Spring 5-11-2024

Beyond the Horizon: Exploring Anomaly Detection Potentials with Federated Learning and Hybrid Transformers in Spacecraft Telemetry

JUAN RODRIGUEZ
Southern Methodist University, juanjose@smu.edu

Beyond the Horizon: Exploring Anomaly Detection Potentials with Federated Learning and Hybrid Transformers in Spacecraft Telemetry

Approved by:

Beyond the Horizon: Exploring Anomaly Detection Potentials with Federated Learning and Hybrid Transformers in Spacecraft Telemetry

A Praxis Presented to the Graduate Faculty of the

Lyle School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Doctor of Engineering

with a

Major in Software Engineering

by

Juan Jose Rodriguez

M.S.E., Software Engineering, Pennsylvania State University
B.S., Applied Math, Arizona State University

April 23, 2024

ACKNOWLEDGMENTS

I extend heartfelt gratitude to my wife Jenna, children Danielle and Joaquin, my parents Sergio and Ana Rodriguez, and my mother-in-law Patricia for their unwavering support during my dissertation journey. Jenna's encouragement and sacrifices have been invaluable, as have my children's patience and understanding. I am thankful to my parents for instilling in me the value of education and perseverance, which motivates me daily. Patricia's support has been instrumental in facilitating my pursuit of this goal, and I am deeply grateful for her contributions. Additionally, I appreciate the mentorship and guidance of my advisor, Dr. Frank Coyle, whose support has been pivotal in my academic growth. This dissertation is dedicated to my family, Patricia, and Dr. Coyle, whose unwavering belief in me made this achievement possible. Thank you for standing by me every step of the way.

Rodriguez, Juan M.S.E., Software Engineering, Pennsylvania State University, 2017
B.S., Applied Math, Arizona State University, 2015

Beyond the Horizon: Exploring Anomaly Detection Potentials with Federated Learning and Hybrid Transformers in Spacecraft Telemetry

Advisor: Professor Frank Coyle

Doctor of Engineering conferred May 11, 2024

Praxis completed May 10, 2024

Telemetry sensors play a crucial role in spacecraft operations, providing essential data on efficiency, sustainability, and safety. However, identifying irregularities in telemetry data can be a time-consuming process that risks the success of missions. With the rise of CubeSats and smallsats, telemetry data has become more abundant, but concerns about privacy and scalability have resulted in untapped data potential. To address these issues, we propose a new approach to anomaly detection that utilizes machine learning models at data sources. These models solely transmit weights to a centralized server for aggregation, resulting in improved dataset performance with a single global model. We have also incorporated self-attention into the federated process to further enhance anomaly detection performance. Our experiments with real-world telemetry data have demonstrated that our approach is state-of-the-art in that we can construct a single model to address multiple telemetry channels while still adhering to the constraints typically seen in space missions. Our framework streamlines anomaly detection, promoting operational efficiency, sustainability, and safety. It facilitates collaborative insights while abiding by mission security constraints and reducing the risk of accidents and downtime, ensuring sustainability.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: Introduction

An integral part of space missions is the ability of the spacecraft to communicate data streams of each onboard instrument. This enables mission operators to capture payload data and critical engineering data regarding the health of the spacecraft. This process is called telemetry monitoring. Once in orbit, the spacecraft constantly sends telemetry data back to Earth, which contains valuable information about the performance and status of the spacecraft's numerous systems and instruments.[20]. The volume and generation of this data is determined by mission complexity, duration, onboard instruments, and spacecraft automation [42]. Throughout the operational phase of a mission, the amount of telemetry data generated can vary depending on the type of instrument and how frequently measurements are taken. For instance, an instrument that measures the temperature of a spacecraft's battery might only produce a handful of data points each minute, while a camera capturing images of a planet could yield several gigabytes of data daily [8]. One critical activity for telemetry monitoring is to observe for anomalies referred to as anomaly detection. Mission planners and designers assume certain states of the spacecraft when in orbit and need to be notified when the spacecraft is in an aberrant state enabling operators to return the spacecraft to a nominal state through a series of corrective telecommands.

The history of spacecraft anomaly detection dates back to the early days of space exploration. In the early days of manned spaceflight, astronauts or ground-based personnel often detected anomalies and manually corrected them through teleoperation or other means [28]. However, the need for more advanced and automated anomaly detection techniques became apparent as spacecraft missions became more complex and sophisticated. One of the earliest examples of automated anomaly detection in spacecraft occurred during the Apollo 11 mission in 1969. The spacecraft's guidance computer encountered an error during the descent to the lunar surface, triggering an alarm. However, the astronauts were able to override the error and complete the landing successfully manually. After the mission, the error was traced to a faulty switch, and new procedures were implemented to prevent a recurrence of the problem. In the following decades, researchers continued developing and refining techniques for automated anomaly detection in spacecraft telemetry data. Early approaches focused on statistical methods, such as hypothesis testing and regression analysis, to detect anomalies in spacecraft data. However, these methods had limited accuracy and required extensive human involvement to interpret the results [49]. With the increase in the utilization of machine learning and data mining techniques seen in the 1990s, researchers began to explore these methods for anomaly detection in spacecraft telemetry data. For example, NASA's Jet Propulsion Laboratory researchers developed the Autonomous Sciencecraft Experiment, which used machine learning algorithms to detect and classify anomalies in Mars rover telemetry data [53]. Since then, research in spacecraft anomaly detection has continued to advance. New techniques and approaches have been developed to address the challenges of detecting anomalies in increasingly complex and diverse spacecraft systems.

1.1 Background

The ever-changing landscape of today's world has highlighted the need for adaptable and resilient systems that can function effectively in both expected and unforeseen circumstances. Deep learning (DL) or the subset of machine learning that uses multi-layered neural networks as function approximators, has emerged as a powerful technology that allows systems to learn rules from vast amounts of data [18]. This approach has proved to be more effective than relying solely on prescriptive rules defined by engineers across a wide range of domains [30].

The resurgence of AI(Artificial Intelligence) in recent years has been remarkable, and deep learning has played a significant role in its success, especially in tasks such as image recognition. Most DL methods assume a single machine will be dedicated to handling the computational task of fitting the model to the data available in the training process, referred to as Centralized Learning (CL) [16]. CL frameworks have been successful for technology companies with access to extensive cloud-based data. However, this approach has its limitations, particularly in situations where data cannot be easily transmitted to a central location due to factors such as limited bandwidth, connectivity issues, privacy concerns, and resource constraints. In healthcare, for example, strict privacy regulations make it challenging to pool data, while in IoT, transmitting raw data can be energy-intensive. Pursuing a centralized learning strategy in these scenarios can result in limited access to new data, leading to outdated decision-making criteria and a decline in system performance over time. Federated Learning emerges as a solution for such challenges. First introduced by [16], FL is defined as sub-field of machine learning focused on settings in which multiple entities collaboratively train a single model without the sharing of data but model parameters [16]. Unlike centralized learning, Federated Learning brings the model to the data rather than vice versa. It is particularly relevant in situations where transmitting raw on-device data is expensive or impractical. Federated Learning allows continuous training of intelligent systems post-production, addressing the limitations of centralized learning. It is especially useful in overcoming legal constraints in cases like the Health Insurance Portability and Accountability Act (HIPAA), where patient data aggregation for medical research is challenging. FL's ability to send model updates instead of bulk data minimizes data transmission and respects privacy concerns. Google's application of FL in its Android operating system showcases its potential benefits for cell phone users concerned about privacy, battery life, and data usage. By communicating only model updates, FL reduces bandwidth usage and enhances privacy, crucial considerations for both cell phones and IoT devices with limited resources [16].

In dynamic environments, users expect intelligent systems to adapt beyond the initial training dataset. FL, by enabling distributed model training on various devices simultaneously, allows systems to continually learn from local data during the Utilization & Support stages of the system lifecycle. The federated learning architecture proposed provides a framework for ongoing system training, overcoming the limitations of a traditional CL approach.

1.2 Federated Learning Basics

Federated learning is a machine learning approach that allows for model training across multiple devices or servers while keeping the data localized [16]. Unlike traditional machine learning methods where data is centralized in one location for training, federated learning enables model training directly on the devices where the data is generated or stored, such as smartphones, IoT devices, or edge servers [24]. This decentralized approach to training offers several advantages, including increased privacy, reduced communication costs, and the ability to leverage data that cannot be easily moved due to privacy concerns or regulatory restrictions.

In federated learning, the training process typically involves three main steps: local training, aggregation, and global model update. During the local training phase, each device or server trains the model using its local data without sharing the raw data. Once the local training is complete, the updated model parameters are sent to a central server or aggregator. The aggregator then combines these local updates to compute a global update for the model. This global update is then sent back to the devices for further refinement, and the process iterates until the model converges to an optimal state or a predefined stopping criterion is met [25].

One of the key benefits of federated learning is enhanced privacy protection. Since the raw data remains on the local devices and only model updates are shared, sensitive information is less exposed to potential breaches or unauthorized access. This makes federated learning particularly well-suited for applications where data privacy is a primary concern, such as healthcare, finance, and personalization services. Additionally, federated learning can also help reduce communication costs by minimizing the amount of data transferred between devices and central servers, making it more efficient for resource-constrained environments [57].

1.3 Research Motivation

The power of deep learning to enhance human understanding and automate spacecraft telemetry analysis is clear. Machine learning, especially in deep learning, can surpass human capabilities and identify subtle changes in complex data patterns. Automating the analysis of telemetry data not only improves operational efficiency but also reduces the cognitive load on personnel who monitor spacecraft data streams. Federated Learning (FL), a concept applicable to both spacecraft telemetry and the Internet of Things (IoT), presents an avenue for intelligent systems to continuously learn post-deployment in operational space environments. Diverging from Centralized Learning (CL), FL proves effective in settings characterized by limited network bandwidth and remote computing resources. This architectural approach is particularly suited for pre-existing, constrained spacecraft networks, bringing about efficiency gains and substantial cost savings by obviating the need to upgrade network capabilities. The diminished reliance on extensive telemetry downlinks not only enhances system adaptability across various operational environments, including deep space missions, but also fosters an overall increase in resilience.

Strategically implementing FL in telemetry systems allows mission planners to judiciously reallocate precious network bandwidth to other critical spacecraft operations. This strategic approach proves especially advantageous in scenarios involving both constrained networks and computationally limited edge devices, as commonly encountered in spacecraft missions.

However, achieving these objectives in space missions necessitates the establishment of a well-defined system architecture. This architecture serves as the cornerstone for designing FL components and understanding their dependencies on the network and available edge resources. A comprehensive grasp of tradeoff considerations in FL systems enables engineers, architects, and mission planners to effectively scope, plan, and execute telemetry projects. In both IoT and spacecraft telemetry applications, recognizing and navigating system limitations are integral to making informed system design decisions. A federated learning architecture adeptly captures these system dependencies, intricately linking them with the anomaly detection (AD) system's performance metrics. Tailoring the architecture to the specific operational environment facilitates continuous learning even under sub-optimal conditions, ensuring the system's maintainability and adaptability in the dynamic and challenging realm of space missions.

1.4 Thesis Statement

This research thesis explores the application of federated learning in spacecraft telemetry anomaly detection, with a primary focus on evaluating its effectiveness in achieving or surpassing state-of-the-art results in anomaly detection accuracy. Additionally, the study aims to explore how hybrid transformer architecture can be used in personalizing attention to each client's data without compromising global performance. By addressing these objectives, the thesis contributes to the advancement of anomaly detection systems *in the context of space exploration, offering insights into the practical implementation and optimization of federated learning techniques.*

1.5 Research Objectives

The objective of this research is to present a demonstrative application and blueprint for a Federated Learning (FL) platform specifically for the task of onboard spacecraft anomaly detection. The focus is primarily on developing a single DL model trained in the FL paradigm capable of detecting anomalies across a wide range of telemetry channels while also understanding the mission implications of our approach. Previous work in this domain has generally taken the approach of building multiple models to address the problem, one unique model for each data channel available. Furthermore, none of the work done before has considered mission implications of doing an onboard approach, thereby demanding a large centralized server to implement.

The unique constraints of these environments dictate a careful structuring of the FL system to ensure its practicality and longevity. The research outcomes will serve to validate the feasibility of employing FL to enhance the performance of telemetry anomaly detection. The findings will be instrumental in evaluating whether the proposed federated learning architecture outperforms centralized telemetry monitoring approaches. This comparative analysis is vital for establishing the efficacy and advantages of the proposed FL system. By providing a clear roadmap and addressing these key aspects, this research aims to contribute valuable insights to the broader field of FL, edge computing, and telemetry monitoring.

1.6 Research Questions and Hypothesis

The principal aim of this research is to evaluate whether Federated Learning (FL) can achieve performance levels comparable to established centralized techniques. Additionally, the study explores the feasibility of applying FL across missions, emphasizing spacecraft personalization instead of demanding maintainability requirements often associated with space missions. If successful, this approach could offer mission planners a more cost-effective means of developing intelligent and rapidly adaptive systems. The importance of spacecraft personalization in space missions is paramount, ensuring sustained functionality, adaptability, and reliability of mission-critical systems. A focus on effective personalization practices guarantees the continued success and safety of spacecraft operations, communication, and scientific data, contributing to the overall mission objectives. It's noteworthy that integrating spacecraft personalization requirements with FL performance is still an emerging area in the machine learning community.

The evaluation of updating machine learning models through a personalized system is conducted both conceptually and through simulation to assess its feasibility, marking a significant step in advancing the understanding of the coupling between personalization and FL performance in the context of space missions.

