BENCHMARKING FEDERATED LEARNING FRAMEWORKS FOR AVIATION NETWORK DATA

by

Barathwaja Subash Chandar

A Thesis

Submitted to the Graduate Faculty

of the

University of North Dakota

In Partial Fulfillment of the Requirements

for the degree of

Master of Science in Computer Science

Grand Forks, North Dakota
December 2023

Name: Barathwaja Subash Chandar

Degree: Master of Science

This document, submitted in partial fulfillment of the requirements for the degree from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

DocuSigned by:
Prakash Ranganathan
2E9D28B343204C6...
Prakash Ranganathan

DocuSigned by:
[illegible]
1EF0CCAE060C410...
Hassan Reza

DocuSigned by:
Wen-Chen Hu
EE71D886B1C04D0...
Wen-Chen Hu

This document is being submitted by the appointed advisory committee as having met all the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

DocuSigned by:
Chris Nelson
2E6AF000C733409...
Chris Nelson
Dean of the School of Graduate Studies

12/1/2023

Date

PERMISSION

Title	Benchmarking Federated Learning Frameworks for Aviation Network Data
Department	Computer Science
Degree	Master of Science (MS)

Barathwaja Subash Chandar
6 December 2023

ACKNOWLEDGMENTS

I sincerely thank Dr. Prakash Ranganathan for his valuable mentorship and constant support, which enabled me to prepare my work successfully. I would also like to extend my gratitude to the members of my thesis committee, particularly Dr. Hassan Reza, whose expertise in software engineering provided a fresh perspective that also influenced the development of my research, and Dr. Wen-Chen Hu for providing me with valuable industrial knowledge and ongoing support from the project's perspective.

I am thankful to the Federal Aviation Administration (FAA) for funding this research work (A44), and I appreciate the support received from the School of Electrical Engineering and Computer Science (SEECS) at the University of North Dakota (UND). Additionally, I appreciate Dr. William Semke, Professor of Mechanical Engineering for his guidance. I am truly grateful for the continuous support and encouragement provided by my relatives, and friends, without which this work would not have been possible. I also want to express special thanks to my parents (Mrs. Rajalakshmi and Mr. Subash Chandar), who believed in me, supported my journey abroad, and have been instrumental in my success.

ABSTRACT

Automatic Dependent Surveillance-Broadcast (ADS-B) is an alternative technology adopted by the FAA instead of ground radar to enhance accurate navigation by relying on GPS satellites for precise aircraft position information. Factors such as jamming, multipath fading, and solar activities influence GPS data integrity issues, causing dropouts or missing data, thus affecting flight safety and navigational accuracy. To mitigate such potential GPS dropout-related incidents, there is a need for robust data-driven models. This thesis focuses on multiple studies: (1) investigate five distinct machine learning (ML) models to impute missing data on ADS-B/GPS information; (2) design a federated learning (FL) framework for aviation network data; and (3) conduct a benchmarking study to validate multiple quality attributes for the proposed aviation Fed-CPS framework. Preliminary results indicate (a) k-NN yields better accuracy over other ML models (Bayesian Ridge, Random Forest, AdaBoost, Extra Tree, and k-NN) even at the highest missing rate of 30%; (b) deployment of LSTM and k-Means in a federated setting indicate that LSTM results in both MAPE and computation run-time savings. Specifically, LSTM shows (i) performance-per-dollar of 1.5 times (client) and 0.5 times (server) than k-Means and (ii) energy-efficiency-per-watt of 1.5 times (client) and 0.5 times (server) than k-Means.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATC	Air Traffic Control
ATM	Air Traffic Monitoring
ATS	Air Traffic System
ADS-B	Automatic Dependent Surveillance-Broadcast
ARIMA	Autoregressive Integrated Moving Average
CPS	Cyber Physical System
DL	Deep Learning
DNN	Deep Neural Network
EEW	Energy Efficiency-per-Watt
EM	Exponential Maximization
XML	Extended Markup Language
ES	Extended Squitter
FAA	Federal Aviation Administration
FedAvg	Federated Averaging
Fed-CPS	Federated Cyber Physical System
FL	Federated Learning
GPS	Global Positioning System
HITL	Hardware-in-the-Loop
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IID	Independently and Identically Distributed
JSON	Javascript Object Notation
k-NN	k-Nearest Neighbor
LR	Linear Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error

