.

While the addition of graph attention layers (Veličković et al., 2018) also did not noticeably improve or diminish the model's ability to detect anomalies, it did provide supplemental information in the form of attention scores which could be useful in diagnosing suspected anomalies. Ultimately, we recommend that this blend of GRU-based model with POT and pruning be applied for detecting anomalies from spacecraft telemetry.

There can be reluctance in adopting new technologies or systems in established space programs. To reduce risk and increase confidence in missions taking place in the unforgiving space environment, these programs rely on flight heritage and proven concepts of operations. Deep learning tools in particular have not yet gained momentum on the space segment, but more and more applications are being developed for ground processing of data collected from space. The system we are recommending is a recurrent neural network that learns to forecast or reconstruct nominal telemetry; such an idea is not terribly difficult to grasp and accept. Furthermore, we propose integrating these anomaly detection methods, not in a manner that disrupts the current flow of operations, but rather as a complementary tool to existing practices. The use of deep learning-based anomaly detection alongside human operators allows trust to be gained in such systems while adding an extra set of autonomous eyes to monitor spacecraft health. Our experiments show that such methods can

detect many of the same anomalies identified by humans while maintaining a manageable rate of false positive predictions. There is potential for these methods to reveal additional insight towards diagnosing anomalous behaviour and as well as detect instances that are overlooked by operators.

Looking at the future of space exploration, the trend is clear: humans are going to the Moon, Mars, and beyond. With each new and distant endeavour comes a different set of challenges; communication between Earth and spacecraft becomes severely limited, as do evacuation options in emergency scenarios. The anomaly detection methods of today are the foundations of the health monitoring systems which will one day be deployed onboard spacecraft to provide real time detection and recovery measures in deep space missions. These intelligent systems will be vital assets, relied upon to avoid catastrophic losses and increase the chances of mission success. We hope our work shines some light on the possibilities and potential of deep learning methods for deep space exploration.

8.1 Future Work

8.1.1 Method Improvements

There is no shortage of interest from the machine learning community in developing models which output scores of anomalousness on different types of data, in a variety of domains. In most cases there is still a need to set a threshold that classifies anomalous samples. Compared to the amount of methods proposed for predicting anomly scores, few dynamic and adaptive thresholding techniques currently exist (Blázquez-García et al., 2021). There is an opportunity to improve anomaly detection performance through the development of novel thresholding techniques. One early idea discussed by Martínez-Heras and Donati (2018) is a network that learns upper and lower thresholds of telemetry values by balancing two objectives: all values should fall within the predicted limits, and the limits should be as narrow as possible. However, this solution has the same weakness as handcrafted limits in that contextual and collective anomalies can still exist within these boundaries.

Another way to improve performance is by incorporating past, confirmed anomaly labels into the thresholding process. Hundman et al. (2018) suggest a mechanism to re-

classify predicted anomalies as nominal if a model produces too many false positives, by making adjustments to threshold parameters. There are also active learning methods which allow human experts to participate in model training by providing occasional feedback based on its predictions. In doing so, the model could learn to disregard less severe anomalous situations, such as the GPS resets mentioned in the false positive analysis (Section 7.2).

Although we do not look at ways to specifically detect collective anomalies, they are some of the more difficult cases to be spotted by humans. Zhao et al. (2020) mention that several channels can be considered together simply by taking the sum of their prediction errors at each time step to produce a global anomaly score, before applying a threshold. Since this does not prevent the scores of one channel from overshadowing another's, we suggest an alternative way that first normalizes the anomaly scores. By computing thresholds and normalized anomaly scores as we have done for the experiments in this work, we can then take a weighted sum across channels using the normalized scores as weights. At this point, a time step during which several channels exhibit anomalous behaviour but to a smaller degree results in a combined score that is significant enough to be predicted as a collective anomaly.

8.1.2 Dataset Improvements

We split our dataset into months and chose to train our models on a single month during which no anomalies are reported. This approach gives us plenty of training samples over a consecutive time period, but we realize seasonal effects are not captured this way. A model trained on nominal data outside of a satellite's eclipse season may not represent nominal behaviour that occurs during this eclipse season, when more time than usual is spent in Earth's shadow. One solution to this might be to split each month into train and test sets, and feed training samples into the model from diverse time periods rather than all from the same month.

Following the methodology of Hundman et al. (2018), we aggregate telemetry values into one-minute windows so that channels which are sampled at different frequencies on the spacecraft can be used together as inputs into our discrete-time models. This is done by

taking the average every minute, which has the downside of masking anomalous behaviour appearing as brief spikes within a window. An alternative approach is interpolating data so that every channel contains a value at each time step of the highest frequency channel, similar to how missing data might be handled. This also has a drawback in that interpolation generates synthetic values or behaviour that may not truly reflect the status of the spacecraft at those times (Tariq et al., 2019). Fusing multiple asynchronous streams of data is a problem often encountered by roboticists, who have over the years found ways to address this such as through continuous time representations. Such solutions in the robotics field could be borrowed to improve our own time series models in anomaly detection tasks.

Our experiments touch on only a small portion of the available data from NEOSSat, in terms of the number of months as well as the number of channels. Since we wish to study and compare multiple methods, training and evaluating on a much larger dataset is not feasible. Now that an initial, broad investigation has been conducted, we could select a single method to run on a complete dataset spanning many years and thousands of channels. Such an experiment might reveal additional signatures linked to existing anomalies, or even uncover anomalies not previously detected by humans. Significant effort would be required from the satellite operations team to review and confirm the predicted anomalies. In a similar vein, a model might benefit from manual selection of particular channels to keep and discard in a dataset. We have thus far simply relied on the ability of deep neural networks to learn for themselves which inputs are pertinent for their predictions.

8.1.3 Further Research

In our current spacecraft anomaly detection setting, present and past (also called back orbit) telemetry is only processed during a ground pass. Therefore, anomalies can be flagged in near real-time or at previous time steps. Predicting anomalies that may occur in the future is a task that has not seen as much research attention but is in line with our goal of detecting anomalous activity as early as possible. One way to do this is by leveraging longer-term time series forecasting methods to predict future spacecraft behaviour, and then performing our usual anomaly detection on those forecasts (O'Meara et al., 2018). Petkovic et al. (2019) propose ensemble machine learning methods for predicting the evolution of thermal power consumption on the Mars Express Orbiter, although numerous models have

been developed in other fields to tackle this problem (Lim and Zohren, 2021; Sezer et al., 2020). If we are to predict impending anomalies, we must first gain confidence in our ability to detect them from actual data, which motivates the experiments we have conducted.