RQ1: *To what extent can a federated learning approach be effectively applied to spacecraft telemetry anomaly detection, aiming to achieve or surpass state-of-the-art results in anomaly detection accuracy?*

RQ2: *How can a hybrid transformer approach be strategically integrated into the federated anomaly detection system to optimize local performance for individual spacecraft,* *thereby advancing anomaly detection accuracy and contributing to the overall efficacy of the framework?*

H1: *By leveraging the decentralized nature of federated learning, spacecraft telemetry anomaly detection can benefit from the contributions of individual spacecraft components. The implementation of this approach is expected to yield comparable results to existing state-of-the-art methods, while enhancing the accuracy of anomaly detection. We hypothesize that this federated learning model will prove to be a robust and effective method for improving the accuracy of spacecraft telemetry anomaly detection, achieving comparable results to current approaches.*

H2: *By strategically incorporating a hybrid transformer approach into the existing federated anomaly detection system, we anticipate a significant boost in the performance of individual spacecraft, resulting in a marked improvement in anomaly detection accuracy. This innovative model is expected to exceed the baseline results obtained through standalone federated learning, and potentially even surpass the performance of state-of-the-art techniques. As such, it represents a potent and effective means of optimizing the overall efficacy of the federated framework for spacecraft telemetry anomaly detection.*

In the pursuit of addressing these inquiries, a set of best practices and sound assumptions surfaces, providing valuable guidance for system designs in the formulation of similar systems and model architectures.

1.7 Scope of Research

This research aims to evaluate the effectiveness of centralized and federated anomaly detection methods implemented through TensorFlow software. Particular emphasis will be placed on exploring diverse model architectures and potential strategies for adapting the system. Furthermore, a systematic approach will be formulated to optimize these methods within the framework of Federated Learning (FL). Considerations pertinent to mission planners will be thoroughly deliberated upon and documented for the deployment of such an architectural framework. Special attention will be dedicated to understanding the tight coupling of dependencies inherent in the deep learning (DL) model algorithms. The variability of these constraints will yield valuable insights into tailoring the proposed federated learning architecture to suit the unique demands of spacecraft missions. The results obtained will quantitatively assess the correlation between architectural modifications and the efficacy of telemetry anomaly detection, contributing to a deeper understanding of the interplay between personalization and anomaly detection methods within the context of space missions.

1.8 Limitations

While this research endeavors to make significant contributions to the field of spacecraft telemetry anomaly detection through the application of federated learning, certain limitations should be acknowledged. The primary constraint lies in the limited availability of data for training and testing purposes. The scarcity of diverse and comprehensive datasets specific to spacecraft telemetry anomalies may impact the generalizability of the proposed federated learning framework. Additionally, the effectiveness of machine learning models, including federated learning, heavily relies on the quality and quantity of data. The limited dataset may pose challenges in achieving robust and highly accurate anomaly detection models. The scarcity of real-world telemetry anomaly instances could potentially constrain the model's ability to adapt to a wide range of scenarios and variations that might be encountered in actual space missions.

Furthermore, the scope of this research may be influenced by the constraints imposed by the availability of resources, both in terms of computational power and time. These limitations may impact the depth and breadth of the experimentation and analysis conducted within the study. Despite these constraints, this research aims to provide valuable insights and establish a foundation for future work in the domain of spacecraft telemetry anomaly detection with federated learning. The findings, while recognizing these limitations, can still offer meaningful contributions and guide further exploration in this evolving field.

1.9 Organization of Praxis

This praxis is centered on a thorough exploration of the implementation of Federated Learning (FL) in the unique context of spacecraft missions. The initial phase involves a review of foundational research, with a specific emphasis on guiding principles for selecting model architectures suitable for FL in spacecraft telemetry. Subsequently, the focus shifts to the introduction of methods crafted to encompass various aspects of a personalized system within the FL framework, particularly through the exploration of hybrid transformers. The ensuing discussion addresses the inherent limitations associated with existing architectural designs, offering insights into how these constraints can be effectively addressed through strategic refactoring techniques.

As the praxis progresses, it transitions into delineating the practical methodologies involved in model training within a federated learning framework, considering the intricacies of spacecraft telemetry. The culmination of the praxis is dedicated to the assessment of anomaly detection (AD) accuracy within the FL model, utilizing relevant performance metrics. The conclusive section consolidates the outcomes of the praxis, including the exploration of hybrid transformers, architectural considerations, and implications to a system design.

Within this concluding framework, tailored recommendations are presented, specifically aimed at mission planners contemplating the integration of FL with hybrid transformers into their space systems. This offers a roadmap for informed decision-making in the dynamic and challenging realm of spacecraft missions.

CHAPTER 2: Previous Work

2.1 Machine Learning in Spacecraft Anomaly Detection

Recent research has witnessed the integration of deep learning techniques, these advancements have shown promise in capturing spatial and temporal dependencies and enhancing anomaly detection capabilities. Moreover, several state-of-the-art anomaly detection methods have emerged in recent years, each contributing to the diverse landscape of anomaly detection techniques. KitNet [32] is recognized as the pioneering work that utilizes auto-encoders, both with and without ensembles(using multiple ML methods), for online anomaly detection. OmniAnomaly [46] introduces a novel approach by employing a variational auto-encoder with gated recurrent units, specifically tailored for multivariate time series anomaly detection. GAN-Li [36] and MAD-GAN [37] present unsupervised multivariate anomaly detection methods based on Generative Adversarial Networks (GANs), leveraging LSTM-RNNs as both generator and discriminator models to capture the temporal correlations in time series distributions. Furthermore, LSTM-VAE [4] focuses on fusing signals and reconstructing their expected distribution using Variational Auto-Encoders (VAEs) with Long Short-Term Memory (LSTM) networks. LSTM-NDT [21] introduces a univariate time series detection method based on LSTMs, incorporating a novel nonparametric anomaly thresholding approach suitable for NASA datasets. DAGMM [17] employs a deep auto-encoder to generate an error sequence, which is then fed into a Gaussian Mixture Model (GMM) for anomaly detection. MTAD-GAT [39] takes a different approach by jointly optimizing a forecasting-based model and a reconstruction-based model to capture time relationships and variable dependencies. GTA [33] presents a novel framework for multivariate time series anomaly detection, involving the automatic learning of a graph structure, graph convolution, and modeling of temporal dependencies using a Transformer-based architecture. Moreover, while these state-of-the-art anomaly detection methods have made significant contributions to the field, it is worth noting that, to the best of our knowledge, the exploration of these advanced techniques within a federated learning domain, particularly in the context of spacecraft anomaly detection, remains relatively unexplored. As such, there is a notable gap in the literature regarding the application of these methods in federated settings, presenting an opportunity for future research to bridge this gap and explore the potential implications and challenges of deploying such techniques in federated environments, particularly in the context of space system design.

2.2 Federated Learning and its Applicability

Federated Learning (FL) is a burgeoning paradigm in machine learning that introduces a decentralized approach to model training. This section provides a literature review on Federated Learning and its applicability across various domains. A foundational work by [16] establishes the concept of Federated Learning, emphasizing its potential to address privacy concerns in distributed learning scenarios. The study introduces the idea of training models across decentralized devices, enabling learning without raw data leaving individual devices. This privacy-preserving aspect of FL has spurred its adoption in applications where data security and confidentiality are paramount. Further exploration of FL's applicability is found in [23], which investigates its effectiveness in scenarios with massive datasets distributed across different geographical locations. The study demonstrates how FL can significantly reduce communication costs and alleviate privacy concerns while maintaining model performance in scenarios characterized by diverse data sources.

In the healthcare domain, [30] explores the use of Federated Learning for collaborative model training across multiple healthcare institutions. The study highlights FL's potential to build robust models without compromising patient data privacy, making it an attractive approach for collaborative medical research and personalized healthcare applications. Moreover, [22] delves into the challenges and opportunities of applying Federated Learning in edge computing environments. The study explores how FL can enhance model training on resource-constrained devices, making it suitable for Internet of Things (IoT) applications and edge computing scenarios. Addressing the scalability of FL, [55] investigates techniques for federated optimization that can efficiently handle large-scale and dynamic datasets. The study contributes insights into the scalability challenges of FL and proposes methodologies to improve its applicability in scenarios involving vast and evolving data sources.

2.3 Transformers

Transformers have emerged as a groundbreaking architecture in the field of natural language processing (NLP) and machine learning. Introduced by [59], transformers have revolutionized various applications, including language understanding, translation, and image processing. The core idea of transformers lies in their attention mechanism, enabling the model to focus on different parts of the input sequence when making predictions. The architecture eliminates the sequential nature of recurrent neural networks (RNNs) and the fixed context window of convolutional neural networks (CNNs), allowing for more effective capture of long-range dependencies. The self-attention mechanism is a fundamental component of transformers. It allows the model to weigh different parts of the input sequence differently, capturing relationships between words regardless of their positions. This mechanism contributes to the model's ability to handle variable-length sequences and has proven to be crucial for tasks like language modeling and machine translation.

Bidirectional Encoder Representations from Transformers (BERT), introduced by [60], marked a significant advancement. BERT and subsequent pre-trained models like GPT-3 have demonstrated the power of transfer learning in NLP. These models are pre-trained on large corpora and fine-tuned for specific tasks, achieving state-of-the-art results in various benchmarks. While transformers gained prominence in NLP, their success has extended to other domains. Vision Transformer (ViT) introduced by [61], showcased the applicability of transformers in computer vision, challenging the dominance of convolutional neural networks (CNNs). Transformers have also been successfully applied in speech processing and reinforcement learning. Despite their success, transformers come with computational challenges due to their quadratic self-attention complexity, limiting their scalability. Researchers have explored techniques like attention pruning and approximations to address these issues. Additionally, advancements like the Longformer model have aimed to extend the reach of transformers to even longer sequences [62]. The concept of hybrid transformers, combining transformer architectures with other neural network structures, has gained attention. This approach aims to capture the strengths of transformers while mitigating their limitations. For example, models combining transformers with convolutional layers or recurrent components have shown promise in certain tasks.

CHAPTER 3: Methodology

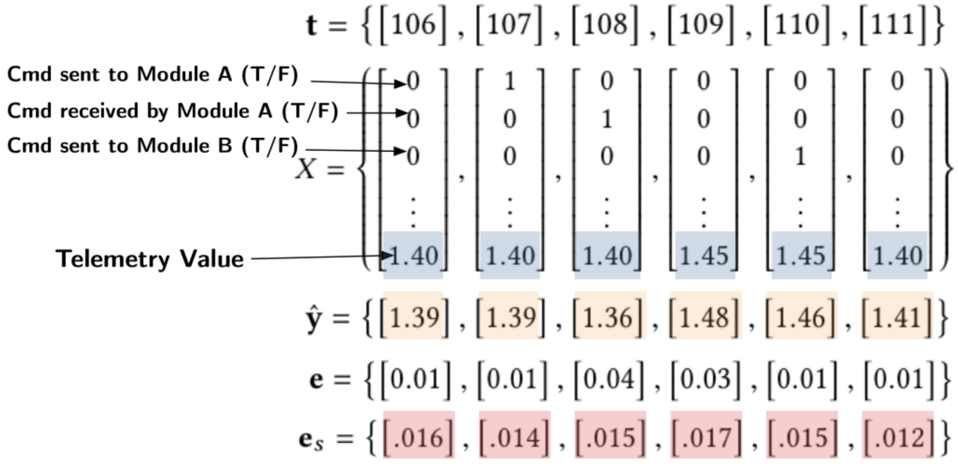
3.1 Introduction

Our study endeavors to assess diverse deep learning models within the context of our federated framework, specifically focusing on their effectiveness in spacecraft anomaly detection and their influence on real-time communication and system resource utilization. Employing a multifaceted approach, we combine a quantitative evaluation of anomaly detection performance with a system-level analysis. The dataset utilized encompasses information from two distinct sources, namely SMAP and MSL, incorporating alterations. The research leverages federated learning, exploring its suitability in real-time scenarios. Our selection of models considers not only their anomaly detection capabilities but also their impact on resource efficiency. This comprehensive methodology ensures practical insights into both model performance and their real-world applicability in the context of space missions.

3.2 Data Sources and Datasets

In this section, we provide an overview of the datasets utilized in our research for anomaly detection within the context of spacecraft systems, specifically from the Mars Science Laboratory (MSL) and Soil Moisture Active Passive (SMAP) missions. The dataset contains 82 normalized telemetry channels of data labeled with 105 anomalies, lending itself to be evaluated in a federated setting. A summary of the characteristics of our data pool is provided in in table 3-1. These datasets have been curated and annotated by the International Society of Automation (ISA) and serve as the foundation for our study. The selection of these datasets is crucial in addressing the unique challenges of spacecraft operations, including both planetary exploration and Earth observation, and ensuring the relevance of our research to the field of aerospace engineering.

The datasets used in this research have been obtained from ISA, a recognized authority in the field of automation and industrial processes. ISA's expertise in standardization and system monitoring makes their annotated datasets particularly valuable for anomaly detection within spacecraft systems, whether in the demanding conditions of planetary exploration (MSL) or Earth observation (SMAP).



Cmd sent to Module A (T/F)
Cmd received by Module A (T/F)
Cmd sent to Module B (T/F)
Telemetry Value
t = {[106], [107], [108], [109], [110], [111]}
X
ŷ = {[1.39], [1.39], [1.36], [1.48], [1.46], [1.41]}
e = {[0.01], [0.01], [0.04], [0.03], [0.01], [0.01]}
e_s = {[.016], [.014], [.015], [.017], [.015], [.012]}

Figure 3-1 Example of data used in this experiment showing actual telemetry value and corresponding metadata (Hundman et al, 2018)

The datasets provided by ISA encompass a wide range of telemetry and sensor data from both the MSL and SMAP missions. For the MSL mission, these datasets capture the intricate telemetry related to the exploration of Mars, covering aspects such as rover mobility, scientific instrument operations, and environmental conditions. For the SMAP mission, the datasets include Earth observation telemetry related to soil moisture levels and environmental factors. An essential aspect of these datasets is the detailed annotations provided by ISA. Figure 3-1 visualizes the general nature of the data and how we intend to use it for the task of anomaly detection, in that a single channel is a multivariable time series data where the first variable is the measurement of the channel while the remain variables are Booleans denoting a command event.

Table 3-1 Summary of two data sources used in this study SMAP at 55 and MSL 27 channels respectively.