Bibliography

Abbasi, V., Jackson, N., Doyon, M., Wessels, R., Sekhavat, P., Cannata, M., Gillett, R., & Eagleson, S. (2019). Resurrecting NEOSSat: How innovative flight software saved canada's space telescope. In H. Pasquier, C. A. Cruzen, M. Schmidhuber, & Y. H. Lee (Eds.), *Space operations: Inspiring humankind's future* (pp. 589–613). Springer International Publishing. https://doi.org/10.1007/978-3-030-11536-4_23

Aggarwal, C. C. (2013). An introduction to outlier analysis. *Outlier analysis* (pp. 1–40). Springer New York. https://doi.org/10.1007/978-1-4614-6396-2_1

Ahn, H., Jung, D., & Choi, H.-L. (2020). Deep generative models-based anomaly detection for spacecraft control systems. *Sensors*, *20*(7), 1991. https://doi.org/10.3390/s20071991

Akcay, S., Atapour-Abarghouei, A., & Breckon, T. P. (2018). GANomaly: Semi-supervised anomaly detection via adversarial training. *arXiv:1805.06725 [cs]*. Retrieved September 13, 2020, from http://arxiv.org/abs/1805.06725

Akoglu, L., Tong, H., & Koutra, D. (2014). Graph-based anomaly detection and description: A survey. *arXiv:1404.4679 [cs]*. Retrieved February 5, 2021, from http://arxiv.org/abs/1404.4679

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein GAN. Retrieved December 15, 2021, from https://arxiv.org/abs/1701.07875v3

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baker, D. (2000). The occurrence of operational anomalies in spacecraft and their relationship to space weather. *IEEE Transactions on Plasma Science*, *28*(6), 2007–2016. https://doi.org/10.1109/27.902228

Balkema, A. A., & de Haan, L. (1974). Residual life time at great age. *The Annals of Probability*, *2*(5). https://doi.org/10.1214/aop/1176996548

Bayer, J., & Osendorfer, C. (2015). Learning stochastic recurrent networks. *arXiv:1411.7610 [cs, stat]*. Retrieved February 1, 2022, from http://arxiv.org/abs/1411.7610

Beirlant, J., Goegebeur, Y., Segers, J., & Teugels, J. L. (2004). *Statistics of extremes: Theory and applications* (Vol. 558). John Wiley & Sons.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. https://doi.org/10.1109/72.279181

Berndt, D. J., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. [Issue: 16]. *KDD workshop*, *10*, 359–370.

Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, *54*(3), 1–33. https://doi.org/10.1145/3444690

Bontemps, L., Cao, V. L., McDermott, J., & Le-Khac, N.-A. (2016). Collective anomaly detection based on long short-term memory recurrent neural networks. *International conference on future data and security engineering*, 141–152.

Boumghar, R., Venkataswaran, A., Brown, H., & Alvarez, X. C. (2021). Behaviour-based anomaly detection in spacecraft using deep learning [Publisher: Unpublished]. https://doi.org/10.13140/RG.2.2.23275.46881

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. *ACM SIGMOD Record*, *29*(2), 93–104. https://doi.org/10.1145/335191.335388

Bruckstein, A. M., Donoho, D. L., & Elad, M. (2009). From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, *51*(1), 34–81. https://doi.org/10.1137/060657704

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). Spectral networks and locally connected networks on graphs. *arXiv:1312.6203 [cs]*. Retrieved March 6, 2022, from http://arxiv.org/abs/1312.6203

Cahuantzi, R., Chen, X., & Güttel, S. (2021). A comparison of LSTM and GRU networks for learning symbolic sequences. *arXiv:2107.02248 [cs]*. Retrieved March 28, 2022, from http://arxiv.org/abs/2107.02248

Canada. (2019, February 28). *Historic investments in canada's space program to create jobs and new industries | prime minister of canada* [Office of the prime minister].

Retrieved February 15, 2022, from https://pm.gc.ca/en/news/news-releases/2019/02/28/historic-investments-canadas-space-program-create-jobs-and-new

Cao, J., Li, Z., & Li, J. (2019). Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical Mechanics and its Applications*, *519*, 127–139. https://doi.org/10.1016/j.physa.2018.11.061

Carlton, A., Morgan, R., Lohmeyer, W., & Cahoy, K. (2018). Telemetry fault-detection algorithms: Applications for spacecraft monitoring and space environment sensing. *Journal of Aerospace Information Systems*, *15*(5), 239–252. https://doi.org/10.2514/1.I010587

Castelvecchi, D. (2016). Can we open the black box of AI? *Nature News*, *538*(7623), 20.

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. *arXiv:1901.03407 [cs, stat]*. Retrieved September 4, 2020, from http://arxiv.org/abs/1901.03407

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, *41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Chauhan, S., & Vig, L. (2015). Anomaly detection in ECG time signals via deep long short-term memory networks. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 1–7. https://doi.org/10.1109/DSAA.2015.7344872

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., et al. (2015). Xgboost: Extreme gradient boosting. *R package version 0.4-2*, *1*(4), 1–4.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv:1601.06733 [cs]*. Retrieved January 6, 2022, from http://arxiv.org/abs/1601.06733

Chimmula, V. K. R., & Zhang, L. (2020). Time series forecasting of COVID-19 transmission in canada using LSTM networks. *Chaos, Solitons & Fractals*, *135*, 109864. https://doi.org/10.1016/j.chaos.2020.109864

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259 [cs, stat]*. Retrieved February 1, 2022, from http://arxiv.org/abs/1409.1259

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder

for statistical machine translation. *arXiv:1406.1078 [cs, stat]*. Retrieved March 4, 2021, from http://arxiv.org/abs/1406.1078

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555 [cs]*. Retrieved March 6, 2022, from http://arxiv.org/abs/1412.3555

David, F. N., & Tukey, J. W. (1977). Exploratory data analysis. *Biometrics*, *33*(4), 768. https://doi.org/10.2307/2529486

Davison, A. C. (1984). Modelling excesses over high thresholds, with an application. In J. T. de Oliveira (Ed.), *Statistical extremes and applications* (pp. 461–482). Springer Netherlands. https://doi.org/10.1007/978-94-017-3069-3_34

Deng, A., & Hooi, B. (2021). Graph neural network-based anomaly detection in multivariate time series. *arXiv:2106.06947 [cs]*. Retrieved September 15, 2021, from http://arxiv.org/abs/2106.06947

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise [event-place: Portland, Oregon]. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 226–231.

Fennell, J., Koons, H., Roeder, J., & Blake, J. (2001). Spacecraft charging: Observations and relationship to satellite anomalies, 25.

Fleetwood, D. M., Winokur, P. S., & Dodd, P. E. (2000). An overview of radiation effects on electronics in the space telecommunications environment. *Microelectronics Reliability*, *40*(1), 17–26. https://doi.org/10.1016/S0026-2714(99)00225-5

Fuertes, S., Picart, G., Tourneret, J.-Y., Chaari, L., Ferrari, A., & Richard, C. (2016). Improving spacecraft health monitoring with automatic anomaly detection techniques. *SpaceOps 2016 Conference*. https://doi.org/10.2514/6.2016-2430

Fuertes, S., Pilastre, B., & D'Escrivan, S. (2018). Performance assessment of NOSTRADAMUS & other machine learning-based telemetry monitoring systems on a spacecraft anomalies database. *2018 SpaceOps Conference*. https://doi.org/10.2514/6.2018-2559

Fujimaki, R., Yairi, T., & Machida, K. (2005). An approach to spacecraft anomaly detection problem using kernel feature space. *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 401–410. https://doi.org/10.1145/1081870.1081917

Galvan, D. A., Hemenway, B., Welser, W., & Baiocchi, D. (2014). Satellite anomalies. *Satellite anomalies: Benefits of a centralized anomaly database and methods for securely sharing information among satellite operators* (pp. 7–28). RAND Corporation. http://www.jstor.org/stable/10.7249/j.ctt14bs1m1.9

Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A., & Veeramachaneni, K. (2020). TadGAN: Time series anomaly detection using generative adversarial networks. *arXiv:2009.07769 [cs, stat]*. Retrieved September 19, 2021, from http://arxiv.org/abs/2009.07769

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *arXiv:1406.2661 [cs, stat]*. Retrieved December 11, 2021, from http://arxiv.org/abs/1406.2661

Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, *2*, 729–734. https://doi.org/10.1109/IJCNN.2005.1555942

Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks* (Vol. 385). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24797-2

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *arXiv:1303.5778 [cs]*. Retrieved December 1, 2021, from http://arxiv.org/abs/1303.5778

Grimshaw, S. D. (1993). Computing maximum likelihood estimates for the generalized pareto distribution [Publisher: Taylor & Francis]. *Technometrics*, *35*(2), 185–191.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein GANs. Retrieved December 15, 2021, from https://arxiv.org/abs/1704.00028v3

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Van Pottelbergh, T., Pasieka, M., Skrodzki, A., Huguenin, N., Dumonal, M., Kościsz, J., Bader, D., Gusset, F., Benheddi, M., Williamson, C., Kosinski, M., Petrik, M., & Grosch, G. (2021). Darts: User-friendly modern machine learning for time series. *arXiv:2110.03224 [cs, stat]*. Retrieved December 4, 2021, from http://arxiv.org/abs/2110.03224

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, *91*(1).

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huber, P. J. (2011). Robust statistics. In M. Lovric (Ed.), *International encyclopedia of statistical science* (pp. 1248–1251). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04898-2_594

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., & Soderstrom, T. (2018). Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 387–395. https://doi.org/10.1145/3219819.3219845

Hunter, J. S. (1986). The exponentially weighted moving average [Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00224065.1986.11979014]. *Journal of Quality Technology*, *18*(4), 203–210. https://doi.org/10.1080/00224065.1986.11979014

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. *International conference on machine learning*, 2342–2350.

Jung, D., Kwon, J. W., Baek, K., & Ahn, H. W. (2019). Attitude control simulator for the korea pathfinder lunar orbiter [Series Title: Lecture Notes in Electrical Engineering]. In X. Zhang (Ed.), *The proceedings of the 2018 asia-pacific international symposium on aerospace technology (APISAT 2018)* (pp. 2521–2532). Springer Singapore. https://doi.org/10.1007/978-981-13-3305-7_202

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems [_eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf]. *Journal of Basic Engineering*, *82*(1), 35–45. https://doi.org/10.1115/1.3662552

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. *arXiv:1312.6114 [cs, stat]*. Retrieved March 4, 2021, from http://arxiv.org/abs/1312.6114

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907 [cs, stat]*. Retrieved February 23, 2021, from http://arxiv.org/abs/1609.02907

Kirsch, M. (2012). Cage instability of XMM-newton's reaction wheels discovered during the development of an early degradation warning system. *SpaceOps 2012 Conference*. https://doi.org/10.2514/6.2012-1275204

Kitagawa, G., & Gersch, W. (1996). Linear gaussian state space modeling [Series Title: Lecture Notes in Statistics]. In P. Bickel, P. Diggle, S. Fienberg, K. Krickeberg, I. Olkin, N. Wermuth, & S. Zeger (**typeredactors**), *Smoothness priors analysis of time series* (pp. 55–65). Springer New York. https://doi.org/10.1007/978-1-4612-0761-0_5

Kriegel, H.-P., Kröger, P., Schubert, E., & Zimek, A. (2009). LoOP: Local outlier probabilities. *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*, 1649. https://doi.org/10.1145/1645953.1646195

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning [Number: 7553 Publisher: Nature Publishing Group]. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lee, W.-H., Ortiz, J., Ko, B., & Lee, R. (2018). Time series segmentation through automatic feature learning. *arXiv:1801.05394 [cs, stat]*. Retrieved January 25, 2022, from http://arxiv.org/abs/1801.05394

Li, D., Chen, D., Shi, L., Jin, B., Goh, J., & Ng, S.-K. (2019). MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. *arXiv:1901.04997 [cs, stat]*. Retrieved September 21, 2021, from http://arxiv.org/abs/1901.04997

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194), 20200209. https://doi.org/10.1098/rsta.2020.0209

Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models [Issue: 1]. *Proc. icml*, *30*, 3.

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. *Proceedings*, *89*, 89–94.

Martínez-Heras, J.-A. (2012). New telemetry monitoring paradigm with novelty detection. *SpaceOps 2012 Conference*. https://doi.org/10.2514/6.2012-1275123

Martínez-Heras, J.-A., & Donati, A. (2018). Novelty detection with deep learning. *2018 SpaceOps Conference*. https://doi.org/10.2514/6.2018-2560

Morgan, P. S. (2005). Fault protection techniques in JPL spacecraft [Publisher: Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space . . .].