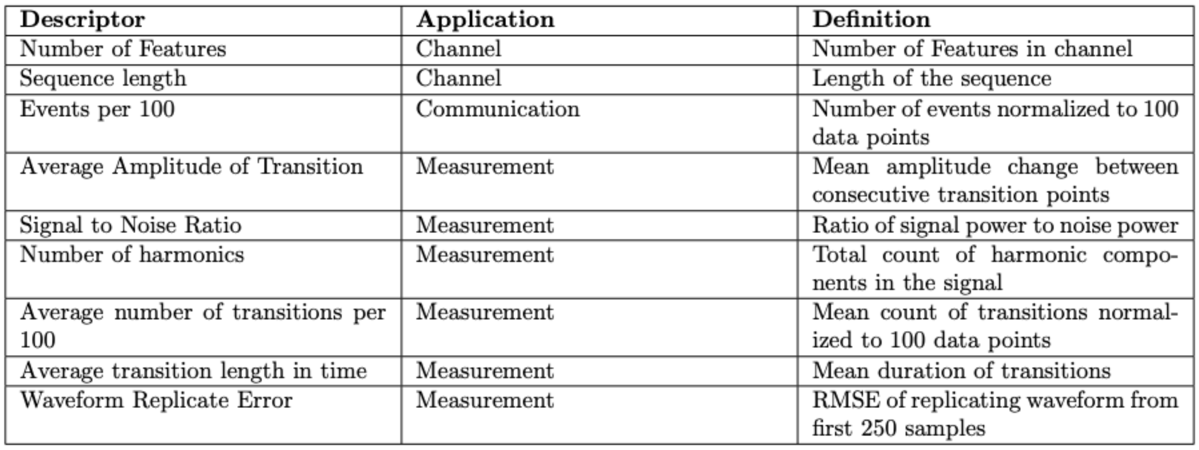
	SMAP	**MSL**
Number of sequences	25	55
Training set size	135183	39312
Testing set size	427617	73729
Anomaly rate(%)	13.13	10.27
Variable information	computational, radiation, temperature, power, activities, etc	

These annotations specify the occurrence and characteristics of anomalies within the telemetry data for both the MSL and SMAP missions. ISA's expertise in anomaly detection and their access to domain knowledge ensures that the annotations are accurate and informative. The presence of these annotations is invaluable for the supervised learning approach we adopt in our research, as they serve as ground truth labels for training and evaluating our machine learning models. The utilization of ISA-annotated datasets from both the MSL and SMAP missions enhances the credibility and reliability of our research while addressing the specific challenges of planetary exploration and Earth observation. By leveraging these datasets, we can develop and assess machine learning models that are tailored to the unique demands of both missions, with a focus on accurate anomaly detection and minimal false alarms.

3.3 Descriptive Statistics

Something we have not seen in similar work are corresponding descriptive statistics outside of the number of features and corresponding sequence lengths. In having these descriptors, we will have a more meaningful way to discuss any result anomaly detection metrics. The data in its raw state has already been normalized using the min-max technique so we know all values for the measurement reading will be in the range of -1 to 1. Commonly discussed in previous works are sequence length and number of features which we also incorporated. In a manual inspection of the data, we felt the best approach to incorporate some common signal descriptors in both the time and frequency domain summarized in Table 3-2.

Table 3-2 Descriptors used in this study and their definitions.



Descriptor	**Application**	**Definition**
Number of Features	Channel	Number of Features in channel
Sequence length	Channel	Length of the sequence
Events per 100	Communication	Number of events normalized to 100 data points
Average Amplitude of Transition	Measurement	Mean amplitude change between consecutive transition points
Signal to Noise Ratio	Measurement	Ratio of signal power to noise power
Number of harmonics	Measurement	Total count of harmonic components in the signal
Average number of transitions per 100	Measurement	Mean count of transitions normalized to 100 data points
Average transition length in time	Measurement	Mean duration of transitions
Waveform Replicate Error	Measurement	RMSE of replicating waveform from first 250 samples

Table 3-3 Summary statics of descriptors 1 of 2.

Descriptor	**Mean**	**Std**	**Min**	**Max**
Sequence length	2417.283	798.53	312.0	4308.0
Events per 100	1.200	0.875	0.0	2.45
Average Amplitude of Transition	0.5565	0.378	0.0	0.9989
Signal to Noise Ratio	2.2218e14	1.3992e15	-13433.84	8.9982e15
Number of harmonics	4.5309	1.6064	0.0	8.0
Average number of transitions per 100	14.8548	19.9994	0.0	67.9915
Average transition length in time	42.4008	211.7181	0.0	1846.0
Waveform Replicate Error	0.4945	0.5422	0.0	3.2771

Table 3-4 Summary statics of descriptors 2 of 2.

Descriptor	**25% (Q1)**	**50% (Median)**	**75% (Q3)**
Sequence length	2208.0	2690.0	2880.0
Events per 100	0.3443	1.1636	2.1014
Average Amplitude of Transition	0.3008	0.5727	0.9411
Signal to Noise Ratio	-0.8093	0.0	0.2498
Number of harmonics	4.0	5.0	6.0
Average number of transitions per 100	0.0347	5.9055	18.4668
Average transition length in time	1.5639	4.7495	15.5010
Waveform Replicate Error	0.0041	0.4062	0.8314

3.4 Model Selection and Description

The heart of our research lies in the selection, development, and description of the deep learning models used for anomaly detection in the MSL and SMAP mission datasets. In this section, we present an overview of the selected models, their specific characteristics, and how they are tailored to address the unique challenges of each mission. For our research, we have chosen a suite of machine learning models known for their effectiveness in anomaly detection. These models have been selected based on their adaptability to the complex telemetry data and their capacity to operate in real-time conditions. The following models have been chosen for this study.

3.4.1.1 Long Short-Term Memory (LSTM) Networks:

LSTM networks are a type of recurrent neural network (RNN) known for their ability to capture sequential dependencies in data. We employ LSTM networks to model temporal dependencies within the telemetry data, making them ideal for capturing anomalies that manifest over time.

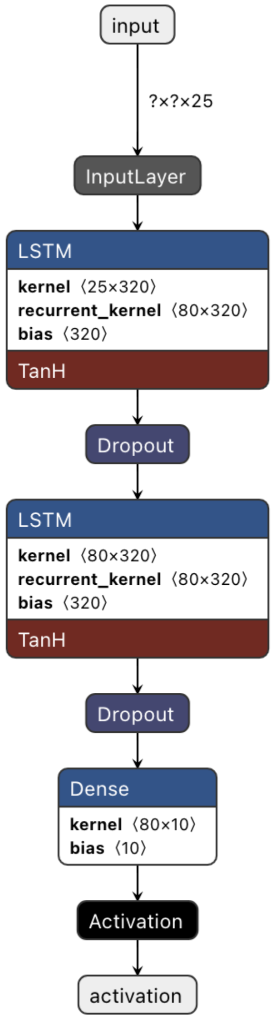


Figure 3-2 Sample LSTM architecture used in this study.

LSTMs are a type of neural network that can handle long sequences without facing issues like vanishing or exploding gradient problems. This is possible due to a more memory mechanism that incorporates several key components such as the cell state, input gate, forget gate, and output gate. These components work together to regulate the flow of information within the network, allowing it to retain or discard information over time. The cell state acts like a conveyor belt that runs through the entire sequence, enabling LSTMs to maintain long-term dependencies by preserving information across many time steps.

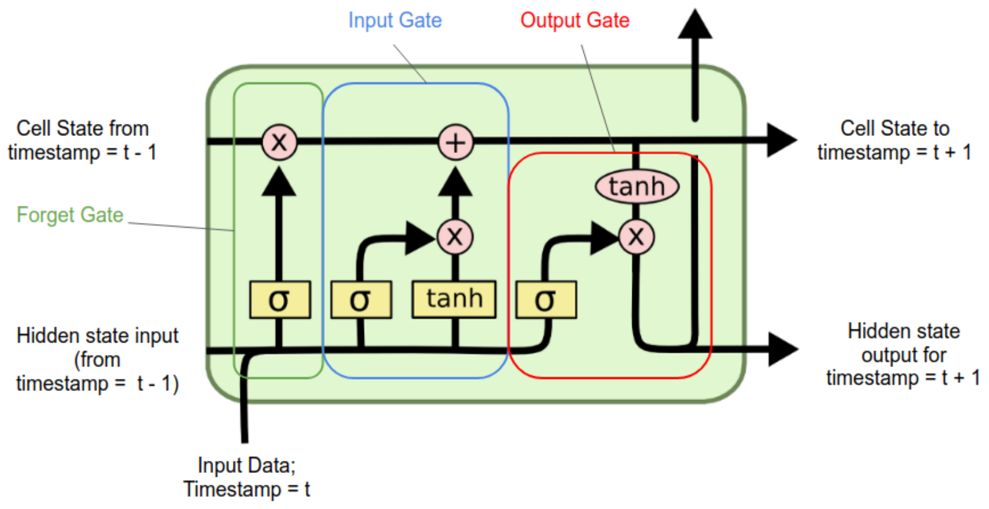

Input Gate
Output Gate
Cell State from timestamp = t - 1
Forget Gate
Hidden state input (from timestamp = t - 1)
Input Data; Timestamp = t
Cell State to timestamp = t + 1
Hidden state output for timestamp = t + 1
σ
σ
tanh
σ
tanh

Figure 3-3 Anatomy of an LSTM layer.

The input gate plays a crucial role in the LSTM network by controlling the flow of new information into the cell state. It analyzes the current input and the network's internal state to decide which parts of the input should be stored in the cell state and which parts should be discarded. This adaptive filtering capability allows LSTMs to selectively retain relevant information. Various studies in spacecraft anomaly detection have demonstrated their suitability for this problem [21].

3.4.1.2 1D Convolutional Neural Networks (1D CNNs)

1D CNNs are well-suited for detecting patterns within one-dimensional data, which is characteristic of many telemetry features. These models excel at feature extraction and are crucial for identifying localized anomalies within the datasets. One-dimensional Convolutional Neural Networks (1D CNNs) are neural network architectures specifically designed for processing one-dimensional sequential data, such as time series or sequences of text. Unlike traditional 2D CNNs used for image processing, which operate on two-dimensional grids of pixels, 1D CNNs apply convolutions across the temporal dimension of the input. In a 1D CNN, the convolutional layers slide one-dimensional filters (kernels) along the input sequence, extracting local patterns and features. These filters capture relationships between adjacent elements in the sequence, enabling the network to learn hierarchical representations of the input data. Our implementation is motivated by attempting to capture the temporal components of the data with a series of convolutional networks with a minimal about of parameters to facilitate an onboard deployment, shown in figure 3-4.

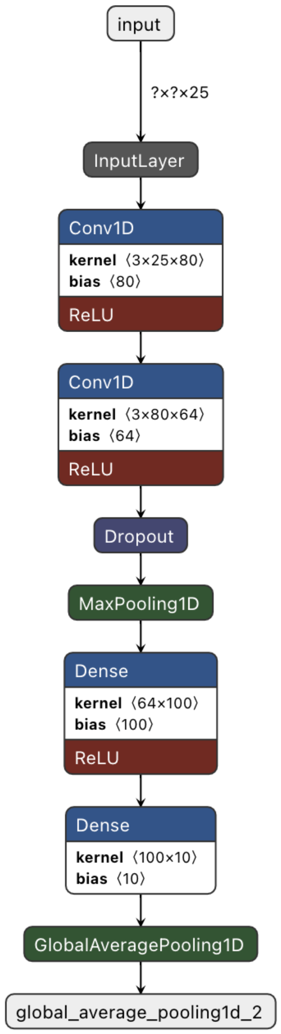


Figure 3-4 Sample CNN architecture used in this study.

By stacking multiple convolutional layers followed by pooling layers, 1D CNNs can capture increasingly abstract and complex patterns in the input sequence. Pooling layers reduce the dimensionality of the feature maps while preserving the most relevant information, helping to prevent overfitting and improve computational efficiency. 1D CNNs are commonly used in various sequential data tasks, including natural language processing (NLP), speech recognition, and time series analysis. They have shown effectiveness in tasks such as sentiment analysis, named entity recognition, and anomaly detection, often achieving competitive performance with simpler architectures compared to recurrent neural networks (RNNs) and long short-term memory (LSTM) networks. With their ability to automatically learn hierarchical representations from sequential data, 1D CNNs offer a powerful tool for capturing patterns and extracting features in a wide range of applications including spacecraft anomaly detection [29].

3.4.1.3 CNN-LSTM Models

Combining the strengths of both 1D CNNs and LSTMs, CNN-LSTM models offer a holistic approach to anomaly detection. They extract spatial and temporal features from the data, making them particularly effective in recognizing complex patterns that may indicate anomalies [63].

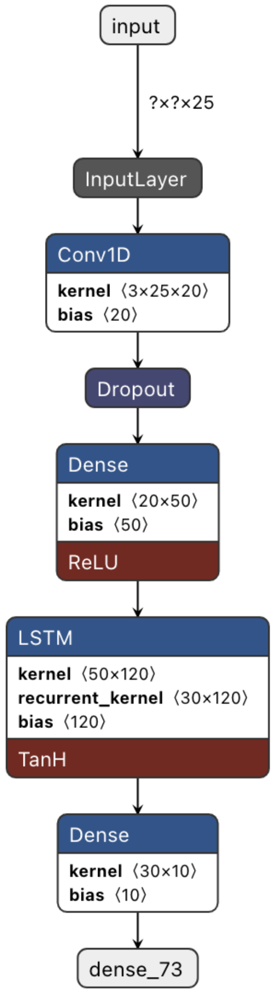


Figure 3-5 Sample CNN-LSTM architecture used in this study.

3.4.1.4 Transformers

Transformers are a neural network architecture that has changed the field of natural language processing (NLP) significantly. They introduced the self-attention mechanism, which is different from traditional sequential models like recurrent neural networks (RNNs) and convolutional neural networks (CNNs). These models process input data sequentially or with fixed-size receptive fields, whereas transformers use a self-attention mechanism. Transformers consist of multiple layers of self-attention and feed-forward neural networks, often stacked on top of each other. These layers enable the model to learn hierarchical representations of the input data, capturing both local and global patterns.