NASA. (2019, June 4). *Artemis program* [NASA]. Retrieved February 15, 2022, from http://www.nasa.gov/artemisprogram

Olive, X. (2012). FDI(r) for satellites: How to deal with high availability and robustness in the space domain? *International Journal of Applied Mathematics and Computer Science*, *22*(1), 99–107. https://doi.org/10.2478/v10006-012-0007-8

O'Meara, C., Schlag, L., & Wickler, M. (2018). Applications of deep learning neural networks to satellite telemetry monitoring. *15th International Conference on Space Operations*. https://doi.org/10.2514/6.2018-2558

Pang, G., Shen, C., Cao, L., & Hengel, A. v. d. (2020). Deep learning for anomaly detection: A review. https://doi.org/10.1145/3439950

Park, D., Erickson, Z., Bhattacharjee, T., & Kemp, C. C. (2016). Multimodal execution monitoring for anomaly detection during robot manipulation. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 407–414. https://doi.org/10.1109/ICRA.2016.7487160

Park, D., Hoshi, Y., & Kemp, C. C. (2017). A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *arXiv:1711.00614 [cs]*. Retrieved October 1, 2021, from http://arxiv.org/abs/1711.00614

Pavón-Carrasco, F. J., & De Santis, A. (2016). The south atlantic anomaly: The key for a possible geomagnetic reversal. *Frontiers in Earth Science*, *4*. https://doi.org/10.3389/feart.2016.00040

Petkovic, M., Simidjievski, N., Boumghar, R., Breskvar, M., Dzeroski, S., Kocev, D., Levatic, J., Lucas, L., Osojnik, A., & Zenko, B. (2019). Machine learning for predicting thermal power consumption of the mars express spacecraft. *IEEE Aerospace and Electronic Systems Magazine*, *34*(7), 46–60. https://doi.org/10.1109/MAES.2019.2915456

Pilastre, B., Boussouf, L., D'Escrivan, S., & Tourneret, J.-Y. (2020). Anomaly detection in mixed telemetry data using a sparse representation and dictionary learning. *Signal Processing*, *168*, 107320. https://doi.org/10.1016/j.sigpro.2019.107320

Pilastre, B., Silva, G., Boussouf, L., d'Escrivan, S., Rodriguez, P., & Tourneret, J.-Y. (2020). Anomaly detection in mixed time-series using a convolutional sparse representation with application to spacecraft health monitoring. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3242–3246. https://doi.org/10.1109/ICASSP40776.2020.9053929

Pilastre, B., Tourneret, J.-Y., Boussouf, L., & D'escrivan, S. (2021). Spacecraft health monitoring using a weighted sparse decomposition [Series Title: Lecture Notes in Mechanical Engineering]. In L. Gelman, N. Martin, A. A. Malcolm, & C. K. (Edmund) Liew (Eds.), *Advances in condition monitoring and structural health monitoring* (pp. 159–168). Springer Singapore. https://doi.org/10.1007/978-981-15-9199-0_15

Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, *99*, 215–249. https://doi.org/10.1016/j.sigpro.2013.12.026

Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. *International conference on machine learning*, 1530–1538.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models [event-place: Beijing, China]. *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, II–1278–II–1286.

Robinson, A., & Fallside, F. (1987). *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering Cambridge.

Rolnick, D., Veit, A., Belongie, S., & Shavit, N. (2018). Deep learning is robust to massive label noise. *arXiv:1705.10694 [cs]*. Retrieved February 15, 2022, from http://arxiv.org/abs/1705.10694

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., & Muller, K.-R. (2021). A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, *109*(5), 756–795. https://doi.org/10.1109/JPROC.2021.3052449

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K.-R., & Kloft, M. (2020). Deep semi-supervised anomaly detection. *arXiv:1906.02694 [cs, stat]*. Retrieved September 21, 2021, from http://arxiv.org/abs/1906.02694

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a, January 3). Learning internal representations by error propagation. *Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1: Foundations* (pp. 318–362). MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Sagheer, A., & Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing*, *323*, 203–213. https://doi.org/10.1016/j.neucom.2018.09.082

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *26*(1), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

Scarselli, F., Gori, M., Ah Chung Tsoi, Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1319. https://doi.org/10.1162/089976698300017467

Schölkopf, B., & Smola, A. J. (2003). A short introduction to learning with kernels. *Advanced lectures on machine learning* (pp. 41–64). Springer.

Schölkopf, B., Williamson, R. C., Smola, A., Shawe-Taylor, J., & Platt, J. (1999). Support vector method for novelty detection. *Advances in neural information processing systems*, *12*.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks [Publisher: Ieee]. *IEEE transactions on Signal Processing*, *45*(11), 2673–2681.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, 106181. https://doi.org/10.1016/j.asoc.2020.106181

Shatkay, H., & Zdonik, S. (1996). Approximate queries and representations for large data sequences. *Proceedings of the Twelfth International Conference on Data Engineering*, 536–545. https://doi.org/10.1109/ICDE.1996.492204

Shipmon, D. T., Gurevitch, J. M., Piselli, P. M., & Edwards, S. T. (2017). Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. *arXiv:1708.03665 [cs, stat]*. Retrieved December 1, 2021, from http://arxiv.org/abs/1708.03665

Siffer, A., Fouque, P.-A., Termier, A., & Largouet, C. (2017). Anomaly detection in streams with extreme value theory. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1067–1075. https://doi.org/10.1145/3097983.3098144

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms [event-place: Lake Tahoe, Nevada]. *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, 2951–2959.

Space Telescope Science Institute. (2012, December). *Hubble space telescope primer for cycle 21* [The HST Primer for Cycle 21 was edited by Shireen Gonzaga, with Senior Technical Editor Susan Rose and contributions from many others at STScI, in particular Alessandra Aloisi, Azalee Bostroem, Stefano Casertano, Ronald Downes, Rosa Diaz, Linda Dressel, Andrew Fox, Dorothy Fraquelli, Andrew Fruchter, Warren Hack, Svea Hernandez, Jason Kalirai, Deborah Kenny, Tim Kimball, Matt Lallo, John MacKenty, Cristina Oliveira, Larry Petro, Charles Proffitt, Tony Roman, Ken Sembach, Linda Smith, Denise Taylor, Leonardo Ubeda, William Workman, and Jim Younger.]. Space Telescope Science Institute. https://www.stsci.edu/itt/review/cp_primer_cy21/cp_primer/primer.pdf

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., & Pei, D. (2019). Robust anomaly detection for multivariate time series through stochastic recurrent neural network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2828–2837. https://doi.org/10.1145/3292500.3330672

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *arXiv:1409.3215 [cs]*. Retrieved December 1, 2021, from http://arxiv.org/abs/1409.3215

Tariq, S., Lee, S., Shin, Y., Lee, M. S., Jung, O., Chung, D., & Woo, S. S. (2019). Detecting anomalies in space using multivariate convolutional LSTM with mixtures of probabilistic PCA. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2123–2133. https://doi.org/10.1145/3292500.3330776

Tipping, M. E., & Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers [Publisher: MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . .]. *Neural computation*, *11*(2), 443–482.