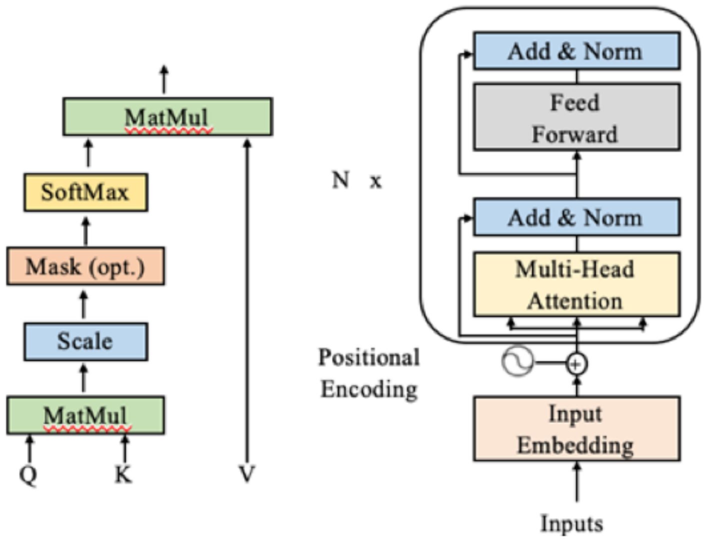


Figure 3-6 Dot product Attention and transformer encoder.

Additionally, transformers use positional encoding to provide the model with information about the order of tokens in the input sequence, as the self-attention mechanism alone does not inherently capture sequential information.

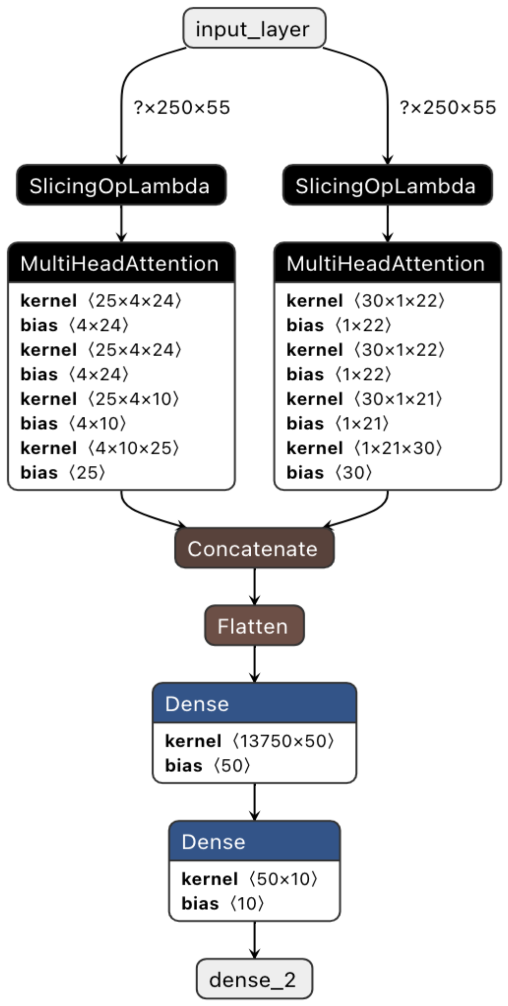


Figure 3-7 Sample Transformer architecture used in this study.

Transformers have shown great success in multiple natural language processing (NLP) tasks, such as machine translation, text generation, and sentiment analysis. They have also been used for spacecraft anomaly detection, as highlighted in a study by [49]. In our research, we employed two variations of this approach. For answering RQ1 or simply employing the model in the vanilla federated averaging implementation, we utilized a series of multi-head attention layers, followed by a dense layer. In RQ2 we looked at augmenting the federated averaging process to operate on a disjointed model, we devised a composite model to separate the two components across different stages of training shown in figure 3-8.

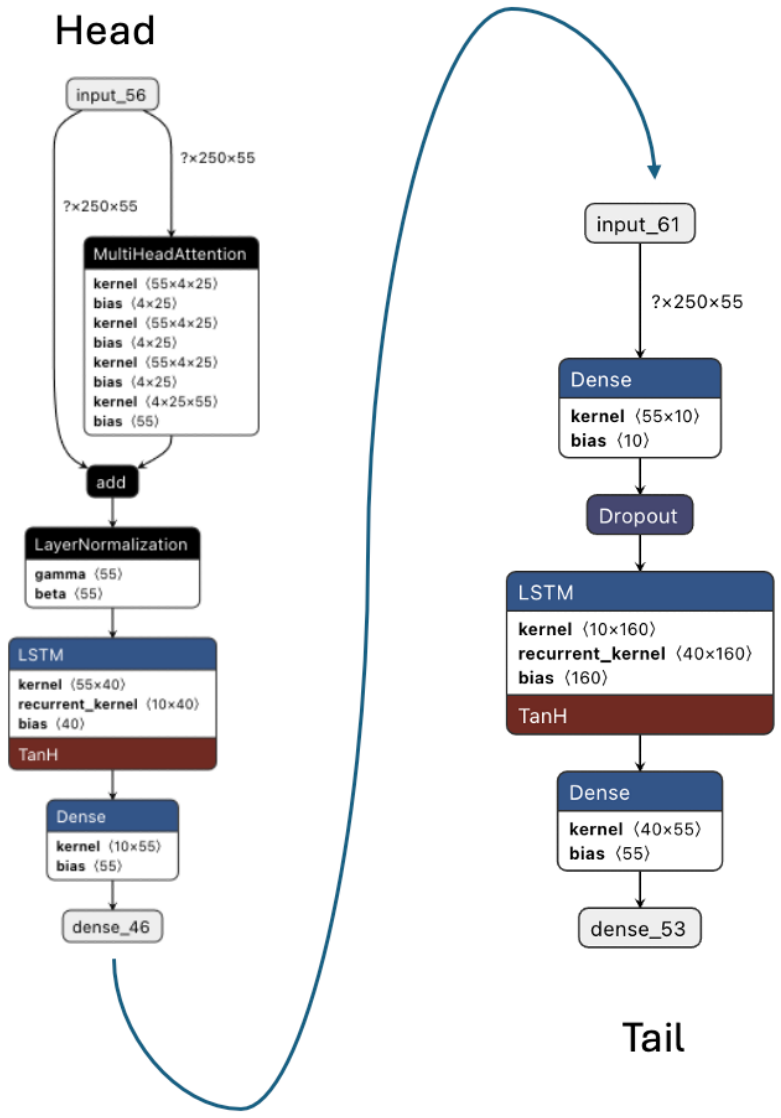


Figure 3-8 Sample Transformer architecture used in RQ2, the front half of hybrid-transformer.

Anticipating the integration of a hybrid transformer, our model selection spans a variety of deep learning models highlighted in the literature review for their established relevance to spacecraft anomaly detection. Each of these models is specifically tailored to address distinct facets of anomaly detection within our federated framework. Together, they provide a comprehensive approach to identifying anomalies in telemetry data, effectively addressing the diverse challenges inherent in both planetary exploration and Earth observation missions.

3.5 Reconstruction Error in Federated Learning

As there are dozens of telemetry anomaly detection methods, we limit our design to focus on gradient-based reconstruction error implementations, representative of most modern DL AD systems, where the goal is to translate the classical approximation function, equation 1, into one where the objective is to reconstruct the original input space as closely as possible, equation 2.

$$F(x, w) \cong y \tag{1}$$

$$F(x, w) \cong \bar{x} \tag{2}$$

The difference between the original input and the reconstructed output can then be used to measure how familiar the approximator is with the sample given and thereby linearly equate to how atypical each input is.

$$e_i = |x_i - \bar{x_i}| \tag{3}$$

In a centralized setting, the objective is to find the set weights that define an algorithm such that it minimizes the error between an output and a given input along with a set of weights, equations 4 and 5.

$$minimize\,\frac{1}{n}\sum_{i=1}^{n} e_i \tag{4}$$

$$F(w) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w) st. f_i(w) = loss(x_i, \overline{x_\iota}, w) \tag{5}$$

To determine the most effective combination of weights, a training process can utilize a gradient method such as stochastic gradient descent. This method involves selecting a random subset of samples, or mini-batches, from a dataset at each epoch to calculate a gradient, which is then used to update the model weights in the following training step. The process of updating is repeated over multiple epochs until the optimal set of weights is obtained. To determine the most effective combination of weights, a training process can utilize a gradient method such as stochastic gradient descent. This method involves selecting a random subset of samples, or mini-batches, from a dataset at each epoch to calculate a gradient, which is then used to update the model weights in the following training step. The process of updating is repeated over multiple epochs until the optimal set of weights is obtained.

$$w_{t+1} \leftarrow wt - \eta\nabla f(w_t; x_k, \overline{x_k}) \tag{6}$$

Building upon this notion, the goal of our design becomes to enable the discovery of a function that minimizes the reconstruction error as an aggregate of k models trained on their own respective datasets or more formally, equation 7.

$$w^G{}_{t+1} = w^G{}_t - \eta \nabla f(w^G{}_t) = w^G{}_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k, st. g_k = \nabla F_k(w_t) \tag{7}$$

Once the reconstruction error is generated, the subsequent step is to establish a threshold value that separates normal and anomalous data points. Due to the likelihood of false positives when using reconstruction error-based techniques, we have adopted a dynamic threshold approach based on the research by [21], as detailed in equations 8 and 9. This approach is used during post-processing to assist with the detection process.

$$\epsilon = \mu\,(e_i) + z\sigma(e_i) \tag{8}$$

$$\epsilon_i = \frac{\Delta\mu(e_i)/\mu(e_i) + \Delta\sigma\ (e_i)/\sigma\ (e_i)}{|e_a| + |E|^2} \tag{9}$$

such that $e_a = \{e_i \in e_i \,|e_i > \epsilon\}$,

$\Delta\mu(e_i) = \mu(e_i) - \mu(\{e_i \in e_i \,|e_i < \epsilon\}),$

$\Delta\sigma(e_i) = \sigma(e_i) - \sigma(\{e_i \in e_i \,|e_i < \epsilon\}), and$

$E = set\ of\ individual\ groupings\ of\ e_a$

Having these formulations, algorithm 1 FedAvg can then be implemented, with the specific task of anomaly detection.

Algorithm 1: FedAvg

```
                    $Server\ executes$:
1. $initialize\ w_0^G$
2. $for\ each\ round\ t = 1,2,\ldots,do$
    i.   $m \leftarrow max(C * K, 1)$
    ii.  $S_t \leftarrow (\ random\ set\ of\ m\ clients)$
    iii. $for\ each\ client\ \ k\ \in S_t\ in\ parallel\ do$
        i. $w_{t+1}^k \leftarrow ClientUpdate(k, w_t^G)$
3. $w_{t+1}^G \leftarrow \sum_{k=1}^K \frac{n_k}{n}\ w_{t+1}^k$

                    $ClientUpdate(k, w)$:
1. $for\ each\ local\ epoch\ i\ do$
    i. $for\ b\ \in Client's\ data\ \ do$
        i. $w_k \leftarrow w - \eta \nabla l(w;\ b)$
2. $return\ w_k\ to\ server$
```

For research question 2, we shift our focus from building a unified model to one that can be trained and personalized simultaneously. To do this, algorithm 1 is modified to incorporate a model that can be split in two and reunited as needed while optimizing for the two objectives.

Algorithm 2: Hybridized FedAvg

```
$\textbf{\textit{Server executes}}$:

    $initialize\ w_0^G\ and\ \emptyset_0$
    $append\ w_0^G\ to\ \emptyset_0\ to\ define\ \emptyset_0^G$
    $for\ each\ round\ t = 1,2,\ldots,do$
            $m \leftarrow max(C * K, 1)$
            $S_t \leftarrow (random\ set\ of\ m\ clients)$
            $for\ each\ client\ k \in S_t\ in\ parallel\ do$
                    $w_{t+1}^k \leftarrow ClientUpdate(k, \emptyset_t^G)$
    $w_{t+1}^G \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

$\textbf{\textit{ClientUpdate}}(\boldsymbol{k}, \emptyset)$:

    $for\ each\ local\ epoch\ i\ do$
            $for\ b \in Client's\ data\ do$
                    $\emptyset_k \leftarrow \emptyset - \eta \nabla l(\emptyset;\ b)$

    $extract\ w_k from\ \emptyset_k$
    $return\ w_k\ to\ server$
```

3.6 Simulation Testbed Design

The design of our testbed aims to achieve scalable simulations while also enabling transitioning quickly to a deployment setting, addressing the rigor demanded by any flight system. In that, we initially started exploring a virtual machine approach but found that the overhead incurred was a detriment to the system's performance and suitability for embedded devices, generally representative of onboard flight computers.

Containers are lightweight and standalone executable software package that includes everything needed to run an application, such as code, libraries, system tools, and settings [11]. A container is isolated from the underlying host operating system and other containers, providing a consistent runtime environment for the application regardless of the underlying infrastructure [11]. Containers are often used to deploy and run applications in a portable and efficient manner.