Trollope, E., Dyer, R., Francisco, T., Miller, J., Pagan Griso, M., & Argemandy, A. (2018). Analysis of automated techniques for routine monitoring and contingency detection of in-flight LEO operations at EUMETSAT. *2018 SpaceOps Conference*. https://doi.org/10.2514/6.2018-2532

UNOOSA. (2022). *Online index of objects launched into outer space* [United nations office for outer space affairs]. Retrieved February 15, 2022, from https://www.unoosa.org/oosa/osoindex/search-ng.jspx?lf_id=

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *arXiv:1706.03762 [cs]*. Retrieved June 21, 2020, from http://arxiv.org/abs/1706.03762

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. *arXiv:1710.10903 [cs, stat]*. Retrieved March 4, 2021, from http://arxiv.org/abs/1710.10903

Verzola, I., Donati, A., Martinez, J., Schubert, M., & Somodi, L. (2016, May 13). Project sibyl: A novelty detection system for human spaceflight operations. *SpaceOps 2016 conference*. American Institute of Aeronautics; Astronautics. https://doi.org/10.2514/6.2016-2405

Villani, C. (2009). *Optimal transport: Old and new* (Vol. 338). Springer.

Vintsyuk, T. K. (1972). Speech discrimination by dynamic programming. *Cybernetics*, *4*(1), 52–57. https://doi.org/10.1007/BF01074755

Wander, A., & Förstner, R. (2013). *Innovative fault detection, isolation and recovery strategies on-board spacecraft: State of the art and research challenges*. Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV Bonn, Germany.

Wang, C., Viswanathan, K., Choudur, L., Talwar, V., Satterfield, W., & Schwan, K. (2011). Statistical techniques for online anomaly detection in data centers. *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, 385–392. https://doi.org/10.1109/INM.2011.5990537

Werbos, P. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560. https://doi.org/10.1109/5.58337

Yairi, T., Takeishi, N., Oda, T., Nakajima, Y., Nishimura, N., & Takata, N. (2017). A data-driven health monitoring method for satellite housekeeping data based on probabilistic clustering and dimensionality reduction. *IEEE Transactions on Aerospace and Electronic Systems*, *53*(3), 1384–1401. https://doi.org/10.1109/TAES.2017.2671247

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *arXiv:1708.02709 [cs]*. Retrieved December 1, 2021, from http://arxiv.org/abs/1708.02709

Zacchei, A., Fogliani, S., Maris, M., Popa, L., Lama, N., Turler, M., Rohlfs, R., Morisset, N., Malaspina, M., & Pasian, F. (2003). HouseKeeping and science telemetry: The case of planck/LFI.

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., & Chawla, N. V. (2018). A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. *arXiv:1811.08055 [cs, stat]*. Retrieved May 14, 2021, from http://arxiv.org/abs/1811.08055

Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., & Zhang, Q. (2020). Multivariate time-series anomaly detection via graph attention network. *arXiv:2009.02040 [cs, stat]*. Retrieved March 3, 2021, from http://arxiv.org/abs/2009.02040

Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., & Xu, B. (2016). Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *arXiv:1611.06639 [cs]*. Retrieved December 1, 2021, from http://arxiv.org/abs/1611.06639

Zhou, T., Krähenbühl, P., Aubry, M., Huang, Q., & Efros, A. A. (2016). Learning dense correspondence via 3d-guided cycle consistency [ISSN: 1063-6919]. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 117–126. https://doi.org/10.1109/CVPR.2016.20

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2242–2251. https://doi.org/10.1109/ICCV.2017.244

Zhuang, D., Zhang, X., Song, S. L., & Hooker, S. (2021). Randomness in neural network training: Characterizing the impact of tooling. *arXiv:2106.11872 [cs]*. Retrieved April 4, 2022, from http://arxiv.org/abs/2106.11872

Zimek, A., Schubert, E., & Kriegel, H.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, *5*(5), 363–387. https://doi.org/10.1002/sam.11161

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. https://openreview.net/pdf?id=BJJLHbb0-

A

Supplemental Material: Experiments

A.1 Running Time

The wall clock running times for training each model variation as well as calculating thresholds are provided in Table A.1. The times shown are per channel; for the multivariate output models GAN-Multi and GAT-Multi, their training times have been divided by the number of channels predicted. Model training and thresholding are both performed on one month of data, or approximately 44,000 time steps. The running times for setting fixed thresholds and applying pruning are negligible, requiring no more than 1 second per channel. The number of experiments, threshold parameters, and channels (344) is the main reason we perform only a single run of each experiment. There is room for optimization in the software implementations, which could enable more efficient experimentation.

If we consider these running times in an operational setting, we note that model training may take a significant amount of time, but the models are trained or retrained infrequently. To add to this, a single type of threshold and set of parameter values would be selected, and the thresholding times are not significant enough to prevent near real-time anomaly detection as telemetry becomes available on the ground.

Experiment	Time (s) Mean	SD
LSTM-Cmd	10	2
LSTM-Solo	8	2
GAN-Cmd	400	157
Recon-Cmd	42	32
GAN-Multi	77	26
GAT-Multi	26	8
GAT-Single	792	372
GRU-Single	433	246

Threshold	Parameters	Time (s) Mean	SD
Dynamic	$w_e = T/4$	23	3
Dynamic	$w_e = T/3$	21	3
Dynamic	$w_e = T/2$	13	2
POT	$q = 10^{-3}$,$u = 0.97$	57	96
POT	$q = 10^{-3}$,$u = 0.99$	29	80
POT	$q = 10^{-3}$,$u = 0.999$	1	1
POT	$q = 10^{-4}$,$u = 0.97$	70	111
POT	$q = 10^{-4}$,$u = 0.99$	39	89
POT	$q = 10^{-4}$,$u = 0.999$	6	11
POT	$q = 10^{-5}$,$u = 0.97$	79	118
POT	$q = 10^{-5}$,$u = 0.99$	47	94
POT	$q = 10^{-5}$,$u = 0.999$	8	133

Table A.1: Wall clock running times for training models on a single channel (left) and calculating thresholds for a single channel over one month of test data (right).

A.2 Hardware and Software Configuration

A.2.1 Hardware

We train and evaluate the anomaly detection methods on compute resources provided by the CSA. The hardware consists of an Intel Xeon Gold 6248 processor running at 2.50 GHz as well as an NVIDIA Grid RTX8000P-8Q with 7 gigabytes of dedicated graphics memory.

A.2.2 Software

All methods and experiments are implemented with Python 3.7.6 using the following main packages and versions:

- numba==0.54.0
- numpy==1.20.0
- pandas==1.2.5

- scipy==1.7.1
- torch==1.9.0

A.3 Telemanom Experiments

A.3.1 Implementation Details

Our implementations of the Telemanom (Hundman et al., 2018) anomaly detection method including the LSTM-based forecasting model and dynamic thresholding technique are adapted from the official repository of Telemanom's creators[1]. We convert their TensorFlow-based implementation to PyTorch.