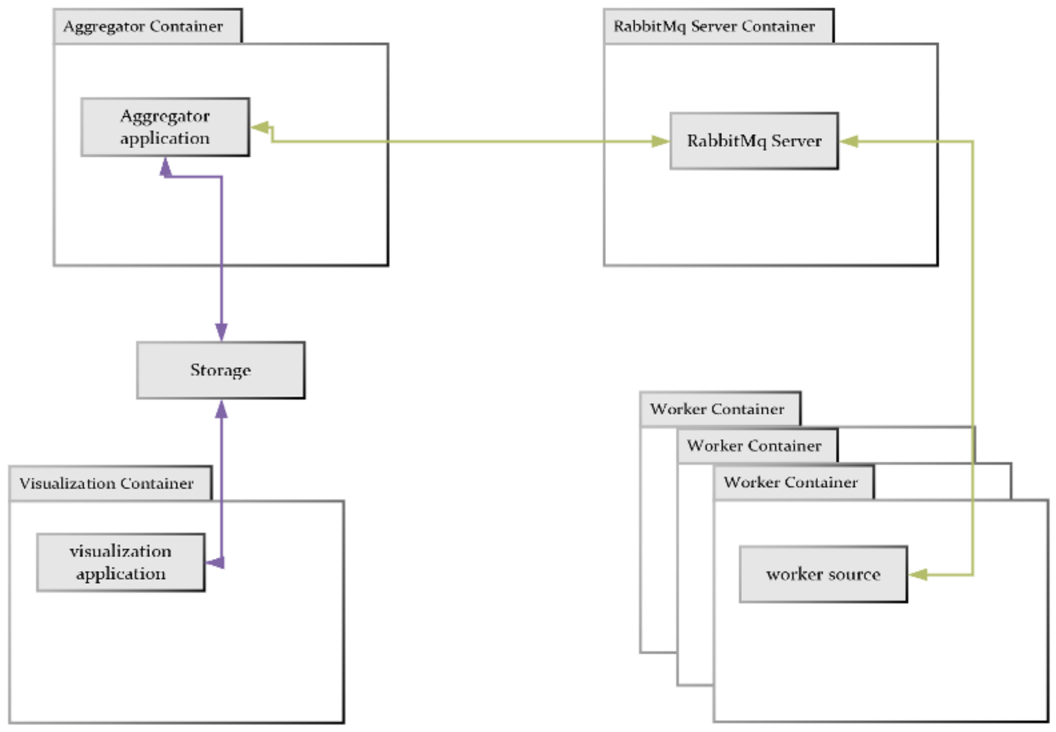


Figure 3-9 Container level overview of Testbed.

They can be easily moved between computing environments such as development, testing, and production. Moreover, containers can be configured with communications ports, available memory, and CPU limits [11]. Communication between worker nodes and the aggregator is done with RabbitMQ, an AMPQ (advance message passing queuing) protocol implementation [31]. A wrapper was implemented to enable customized functionality, including additional tunable security measures and a rule engine object to facilitate adaptability and futurization. The aggregator and worker nodes are decedents of the communication class; thus, both instantiate a base rule engine as an aggregate object, which the aggregator can then modify during runtime. In doing so, each party implicitly agrees upon a uniform set of rules and corresponding actions.

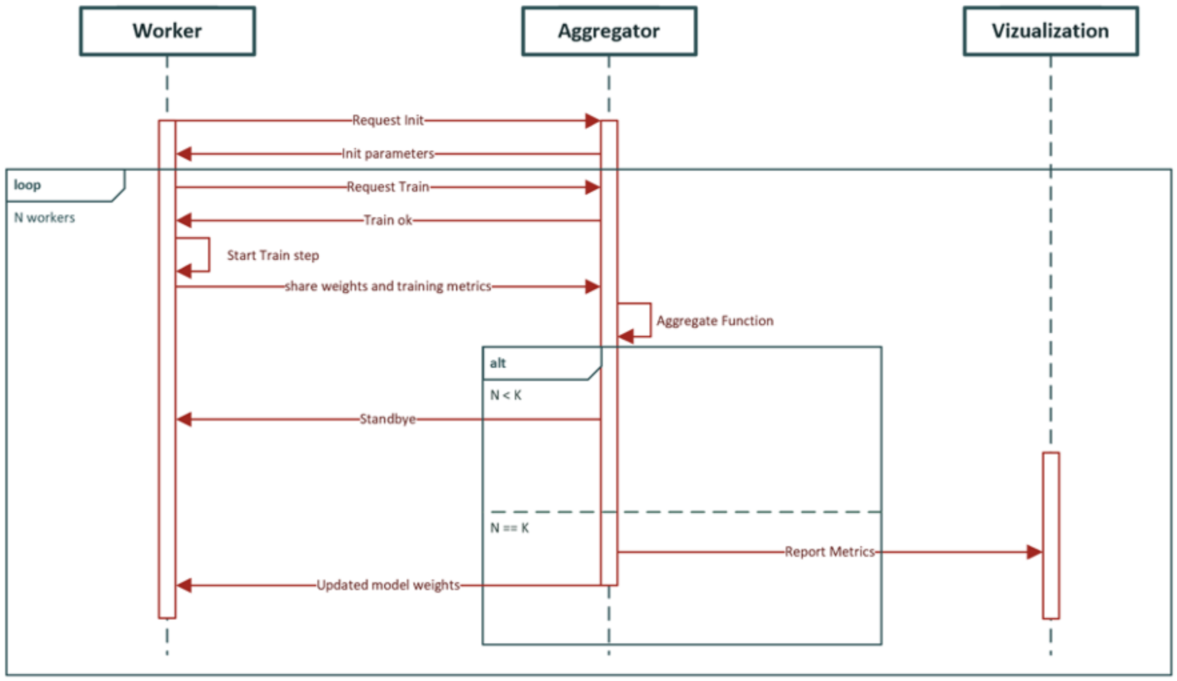


Figure 3-10 Data flow diagram of FL Testbed.

3.7 Experimental Design

In addressing Research Question 1 (RQ1), which delves into the effectiveness of federated learning (FL) for spacecraft telemetry anomaly detection using various deep learning (DL) models, our experiment seeks to elucidate the performance dynamics within the federated learning paradigm. Structured to evaluate the suitability of different DL models for real-world deployment on edge devices in spacecraft telemetry, the experiment commences with 30 simulations for each model, resulting in a total of 120 simulation runs using SMAP data exclusively. Each simulation randomly allocates 10 channels for global model assessment and distributes the remaining SMAP channels among clients, capped at 30. Hyperparameters are randomly selected from predefined ranges specific to each model architecture shown in tables 3-5 and 3-6. The experiment proceeds with 5 client epochs per batch of data, with the current global model computing an average reconstruction loss on holdout data at each federation step, terminating after 5 divergence steps are noted.

Baseline federated learning performance for each model is established using vanilla federated averaging, employing fixed numbers of local epoch architecture-specific hyperparameters. Simulations run until a patience value is exceeded, involving training and evaluation of DL models on decentralized fragments of the SMAP dataset. Metrics such as reconstruction loss across clients over federation steps, reconstruction error for the global model against a holdout dataset, number and size of messages, and CPU usage for training and inference serve as benchmarks for subsequent comparisons and form the basis for evaluating the impact of the federated learning paradigm. Incorporation of MSL data, with an additional 30 features, follows a similar protocol.

Table 3-5 Hyperparameters varied in this study.

Hyperparamter	**Application**	**low**	**High**
Number Filters	Convolution Layer	6	30
Kernel Size	Convolution Layer	6	20
Units	Dense Layer	5	20
Rate	Dropout Layer	0.1	0.4
Units	LSTM Layer	30	45
Key Dimension	Self-Attention Layer	3	15
Value Dimension	Self-Attention Layer	10	25

Throughout the federated learning process, continuous monitoring and recording of reconstruction loss for anomaly detection, quantification of communication overhead through message exchange, determination of the number of federation steps required for convergence, and measurement of local training time on each edge device are emphasized. Adherence to resource constraints, including message size and CPU time capped at 2GB, is paramount. The experiment also involves repeating a truncated version with both SMAP and SOIL datasets, forcing the system to handle varying input spaces for all models and adding a padding step to the preprocessing pipeline. Research Question 2 (RQ2) is further explored using both SMAP and MSL datasets in a similar protocol, incorporating the rapid adaptation component where the model is given a small sample of available training data for each channel before commencing the anomaly detection component.

Table 3-6 Hyperparameters held constant in this study.

Hyperparamter	**Application**	**Value**
Learning Rate	Training	0.01
Epochs	Data	9
Number of Heads	Self-Attention Layer	1
Batch Size	Data	70
Window Length(time-steps)	Data	196
Prediction Window Length(time-steps)	Data	10
Patience	Federated Training	3
Number Parallel Self-Attention layers	Transformer Model	2
Optimizer	Trainable Parameters	Adam
Loss	Trainable Parameters	RMSE

Having a collection of globally trained models from these protocols the experimentation continues by examining them for their AD performance by applying each of the models to the entire test dataset using the equation 9 to make the determination of anomalies detected.

3.8 Metrics

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{12}$$

We assessed the effectiveness of various methods using the commonly employed metrics in anomaly detection, namely Precision, Recall, and F1 scores, where TP, FP, and FN represent the counts of true positives, false positives, and false negatives, respectively. The definitions for precision, recall, and F1 are provided in equations 10, 11, and 12 respectively. Consistent with the evaluation approach outlined in [21], we employed point-wise scoring. In real-world scenarios, anomalies in time series data typically manifest as contiguous abnormal segments. Anomaly alerts may be triggered at any subset within the actual anomaly window. Thus, the entire anomalous segment is considered correctly detected if the model identifies any single time point as an anomaly.

CHAPTER 4: Results

4.1 Simple Federated Anomaly Detection Experiment (RQ1)

In this work, we set out to investigate the viability of employing a federated learning architecture for the task of spacecraft anomaly detection. As described in our experimental design section, we used a collection of previous works to derive the minimal target threshold of approximately 0.2 RMSE in the reconstruction error phase of our system before any meaningful anomaly detection can commence. Using only the channel data with 25 features, figure 4-1 shows the average loss at each federation step for each of the architectures evaluated when tested on each client's data available at that time step. All architectures were able to achieve this goal before the study limit of 100 steps with the fastest being the CNN-LSTM model which reached the 0.2 target at 35 steps followed by the transformer and LSTM model at approximately 45 steps.

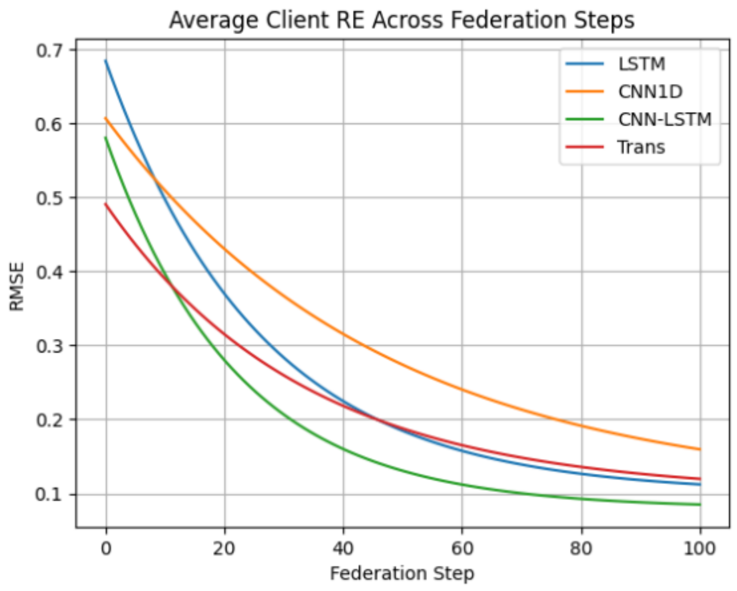


Figure 4-1 Distribution of reconstruction error of client models across all simulations categorized by model architecture using only 25 feature data.

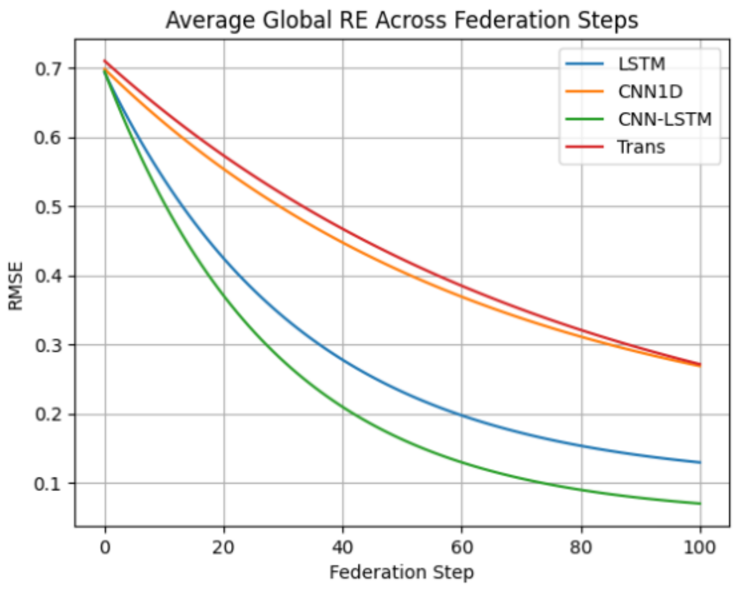


Figure 4-2 Average reconstruction error of global model across all simulations categorized by model architecture on 25 feature data.

To assess the viability of deploying these models on unseen data, the global model is tested with 10 additional channels worth of hold-out data progressing at a similar cadence with the client data. Figure 4-2 shows that with unseen data, only the LSTM and CNN-LSTM are able to reach the target threshold while the CNN and transformer model settle around a loss of 0.28 upon the study limit of 100 steps.

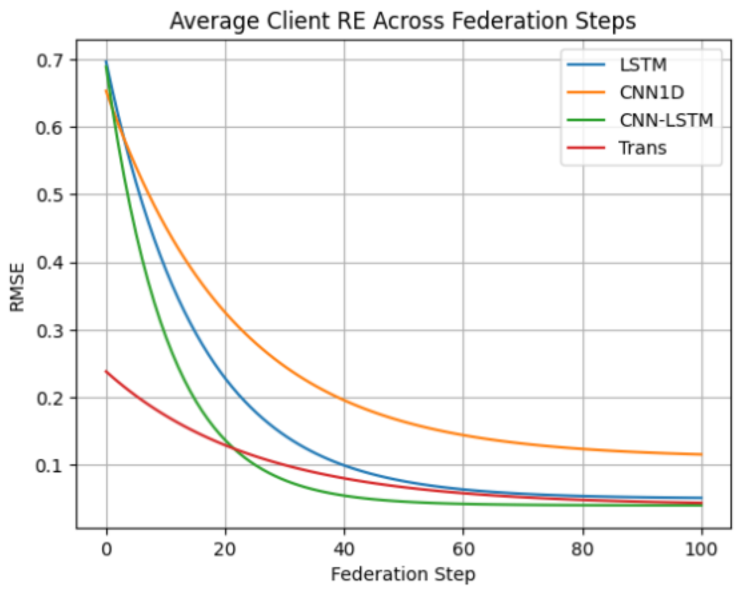


Figure 4-3 Distribution of reconstruction error of client models across all simulations categorized by model architecture using both 25 and 55 feature data.