A.3.2 Method Parameters

In addition to the configurations described in Section 4.3, we adopt the hyperparameters presented in Table A.2, which are the same as those used by Hundman et al. (2018) in their experiments on SMAP and MSL data.

Model Parameters	
Train batch size	64
Dropout	0.3
Input sequence length l_s	250

Table A.2: Supplementary model parameters used in our Telemanom experiments.

The choice of thresholding window sizes w_e and step size w_s follow the default values of TadGAN. We use a smoothing window size of $0.02w_e$ rather than the default size of $0.05w_e$ from Telemanom. The best parameters found for each thresholding method are described in Table A.3.

[1] https://github.com/khundman/telemanom

	Fixed		Dynamic		POT		
Experiment	p	w_e	p	w_e	p	q	u
LSTM-Cmd	0.2	0.5	0.7	0.25	0.3	10^{-4}	0.97
LSTM-Solo	0.2	0.5	0.2	0.5	0.3	10^{-3}	0.99

Table A.3: Thresholding parameters that achieved the best F1 scores for LSTM-Cmd and LSTM-Solo.

A.4 TadGAN Experiments

A.4.1 Implementation Details

We adapt code from the official implementation[2] as well as a community implementation[3] to perform our TadGAN (Geiger et al., 2020) experiments.

Modifications for Multivariate Output

The TadGAN method supports reconstruction of $N \geqslant 1$ channels, and although the official and unofficial implementations support multivariate inputs, neither allow for multivariate outputs. We make the following basic modifications to provide this capability:

- the last fully connected layer of the encoder outputs latent data of dimension N*20 which is then accepted by the CriticZ module;
- the first LSTM layer in the generator and the first Conv1D layer in the CriticX module accept inputs of size N;
- the last fully connected layer of the generator outputs N channels.

The CriticX module still outputs a single score for the entire sample, as a measure of realness of all channels together rather than scores for each channel individually.

[2] https://github.com/sintel-dev/Orion
[3] https://github.com/arunppsg/TadGAN

A.4.2 Method Parameters

Our model architectures and configurations are based on the original TadGAN implementation. Table A.4 presents experiment settings to supplement the details mentioned in Section 5.3.

Model Parameters	
Train batch size	64
Input sequence length L_s	100
Latent space dimension	N*20
Dropout	0.2
Gradient penalty loss weight λ	10
Forward consistency loss weight γ	10
Learning rate	5e-04

Table A.4: Supplementary model parameters used in our TadGAN experiments. N is the number of channels handled by the model.

The same thresholding window sizes w_e, step size w_s, and smoothing window as in the Telemanom experiments are used. The setup that achieves the highest F1 scores for GAN-Cmd, Recon-Cmd, and GAN-Multi are shown in Table A.5.

	Fixed		Dynamic		POT		
Experiment	p	w_e	p	w_e	p	q	u
GAN-Cmd	0.2	0.33	0.3	0.33	0.6	10^{-3}	0.99
Recon-Cmd	0.2	0.5	0.3	0.5	0.5	10^{-3}	0.97
GAN-Multi	0.2	0.5	0.4	0.25	0.3	10^{-3}	0.97

Table A.5: Thresholding parameters that achieved the best F1 scores for GAN-Cmd, Recon-Cmd, and GAN-Multi.

Hyperparameter Search

The GAN-Cmd and Recon-Cmd models use default settings from TadGAN for the number of hidden units in the encoder $\mathcal{E}$ and generator $\mathcal{G}$: 100/64, respectively. In the GAN-Multi model, we explore a few settings for each packet since they contain varying numbers and

types of channels: 100/64, 200/128, and 300/256. The settings that result in the lowest reconstruction error on the validation set are used.

A.5 MTAD-GAT Experiments

A.5.1 Implementation Details

Our experiments with MTAD-GAT (Zhao et al., 2020) are based on an unofficial model implementation[4]. We follow their lead in using a regular autoencoder with GRU layers in place of a VAE for reconstruction, as the original work does not sufficiently describe how the variational model is implemented. The POT thresholding algorithm (Siffer et al., 2017) we use comes from the official implementation[5].

A.5.2 Method Parameters

Table A.6 supplements the model configurations described in Section 6.3. In addition, the moving average window size d to calculate relative differences for POT thresholding is set to 1000, or roughly 2.5% of the size of the test data. This is the same proportion as used by Siffer et al. (2017) on their experiments with satellite sensor measurements. The range of values for $q \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ and $u \in \{0.97, 0.99, 0.999\}$ are inspired by the anomaly detection experiments conducted by Siffer et al. (2017) and Su et al. (2019). The top F1 scores are achieved with the configurations described in Table A.7 for each threshold type and experiment.

Model Parameters	
Train batch size	64
Dropout	0.2
Input sequence length T	100

Table A.6: Supplementary model parameters used in our MTAD-GAT experiments.

[4] https://github.com/ML4ITS/mtad-gat-pytorch
[5] https://github.com/Amossys-team/SPOT

	Fixed		Dynamic		POT		
Experiment	p	w_e	p	w_e	p	q	u
GAT-Multi	0.4	0.5	0.5	0.25	0.5	10^{-3}	0.97
GAT-Single	0.1	0.5	0.2	0.5	0.7	10^{-3}	0.999
GRU-Single	0.2	0.5	0.7	0.25	0.9	10^{-3}	0.999
GRU-Forecast	0.1	0.5	0.1	0.5	0.7	10^{-3}	0.999
GRU-Recon	0.5	0.5	0.3	0.5	0.9	10^{-3}	0.97
GRU-Solo	0.2	0.5	0.8	0.25	0.6	10^{-3}	0.999
GRU-NoConv	0.2	0.5	0.2	0.5	0.7	10^{-3}	0.99

Table A.7: Thresholding parameters that achieved the best F1 scores for GAT and GRU experiments.

Hyperparameter Search

Like in the GAN-Multi experiment, we explore additional values for the number of hidden units in the forecast and reconstruction networks of GAT-Multi and GAT-Single: 80, 150, and 300 units. For each packet type, the configuration that results in the smallest prediction error on the validation set are kept. The remaining GRU experiments, including ablations, follow the same setup as GAT-Single for these hidden units.

B

Supplemental Material: Discussion

B.1 Full Results

The precision, recall, and F1 scores of all experiments with dynamic and POT thresholds are shown in Table B.1.