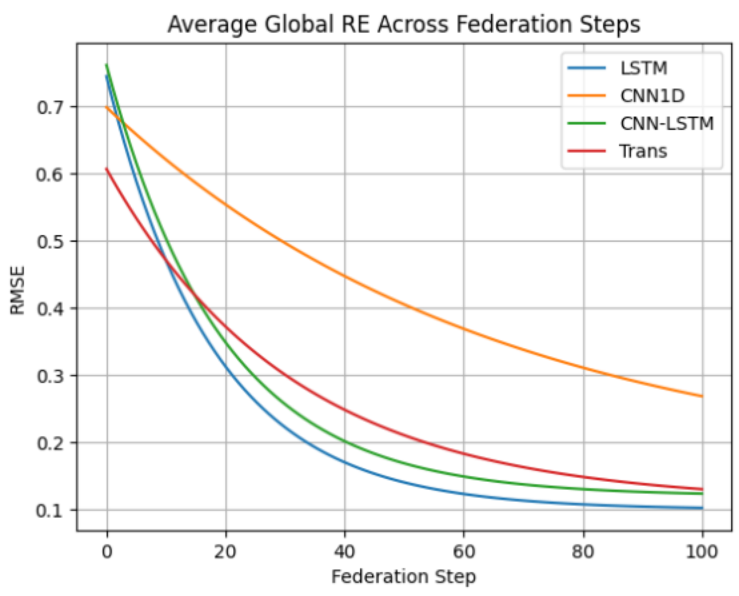


Figure 4-4 Distribution of reconstruction error of Global models across all simulations categorized by model architecture using both 25 and 55 feature data.

To further understand the behavior of the federated learning approach in a deployment setting we reran the simulations with incorporating the datasets of 55 channels each along with the original set. Figures 4-4 and 4-5 show the reconstruction performance respectively for the client and global models, yielding a similar result to the 25 features-only experiment where all models were able to reach the target loss before the study limit on client data and the CNN-LSTM and LSTM models being the only two to achieve this milestone for the global data which was also supplemented with sets of 55 feature data.

Table 4-1 Precision, recall and F1-Scores by architecture on 25 feature data only. (*note: unique model per channel data in centralized setting)

Type	Precision	Recall	F1 Score
LSTM*	0.885	0.780	0.829
CNN-LSTM	0.878	0.584	0.702
Transformer*	0.668	0.547	0.601
LSTM	0.903	0.497	0.642
Transformer	0.867	0.469	0.608
CNN	0.820	0.464	0.592

Having a collection of fitted global models, we proceeded to evaluate their anomaly detection abilities using equations 8 and 9 across all 82 channels worth of test data yielding tables 4-1 and 4-2 for the 25 feature only and 25 and 55 feature data respectively. For both experiments, the performance in reconstruction error linearly maps directly to the anomaly detection performance. Regarding the 25-feature experiment, all models reached a precision of 0.8 or greater however struggled with recall averaging around 0.49.

Table 4-2 Precision, recall and F1-Scores by architecture for 25 and 55 feature data. (*Note: Unique model per channel data in a centralized setting.

Type	Precision	Recall	F1 Score
LSTM*	0.800	0.875	0.836
Transformer*	0.780	0.680	0.727
CNN-LSTM	0.912	0.516	0.659
LSTM	0.852	0.513	0.640
Transformer	0.855	0.510	0.639
CNN1D	0.846	0.388	0.532

The best-performing model was the CNN-LSTM reached an F1 Score of 0.702 surpassing the performance of the channel-specific transformer models of [49] but still not besting the channel-specific LSTM models by [21]. In the 25 and 55 feature data, our best was also the CNN-LSTM model reaching an F1 score of 0.65 and a precision value of 0.91 surpassing both the channel-specific LSTM and transformer models however failed to do so when comparing recall.

4.2 Hybrid Model Anomaly Detection Experiment (RQ2)

As part of our efforts to answer research question 2, which aims to enhance the performance of anomaly detection by customizing the head component of our model for each channel while creating a universal model that can be used for new problems, we have been focusing on developing a distinct model. This model consists of a self-attention head and an LSTM tail. Our objective was to optimize this composite model on each of the client's data, with the resultant global module serving as the tail portion. The experimentation protocol was similar in that we evaluated the performance in terms of reconstruction error on both the client and global data pools shown in Figure 4-6. With this configuration, the target loss is reached with 30 steps on average with the client's data while the global target never reaches the target value within the study limit however a correlation can be used to infer the performance when re-appending the head component.

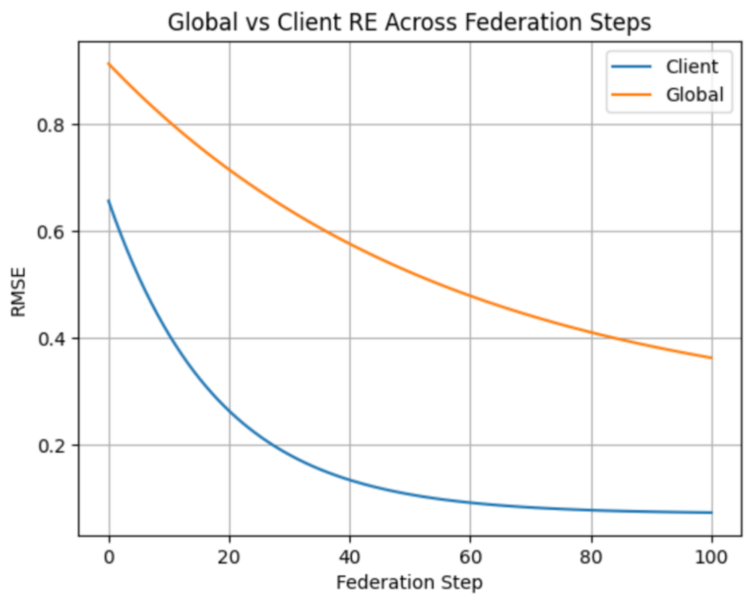


Figure 4-5 Comparison of Global vs client reconstruction error across federation steps.

In comparing the performance of anomaly detection, the previous protocol underwent a shift by incorporating a mini-fit step before running the AD testing protocol. This mini-fit step exposed each model to a small fraction of the available data, varying batch size, learning rate, number of steps, and epochs. The findings, as summarized in Table 4-3, reveal that optimal precision was achieved at 0.8501 with a batch size of 70, a learning rate of 0.05, 1 epoch, and 10 steps. However, this came at the cost of recall, which was 0.6545. Regarding the F1 scores, the optimal configuration was noted to be a batch size of 50, a learning rate of 0.02, 5 epochs, and 5 steps, resulting in an F1 score of 0.7531. This score surpassed the channel-specific transformers models from [49] of 0.72. Lastly, in terms of recall, the optimal configuration was a batch size of 50, a learning rate of 0.02, 40 epochs, and 2 steps, which yielded a value of 0.7090.

Table 4-3 Comparison of Mini-fitting Parameters across and impact on AD performance.

Batch Size	Epochs	Steps	Learning Rate	RMSE	F1 Score	Precision	Recall
50	5	3	0.020	0.0084	0.7451	0.7940	0.7026
		5	0.020	0.0079	*0.7531	0.8159	0.6994
	20	5	0.020	0.0091	0.7386	0.7791	0.7026
	40	2	0.020	0.0086	0.7443	0.7834	0.7090
70	1	10	0.050	0.0096	0.7394	0.8501	0.6545
	5	5	0.001	0.0759	0.5455	0.8165	0.4109
			0.050	0.0090	0.7393	0.8017	0.6865
	10	2	0.080	0.0143	0.7003	0.8376	0.6032
		3	0.050	0.0104	0.7265	0.8272	0.6481
		5	0.010	0.0096	0.7333	0.8163	0.6673
			0.050	0.0108	0.7219	0.7822	0.6705
	20	5	0.001	0.0111	0.7270	0.8239	0.6545
		10	0.020	0.0153	0.6900	0.6941	0.6865

CHAPTER 5: Discussion and Conclusion

5.1 Discussion

In this study, we aimed to explore the potential of federated learning (FL) for onboard spacecraft detection and to enhance anomaly detection (AD) performance through the use of hybrid-transformer architecture. Our investigation into research question 1 involved simulating various FL processes across multiple deep-learning architectures. While some architectures showed promising results, the conventional centralized approach using a single model per channel consistently outperformed the FL implementations. Regarding research question 2, we found that employing a hybrid transformer framework alongside a multi-head attention module significantly improved AD performance with minimal initial data. In interpreting the results for research question 1, we observed that CNN-LSTM and LSTM models achieved superior performance, emphasizing the importance of temporal components in telemetry channels. These models demonstrated proficiency in capturing and utilizing temporal information inherent in telemetry data, owing to their ability to handle sequential dependencies and long-term patterns. Conversely, 1DCNN and multi-head attention models performed poorly compared to CNN-LSTM and LSTM models, likely due to their limitations in capturing temporal dynamics and handling varying lengths of sequential data.

All FL models struggled with the recall metric, indicating the vast variability across different channel data that a single model failed to capture adequately for onboard deployment. For research question 2, we achieved competitive AD performance by exposing a hybrid architecture to the FL process, allowing for the development of a global understanding of underlying patterns. Leveraging the transformer module's normalization effect and multi-head attention mechanism, the hybrid model captured temporal intricacies and channel-specific patterns, resulting in robust performance in AD tasks.

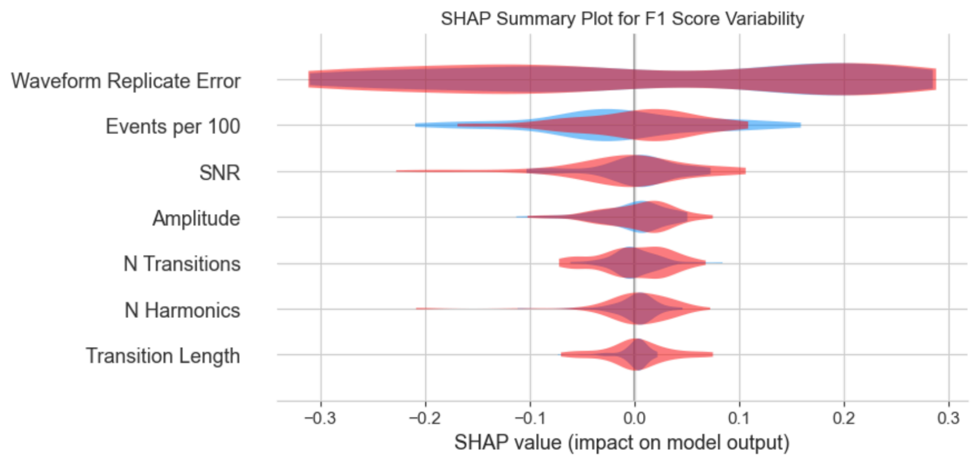


Figure 5-1 Shap plot of RF regressor targeting F1 scores for LSTM(Blue) and Hybrid-Transformer (Red) on AD task.

Despite the effectiveness of the hybrid-transformer approach, it did not surpass the performance of the channel-specific LSTM model. To understand this, we analyzed the impact of descriptors on model performance using Shapley Additive Explanation analysis. We found that the waveform-replicate error descriptor had the largest impact on both LSTM and hybrid-transformer models, with simple reconstruction tasks contributing to higher F1 values. The distribution of the events per 100 timesteps descriptor had a more negative impact on the channel-specific LSTM model compared to the hybrid model, indicating the latter's ability to incorporate temporal data effectively.

Table 5-1 Mission critical metrics of onboard deployment for models in this study.

Model	N Parameters	Storage Size	Message Size	Wall-Time	F1
Transformer	13145	51.1kb	27433b	5201ms	0.639
LSTM	95850	374.41kb	31758b	8251ms	0.640
CNN	36214	141.46kb	29888b	3457ms	0.532
CNN-LSTM	14400	56.25kb	31554b	4952ms	0.659
Hybrid-Transformer	34515	134.82kb	12238b	22921ms	0.753

Regarding mission implications, our findings suggest the viability of employing FL onboard small satellites for AD, potentially opening up precious downlink bandwidth. However, considerations such as message size and computational resource consumption need to be addressed. We recommend the hybrid-transformer model for its superior performance and minimal message size, although missions with more conservative computing budgets may find the CNN-LSTM model suitable with further AD performance refinements. Future research should focus on optimizing computational resources and refining AD performance for practical deployment.

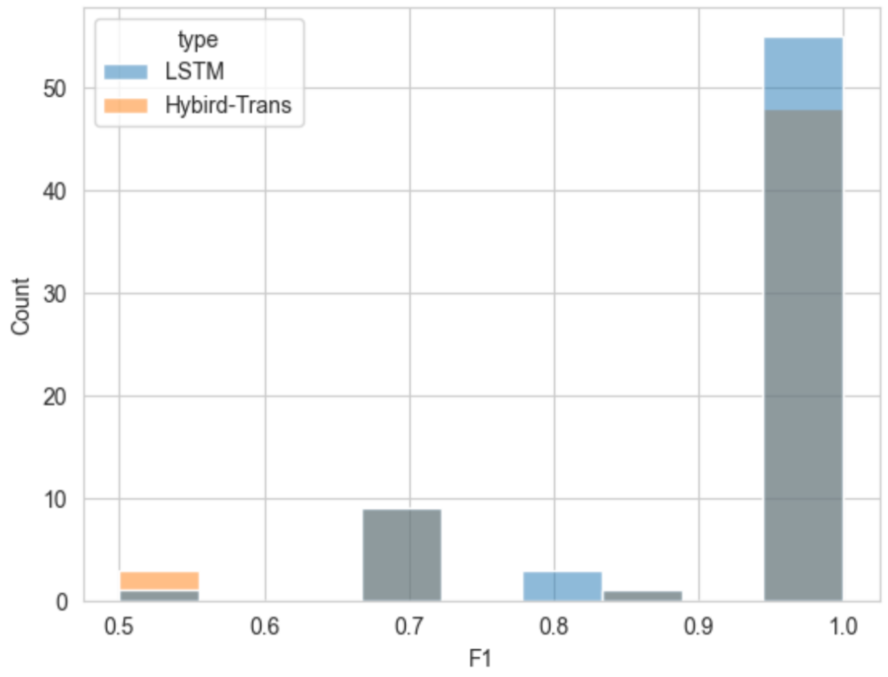
5.2 Conclusion

In conclusion, our investigation into federated learning for onboard telemetry anomaly detection highlights the intricate balance between model performance, computational resources, and practical deployment considerations. Our study aimed to address two key questions: the feasibility of utilizing a federated learning approach for onboard telemetry anomaly detection and the potential of a hybrid transformer architecture to achieve state-of-the-art results. To answer these questions, we developed a container-based simulation framework with its own communication and scheduling mechanism. Our findings reveal that while federated learning processes generally yield decently performing models in terms of anomaly detection precision, further refinement is necessary due to underwhelming recall scores. Additionally, our exploration of the hybrid transformer architecture, while comparable to state-of-the-art methods, comes with slower training times that may impact mission resource considerations. Notably, the integration of the multi-head attention component was found to enhance the capture of communication events in telemetry channels even when compared to channel-specific implementations.

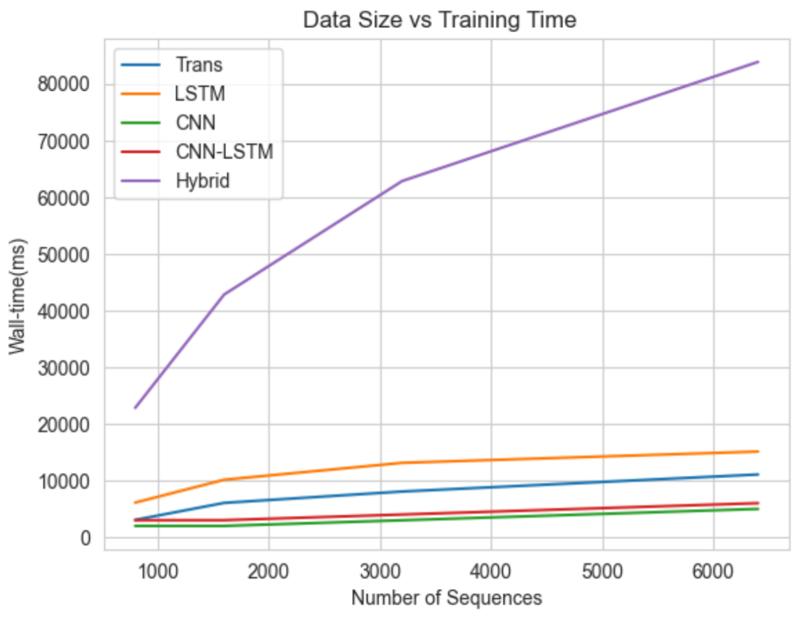
An analysis of mission-critical metrics suggests that the hybrid transformer implementation is the most promising option, albeit requiring significantly more budget for onboard compute resources. For future work, we propose porting our implementation onto a collection of boards used in cube and nanosat missions to validate our findings. Furthermore, we aim to refine the CNN-LSTM implementation with a similar paradigm as the hybrid transformer, modularizing the CNN component for personalization. Using the findings of the SHAP analysis can also be studied further as means for identifying channel categories and stratifying results. Additionally, we are interested in integrating a large-scale foundational model into the server side for further enhancement of anomaly detection performance. By addressing these avenues for future research and considering the practical implications of our findings, we aim to contribute to advancements in telemetry anomaly detection and pave the way for more efficient and effective onboard spacecraft detection systems.

CHAPTER 6: APPENDIX

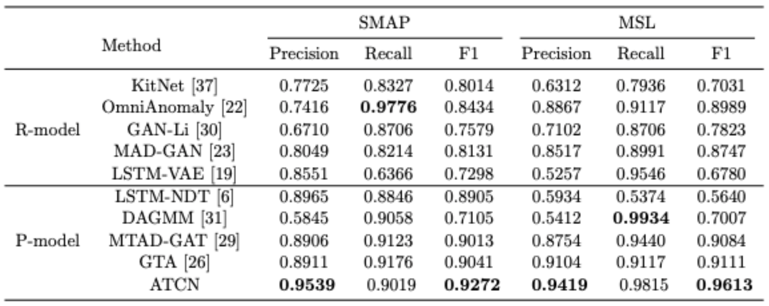
6.1.1.1 F1 Comparison of LSTM[21] and our hybrid transformer



6.1.1.2 Wall-times for each model in our study across ranging batch size.



6.1.1.3 Comparison of SOTA results



	Method	SMAP			MSL		
		Precision	Recall	F1	Precision	Recall	F1
R-model	KitNet [37]	0.7725	0.8327	0.8014	0.6312	0.7936	0.7031
	OmniAnomaly [22]	0.7416	**0.9776**	0.8434	0.8867	0.9117	0.8989
	GAN-Li [30]	0.6710	0.8706	0.7579	0.7102	0.8706	0.7823
	MAD-GAN [23]	0.8049	0.8214	0.8131	0.8517	0.8991	0.8747
	LSTM-VAE [19]	0.8551	0.6366	0.7298	0.5257	0.9546	0.6780
P-model	LSTM-NDT [6]	0.8965	0.8846	0.8905	0.5934	0.5374	0.5640
	DAGMM [31]	0.5845	0.9058	0.7105	0.5412	**0.9934**	0.7007
	MTAD-GAT [29]	0.8906	0.9123	0.9013	0.8754	0.9440	0.9084
	GTA [26]	0.8911	0.9176	0.9041	0.9104	0.9117	0.9111
	ATCN	**0.9539**	0.9019	**0.9272**	**0.9419**	0.9815	**0.9613**

6.1.1.4 Descriptor functions python code.

```python
from numpy.fft import fft, ifft
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
def snr(x:pd.DataFrame):
    axis=0
    ddof=0
    a = x[0].to_numpy()
    a = np.asanyarray(a)
    m = a.mean(axis)
    sd = a.std(axis=axis, ddof=ddof)
    return np.where(sd == 0, 0, m/sd)

def get_harmonics(x:pd.DataFrame, d:int):
    sr =1
    ts = len(x[0])/sr
    t = np.arange(0,ts,1)
    # Perform FFT
    X = fft(x[0])

    # Perform peak detection
    peaks, _ = find_peaks(np.abs(X), distance=d)
    freqs = np.fft.fftfreq(len(X), 1)
    peak_freqs = freqs[peaks]
    if not len(peak_freqs):
        return 0

    # Assuming the first peak corresponds to the fundamental frequency
    fundamental_frequency = peak_freqs[0]
    harmonic_numbers = np.round(peak_freqs / fundamental_frequency)
    harmonic_freqs = fundamental_frequency * harmonic_numbers


    fundamental_frequency = peak_freqs[0]  # Assuming the first peak corresponds to the fundamental frequency
    harmonic_numbers = np.round(peak_freqs / fundamental_frequency)
    return len(harmonic_numbers)

def waveform_replicate(x:pd.DataFrame, size:int):
    #sample first n timesteps
    sample = x[0].head(size).values
    #get total number of timesteps
    dsize = x[0].shape[0]
    #determine number of replicates needed
    nsamples = int(np.ceil(dsize/size))
    #create replicates in 2D
    repeats = np.array([ sample for _ in range(nsamples)])
    #compute rmse between original data and replicates 1D
    return np.sqrt(mean_squared_error(repeats.flatten()[:dsize],x[0]))

def get_len(x:pd.DataFrame):
    return x[0].shape[0]

def get_des(x:pd.DataFrame):
    return x[0].mean(),x[0].std(),x[0].min(), x[0].max()

def count_events(x:pd.DataFrame):
    #get columns in data, exclude telemetry point 0
    cs = [*list(x.columns)]
    cs.pop(0)
    #create array of sum of sums across both axis for every 100 steps
    sums = [x[cs][i:i+100].sum().sum() for i in np.arange(0,x.shape[0],100)[:-1]]
    #return average
    return np.mean(sums)/x.shape[1]
def dsp_metrics(x,l):
    #average out transitent spikes by l
    y = x[0].rolling(l).mean()
    if not y.std():
        return 0,0,0
    #init prev, spans array, amps array and previous direction
    prev = y[0]
    spans = []
    span = 1
    amps = [0]
    amp = 0
    prev_direction = False

    #for each sample, get delta from previous then compute direction
    for i in y[1:]:
        dt = (i-prev)
        direction = dt >= 0
        #if nonzero direction change, store length and amplitude
        if prev_direction != direction and dt != 0:
            spans.append(span)
            amps.append((np.abs(amp-i)))
            amp = 0
            span =1
        #else increment
        else:
            span += 1
        prev_direction = direction
        prev = i
        size = y.shape[0]
    return np.mean(np.array(spans)), len(spans)/size*100, np.mean(amps)
```

6.1.1.5 Sample rule engine for Hybrid experiment

```
class GT_fedAvgh_Engine(GT_fedAvg_Engine):
    def __int__(self, connection_handler):
        super().__init__(connection_handler)

    def start_train(self, data):
        x, y = next(self.ch.idata)
        x = padd(x)
        tl = 0
        t1 = time.time()
        if self.ch.train_steps:
            LOGGER.warning("Time since last train {:.2f} seconds".format(t1 - self.last_train))
        for _ in range(self.ch.cfg['epochs']):
            tl = self.step(x, y)
            LOGGER.warning(f"loss: {tl}")
        t2 = time.time()
        self.last_train = t2
        LOGGER.warning("Elapsed time: {:.2f} seconds".format(t2 - t1))
        msg = {'id': self.ch.QUEUE, 'loss': tl.numpy().tobytes().hex(),
               'step': self.ch.train_steps}
        for i in range(len(self.ch.model.layers[1].trainable_variables)):
            msg[i] = self.ch.model.layers[1].trainable_variables[i].numpy().tobytes().hex()
        self.ch.add_msg_to_q(self.ch.wt, self.ch.QUEUE, dumps(msg), 'train_metrics')
        self.ch.train_steps += 1

    def process_metrics_job(self, data):
        dct = loads(data)
        self.ch.buffer[dct['step']].append(dct)
        ids = set(map(lambda x: x['id'], self.ch.buffer[self.ch.step]))
        if (len(ids.intersection(self.ch.C)) == len(self.ch.C) and
            self.ch.round != 0) or (self.ch.round == 0 and len(ids) >= 10):
            data = {}
            for d in self.ch.buffer[self.ch.step]:
                data[f"{d['id']}-step"] = d['step']
                data[f"{d['id']}-loss"] = float(np.ndarray(1, dtype=np.float32, buffer=bytes.fromhex(d['loss']))[0])

            for i in range(len(self.ch.tail.trainable_variables)):
                update = np.zeros(self.ch.tail.trainable_variables[i].numpy().shape)
                vals = list(map(lambda x: x[str(i)], self.ch.buffer[self.ch.step]))
                for v in vals:
                    update += np.ndarray(self.ch.tail.trainable_variables[i].numpy().shape, dtype=np.float32,
                                         buffer=bytes.fromhex(v))
                update = update / len(self.ch.comm_metrics)
                self.ch.tail.trainable_variables[i].assign(update)

            scores = []
            for id in self.ch.idata:
                x, y = next(id)
                x = padd(x)
                yhat = self.ch.tail.predict(x)
                scores.append(mse(yhat, y).numpy())
            data['global-loss'] = float(np.mean(scores))

            if int(self.ch.step) > 1:
                less_than = bool(self.ch.last_target_score > data['global-loss'])
                if less_than and self.ch.cfg['direction_min']:
                    self.ch.tail.save(os.path.join(self.ch.run_metrics_location, "tail.h5"))
                    self.ch.g_min = data['global-loss']
                    self.ch.patience_test = 0
                else:
                    self.ch.patience_test += 1
            self.ch.run_data.append(data)
            save_file = open(os.path.join(self.ch.run_metrics_location, "training.json"), "w")
            json.dump(self.ch.run_data, save_file, indent=6)
            save_file.close()

            LOGGER.warning(f"TEST UPDATE: {data['global-loss']}")
            weights = {}
            for i in range(len(self.ch.tail.trainable_variables)):
                weights[i] = self.ch.tail.trainable_variables[i].numpy().tobytes().hex()
            # send broadcast signal to update_weights as long patience is within tolerance
            if self.ch.patience_test < self.ch.cfg['patience']:
                self.post_agg_processing(weights)
            else:
                print("***Patience exceeded, experiment terminated!***\n\n\n")
                q_items = list(self.ch.comm_metrics.keys())
                for prty in q_items:
                    self.ch.add_msg_to_q(prty, self.ch.QUEUE, "blank", 'reset')
                self.ch.comms_enabled = False
            self.ch.last_target_score = data['global-loss']
            self.ch.round += 1
            self.ch.step += 1
            self.ch.buffer[self.ch.step] = []
        else:
            self.ch.add_msg_to_q(dct['id'], self.ch.QUEUE, "standbye", 'info')

    def update_weights_job(self, data):
        dct = loads(data)
        for i in range(len(self.ch.model.layers[1].trainable_variables)):
            weights = np.ndarray(self.ch.model.layers[1].trainable_variables[i].numpy().shape, dtype=np.float32,
                                 buffer=bytes.fromhex(dct[str(i)]))
            self.ch.model.layers[1].trainable_variables[i].assign(weights)
        self.ch.add_msg_to_q(self.ch.wt, self.ch.QUEUE, self.ch.QUEUE, 'train_req')
```

CHAPTER 7: BIBLIOGRAPHY

[1] T. Stibor, J. Timmis, and C. Eckert, “A Comparative Study of Real-Valued Negative Selection to Statistical Anomaly Detection Techniques,” in *Artificial Immune Systems*, C. Jacob, M. L. Pilat, P. J. Bentley, and J. I. Timmis, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 262–275. doi: 10.1007/11536444_20.