	Dynamic Threshold			POT Threshold		
Experiment	Precision	Recall	F1	Precision	Recall	F1
LSTM-Cmd	0.58(0.31)	0.89(0.17)	**0.64**(0.27)	0.64(0.38)	0.61(0.28)	0.57(0.29)
LSTM-Solo	0.55(0.28)	0.91(0.17)	**0.63**(0.31)	0.62(0.39)	0.67(0.30)	0.60(0.35)
GAN-Cmd	0.60(0.36)	0.98(0.05)	0.66(0.37)	0.70(0.25)	0.94(0.11)	**0.76**(0.25)
Recon-Cmd	0.59(0.36)	0.88(0.15)	0.64(0.34)	0.58(0.23)	0.97(0.10)	**0.69**(0.25)
GAN-Multi	0.58(0.34)	1.00(0.00)	**0.67**(0.33)	0.55(0.23)	0.92(0.11)	0.65(0.24)
GAT-Multi	0.49(0.29)	0.92(0.20)	0.60(0.29)	0.63(0.30)	0.94(0.11)	**0.70**(0.27)
GAT-Single	0.57(0.36)	0.91(0.11)	0.59(0.36)	0.76(0.26)	0.97(0.10)	**0.81**(0.26)
GRU-Single	0.52(0.31)	0.81(0.21)	0.58(0.26)	0.83(0.26)	0.92(0.17)	**0.82**(0.25)
GRU-Forecast	0.49(0.32)	0.94(0.11)	0.56(0.35)	0.80(0.25)	0.91(0.13)	**0.81**(0.23)
GRU-Recon	0.58(0.30)	0.85(0.20)	0.62(0.31)	0.83(0.28)	0.83(0.31)	**0.83**(0.29)
GRU-Solo	0.59(0.32)	0.71(0.20)	0.59(0.22)	0.79(0.27)	0.81(0.30)	**0.79**(0.27)
GRU-NoConv	0.54(0.35)	0.83(0.19)	0.56(0.35)	0.80(0.28)	0.80(0.31)	**0.79**(0.27)

Table B.1: Mean(standard deviation) of scores from all experiments across all test datasets, with dynamic and POT thresholds.

B.2 Threshold Parameter Sensitivity

We explore the variance in scores achieved with different parameter values for dynamic and POT thresholding, in Figure B.1. In our experiments, only three options are selected for each of the parameters w_e, q, and u. The variance is generally low for both threshold types, and neither is consistently less sensitive to its parameters than the other.

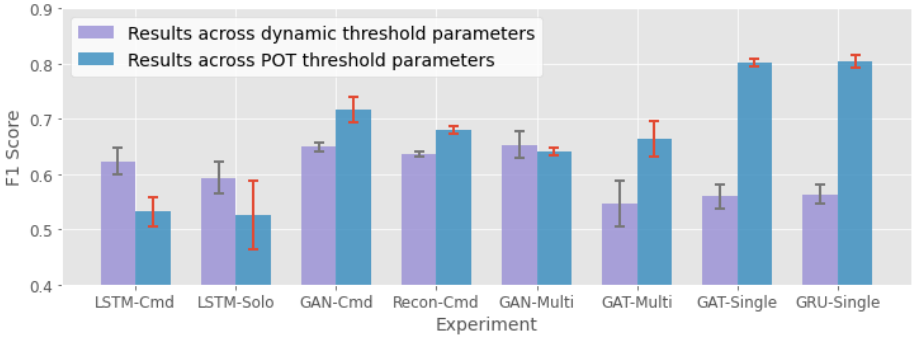

Figure B.1: Sensitivity of thresholding parameters, shown as error bars representing one standard deviation of F1 scores. For dynamic thresholding, we use window sizes $w_e \in \{\frac{T}{4}, \frac{T}{3}, \frac{T}{2}\}$. For POT the choices of parameters are $q \in \{10^{-3}, 10^{-4}, 10^{-5}\}$ and $u \in \{0.97, 0.99, 0.999\}$. The scores are obtained after anomaly pruning is applied.

B.3 False Positives Investigation

Ten cases of false positives are extracted from the results of the GAT-Single experiment after applying POT thresholding. As this is one of the top-performing methods, we present these cases to NEOSSat operators from CSA to confirm that they are actually false positives and to offer clues in interpreting the cause of their prediction. The abridged feedback from the satellite operations crew is listed in Table B.2. The cases labeled as *FP5-FP10* occur in March, a month that produces extraordinarily low precision scores for all methods.

B.4 Sorted Anomaly Scores

To assess whether the calculated anomaly scores represent some level of anomalousness of predicted anomaly sequences, we sort sequences by their maximum anomaly score and

ID	Year, Days	Comments
FP1	2018, 173-175	coincides with alarm triggered on *ROE_SCI_PREAMP_TEMP*; attributed to noise in telemetry, so not reported as anomaly.
FP2	2018, 251-253	coincides with GPS2 reset; does not explain non-GPS related indicators in telemetry.
FP3	2018, 284	due to recovery operations for earlier anomaly (SAF 60-01).
FP4	2018, 334	due to recovery operations for earlier anomaly (SAF 25-02).
FP5	2019, 065	cause is unclear.
FP6	2019, 070-071	cause is unclear.
FP7	2019, 073-074	GPS1 reset, not significant enough to be reported as anomaly.
FP8	2019, 078-079	GPS1 reset, not significant enough to be reported as anomaly.
FP9	2019, 084	cause is unclear.
FP10	2019, 086-088	GPS1 reset, not significant enough to be reported as anomaly.

Table B.2: Explanations and time periods for the ten false positive cases selected for further investigation.

visualize the distribution of true positive and false positive results. We first combine the results of each channel in order to look at overall predicted sequences instead of individual channels' results. To do this for true positives, we find the maximum normalized anomaly score that overlaps a true anomaly sequence. There is no straightforward way to combine the false positive predictions across channels so we again rely on overlapping sequences: we look at all the time steps that are considered false predictions by any channel, and take contiguous sequences of these false positive time steps as overall false positive sequences. The score associated with these sequences is the maximum normalized anomaly score found in the time steps of the sequence. This is done for each model using dynamic and POT thresholding after pruning (Figure B.2).

B.5 Results by Month

The precision, recall, and F1 scores of each model using dynamic and POT thresholding after pruning are displayed in Figure B.3 for each month of test data. We mostly only consider F1 score in our analysis of the results since it balances the other two types of metrics, but it can be helpful to understand how each metric differs over the ten months. Additionally, these plots allow us to observe trends such as particularly low scoring months.