[2] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, “A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs,” in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, Vienna Austria: ACM, Aug. 2021, pp. 1–7. doi: 10.1145/3465481.3469190.

[3] D. Park, Y. Hoshi, and C. C. Kemp, “A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, Jul. 2018, doi: 10.1109/LRA.2018.2801475.

[4] “A Multimodal Anomaly Detector for Robot-Assisted Feeding Using an LSTM-Based Variational Autoencoder | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Mar. 29, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8279425

[5] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, “A review of applications in federated learning,” *Computers & Industrial Engineering*, vol. 149, p. 106854, Nov. 2020, doi: 10.1016/j.cie.2020.106854.

[6] P. P. Shinde and S. Shah, “A Review of Machine Learning and Deep Learning Applications,” in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Aug. 2018, pp. 1–6. doi: 10.1109/ICCUBEA.2018.8697857.

[7] Y. Yu, X. Si, C. Hu, and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures,” *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/neco_a_01199.

[8] M. M. Khan, S. Hossain, P. Mozumdar, S. Akter, and R. H. Ashique, “A review on machine learning and deep learning for various antenna design applications,” *Heliyon*, vol. 8, no. 4, p. e09317, Apr. 2022, doi: 10.1016/j.heliyon.2022.e09317.

[9] P. Kairouz *et al.*, “Advances and Open Problems in Federated Learning,” *MAL*, vol. 14, no. 1–2, pp. 1–210, Jun. 2021, doi: 10.1561/2200000083.

[10] G. Biswas, H. Khorasgani, G. Stanje, A. Dubey, S. Deb, and S. Ghoshal, “An Approach To Mode and Anomaly Detection with Spacecraft Telemetry Data,” *International Journal of Prognostics and Health Management*, vol. 7, no. 4, Art. no. 4, 2016, doi: 10.36001/ijphm.2016.v7i4.2467.

[11] B. Bashari Rad, H. Bhatti, and M. Ahmadi, “An Introduction to Docker and Analysis of its Performance,” *IJCSNS International Journal of Computer Science and Network Security*, vol. 173, p. 8, Mar. 2017.

[12] Q. Li, X. Zhou, P. Lin, and S. Li, “Anomaly detection and fault Diagnosis technology of spacecraft based on telemetry-mining,” in *2010 3rd International Symposium on Systems and Control in Aeronautics and Astronautics*, Jun. 2010, pp. 233–236. doi: 10.1109/ISSCAA.2010.5633180.

[13] D. Li, D. Chen, J. Goh, and S. Ng, “Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series.” arXiv, Jan. 15, 2019. doi: 10.48550/arXiv.1809.04758.

[14] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, p. 15:1-15:58, Jul. 2009, doi: 10.1145/1541880.1541882.

[15] X.-H. Le, H. V. Ho, G. Lee, and S. Jung, “Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting,” *Water*, vol. 11, no. 7, Art. no. 7, Jul. 2019, doi: 10.3390/w11071387.

[16] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data.” arXiv, Jan. 26, 2023. doi: 10.48550/arXiv.1602.05629.

[17] B. Zong *et al.*, “Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection,” presented at the International Conference on Learning Representations, Jan. 2023. Accessed: Apr. 28, 2023. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[18] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, “Deep Learning for Anomaly Detection: A Review,” *ACM Comput. Surv.*, vol. 54, no. 2, p. 38:1-38:38, Mar. 2021, doi: 10.1145/3439950.

[19] R. Chalapathy and S. Chawla, “Deep Learning for Anomaly Detection: A Survey.” arXiv, Jan. 23, 2019. doi: 10.48550/arXiv.1901.03407.

[20] “Detecting anomalies in spacecraft telemetry using evolutionary thresholding and LSTMs | Proceedings of the Genetic and Evolutionary Computation Conference Companion.” Accessed: Apr. 26, 2023. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3449726.3459411

[21] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2018, pp. 387–395. doi: 10.1145/3219819.3219845.

[22] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated Learning: Challenges, Methods, and Future Directions,” *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[23] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated Learning: Strategies for Improving Communication Efficiency.” arXiv, Oct. 30, 2017. doi: 10.48550/arXiv.1610.05492.

[24] C. He *et al.*, “FedML: A Research Library and Benchmark for Federated Machine Learning.” arXiv, Nov. 08, 2020. doi: 10.48550/arXiv.2007.13518.

[25] Y. Liang, Y. Guo, Y. Gong, C. Luo, J. Zhan, and Y. Huang, “FLBench: A Benchmark Suite for Federated Learning,” in *Intelligent Computing and Block Chain*, W. Gao, K. Hwang, C. Wang, W. Li, Z. Qiu, L. Wang, A. Zhou, W. Qian, C. Jin, and Z. Zhang, Eds., in Communications in Computer and Information Science. Singapore: Springer, 2021, pp. 166–176. doi: 10.1007/978-981-16-1160-5_14.

[26] M. H. Garcia, A. Manoel, D. M. Diaz, F. Mireshghallah, R. Sim, and D. Dimitriadis, “FLUTE: A Scalable, Extensible Framework for High-Performance Federated Learning Simulations.” arXiv, Nov. 14, 2022. doi: 10.48550/arXiv.2203.13789.

[27] D. Sculley *et al.*, “Hidden Technical Debt in Machine Learning Systems,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015. Accessed: Nov. 17, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

[28] S. Fuertes, G. Picart, J.-Y. Tourneret, L. Chaari, A. Ferrari, and C. Richard, “Improving Spacecraft Health Monitoring with Automatic Anomaly Detection Techniques,” in *SpaceOps 2016 Conference*, Daejeon, Korea: American Institute of Aeronautics and Astronautics, May 2016. doi: 10.2514/6.2016-2430.

[29] J. Hong, C.-C. Liu, and M. Govindarasu, “Integrated Anomaly Detection for Cyber Security of the Substations,” *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1643–1653, Jul. 2014, doi: 10.1109/TSG.2013.2294473.

[30] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, “IoT Healthcare Analytics: The Importance of Anomaly Detection,” in *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, Mar. 2016, pp. 994–997. doi: 10.1109/AINA.2016.158.

[31] P. Dobbelaere and K. S. Esmaili, “Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper,” in *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, in DEBS ’17. New York, NY, USA: Association for Computing Machinery, Jun. 2017, pp. 227–238. doi: 10.1145/3093742.3093908.

[32] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection.” arXiv, May 27, 2018. doi: 10.48550/arXiv.1802.09089.

[33] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng, “Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT,” *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9179–9189, Jun. 2022, doi: 10.1109/JIOT.2021.3100509.

[34] S. K. Ibrahim, A. Ahmed, M. A. E. Zeidan, and I. E. Ziedan, “Machine Learning Methods for Spacecraft Telemetry Mining,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 4, pp. 1816–1827, Aug. 2019, doi: 10.1109/TAES.2018.2876586.

[35] S. Omar, A. Ngadi, and H. H. Jebur, “Machine Learning Techniques for Anomaly Detection: An Overview,” *IJCA*, vol. 79, no. 2, pp. 33–41, Oct. 2013, doi: 10.5120/13715-1478.

[36] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, “MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks,” in *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, Eds., Cham: Springer International Publishing, 2019, pp. 703–716. doi: 10.1007/978-3-030-30490-4_56.

[37] D. Li, D. Chen, L. Shi, B. Jin, J. Goh, and S.-K. Ng, “MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks.” arXiv, Jan. 15, 2019. doi: 10.48550/arXiv.1901.04997.

[38] K.-H. Lee *et al.*, “Multi-Game Decision Transformers”.

[39] H. Zhao *et al.*, “Multivariate Time-series Anomaly Detection via Graph Attention Network.” arXiv, Sep. 04, 2020. doi: 10.48550/arXiv.2009.02040.

[40] “Multivariate Time-Series Anomaly Detection via Graph Attention Network | IEEE Conference Publication | IEEE Xplore.” Accessed: Mar. 29, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9338317

[41] M. Wetherholt, “NASA’s Approach to Software Assurance,” Sep. 01, 2015. Accessed: Nov. 17, 2023. [Online]. Available: https://ntrs.nasa.gov/citations/20150015579

[42] B. Lal, K. Calvin, L. M. Barbier, A. K. Kludze, and E. L. Mclarney, “NASA’s Responsible AI Plan.” Sep. 19, 2022. Accessed: Nov. 17, 2023. [Online]. Available: https://ntrs.nasa.gov/citations/20220013471

[43] D. Chen, D. Gao, W. Kuang, Y. Li, and B. Ding, “pFL-Bench: A Comprehensive Benchmark for Personalized Federated Learning.” arXiv, Oct. 13, 2022. doi: 10.48550/arXiv.2206.03655.

[44] V. Mugunthan, A. Peraire-Bueno, and L. Kagal, “PrivacyFL: A Simulator for Privacy-Preserving and Secure Federated Learning,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, in CIKM ’20. New York, NY, USA:

Association for Computing Machinery, Oct. 2020, pp. 3085–3092. doi: 10.1145/3340531.3412771.

[45] A. Jeffery, H. Howard, and R. Mortier, “Rearchitecting Kubernetes for the Edge,” in *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*, in EdgeSys ’21. New York, NY, USA: Association for Computing Machinery, Apr. 2021, pp. 7–12. doi: 10.1145/3434770.3459730.

[46] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, “Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, in KDD ’19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 2828–2837. doi: 10.1145/3292500.3330672.

[47] R. Muddinagiri, S. Ambavane, and S. Bayas, “Self-Hosted Kubernetes: Deploying Docker Containers Locally With Minikube,” in *2019 International Conference on Innovative Trends and Advances in Engineering and Technology (ICITAET)*, Dec. 2019, pp. 239–243. doi: 10.1109/ICITAET47105.2019.9170208.

[48] S. K. Khatri, D. Kumar, A. Dwivedi, and N. Mrinal, “Software Reliability Growth Model with testing effort using learning function,” in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, Sep. 2012, pp. 1–5. doi: 10.1109/CONSEG.2012.6349470.

[49] H. Meng, Y. Zhang, Y. Li, and H. Zhao, “Spacecraft Anomaly Detection via Transformer Reconstruction Error,” in *Proceedings of the International Conference on Aerospace System Science and Engineering 2019*, Z. Jing, Ed., in Lecture Notes in Electrical Engineering. Singapore: Springer, 2020, pp. 351–362. doi: 10.1007/978-981-15-1773-0_28.

[50] L. Liu, L. Tian, Z. Kang, and T. Wan, “Spacecraft Anomaly Detection with Attention Temporal Convolution Network.” arXiv, Mar. 13, 2023. Accessed: Jan. 18, 2024. [Online]. Available: http://arxiv.org/abs/2303.06879

[51] L. Liu, L. Tian, Z. Kang, and T. Wan, “Spacecraft Anomaly Detection with Attention Temporal Convolution Network.” arXiv, Mar. 13, 2023. doi: 10.48550/arXiv.2303.06879.

[52] Z. Zeng, G. Jin, C. Xu, S. Chen, and L. Zhang, “Spacecraft Telemetry Anomaly Detection Based on Parametric Causality and Double-Criteria Drift Streaming Peaks over Threshold,” *Applied Sciences*, vol. 12, no. 4, Art. no. 4, Jan. 2022, doi: 10.3390/app12041803.

[53] S. Baireddy *et al.*, “Spacecraft Time-Series Anomaly Detection Using Transfer Learning,” presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 1951–1960. Accessed: Apr. 26, 2023. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021W/AI4Space/html/Baireddy_Spacecraft_Time-Series_Anomaly_Detection_Using_Transfer_Learning_CVPRW_2021_paper.html

[54] M. Khorasani, M. Abdou, and J. Hernández Fernández, “Streamlit Use Cases,” in *Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework*, M. Khorasani, M. Abdou, and J. Hernández Fernández, Eds., Berkeley, CA: Apress, 2022, pp. 309–361. doi: 10.1007/978-1-4842-8111-6_11.

[55] N. Rieke *et al.*, “The future of digital health with federated learning,” *npj Digit. Med.*, vol. 3, no. 1, Art. no. 1, Sep. 2020, doi: 10.1038/s41746-020-00323-1.

[56] C. W. Johnson, “The Natural History of Bugs: Using Formal Methods to Analyse Software Related Failures in Space Missions,” in *FM 2005: Formal Methods*, J. Fitzgerald, I. J. Hayes, and A. Tarlecki, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 9–25. doi: 10.1007/11526841_3.

[57] L. Lyu, H. Yu, and Q. Yang, “Threats to Federated Learning: A Survey.” arXiv, Mar. 04, 2020. doi: 10.48550/arXiv.2003.02133.

[58] K. Bonawitz *et al.*, “Towards Federated Learning at Scale: System Design,” *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, Apr. 2019.

[59] A. Vaswani *et al.*, “Attention Is All You Need.” arXiv, Aug. 01, 2023. doi: 10.48550/arXiv.1706.03762.

[60] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.

[61] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” arXiv, Jun. 03, 2021. doi: 10.48550/arXiv.2010.11929.

[62] I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The Long-Document Transformer.” arXiv, Dec. 02, 2020. doi: 10.48550/arXiv.2004.05150.

[63] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, “A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs,” in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, Vienna Austria: ACM, Aug. 2021, pp. 1–7. doi: 10.1145/3465481.3469190.