B.5 Results by Month

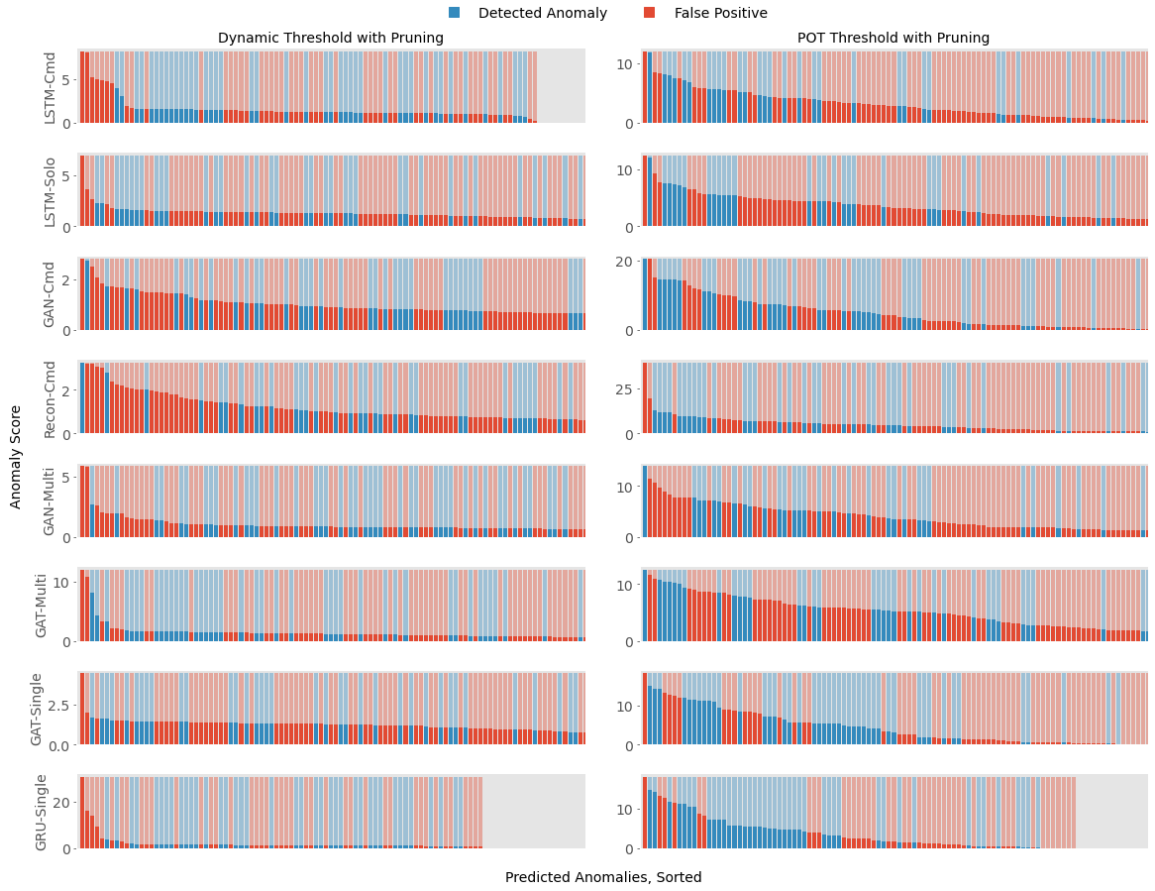

Figure B.2: Predicted anomaly sequences sorted by anomaly score with true positive and false positive labels, for all models. The top 100 scores are shown.

Finally, the total number of predicted anomalies of each method is presented since high precision methods can still have a large amount of false positives if there is an even greater quantity of true positive results.

B.5 Results by Month

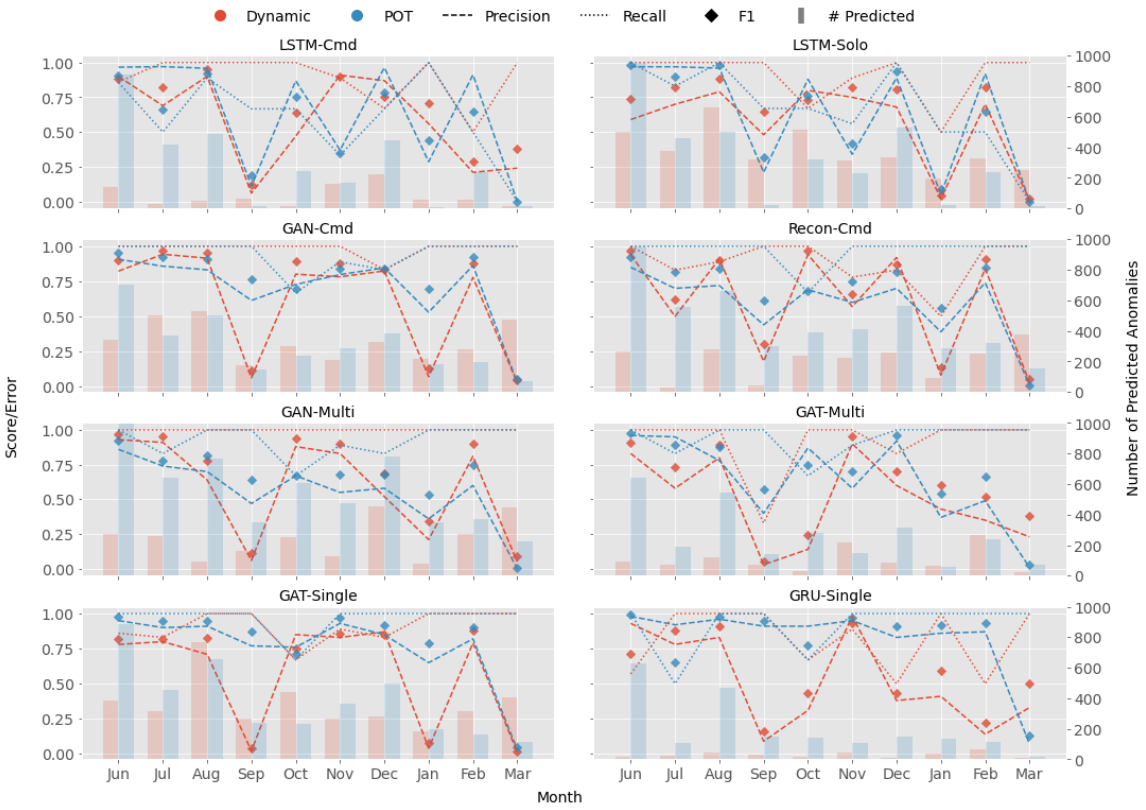

Figure B.3: Precision, recall, and F1 scores for each model by month of test data. The total number of predicted sequences is plotted in the background.