MAR	Missing at Random
MCAR	Missing Completely at Random
MNAR	Missing Not at Random
MLP	Multi-Layer Perceptron
Non-IID	Non-Independently and Identically Distributed
NMAR	Not Missing at Random
PD	Performance-per-dollar
RADAR	Radio Detection and Ranging
RF	Random Forest
RNN	Recurrent Neural Network
RPC	Remote Procedure Call
RMSE	Root Mean Squared Error
SSR	Secondary Surveillance Radar
SITL	Software-in-the-Loop
SVM	Support Vector Machine
UML	Unified Modeling Language
UAT	Universal Access Transceiver
UAM	Urban Air Mobility
UI	User Interface

CHAPTER I

INTRODUCTION

1.1. Motivation and Problem Statement

Cyber-Physical Systems (CPS) are intelligent systems integrating physical devices with computational components to enable real-time monitoring and control, which provides more scalable, safe, secure, and robust systems. Before CPS, these systems used to run independently with little or no connectivity, making it challenging to make decisions and carry out tasks efficiently [1]. The main characteristic of CPS is the seamless exchange of data between multiple systems that need to detect and act upon environmental changes quickly and accurately in real time to achieve a particular goal.

CPS requires complex design, testing, and monitoring while managing complexity to deliver effective performance with redundancy, fault tolerance, and reliable communication. Moreover, CPS should use standardized protocols and interfaces created with interoperability to exchange information with other systems or devices [2], [3]. CPS is used in several domains, including manufacturing, healthcare, transportation, and energy.

Figure 1 shows the components associated with CPS for aviation. These components are interconnected to enhance routine maintenance, reduce maintenance expenses, decrease fuel costs, and increase flight safety. One of the key avionics systems important for flight safety and navigational accuracy is the Global Positioning System (GPS).

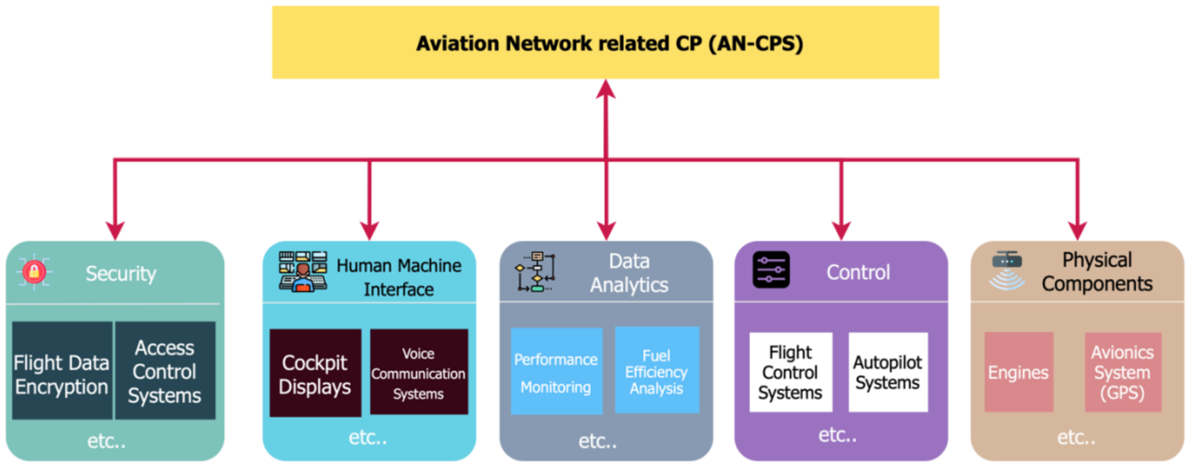


Figure 1. Components of aviation network-related Cyber-Physical Systems.

GPS is a navigational system that uses satellites to help find the device location. It operates based on the principle of trilateration, where GPS receivers measure its distance from multiple satellites to determine its location accurately [4]. Figure 2 highlights some applications that use GPS in aviation, including Flight Management Systems, Weather and Environment Monitoring, Auto-Pilot and Flight Control Systems, Search and Rescue, and Collision Avoidance (ADS-B) [5]. As the number of commercial and private airlines increased, the Federal Aviation Administration (FAA) introduced a new technology called Automatic Dependent Surveillance-Broadcast (ADS-B). This technology uses GPS and onboard sensors to accurately determine the location of aircraft to enhance the flight safety. However, despite these advancements, there are various reasons that can cause data integrity issues. Addressing these challenges is essential for maintaining the effectiveness of ADS-B and, in turn, enhancing flight safety and navigational accuracy.

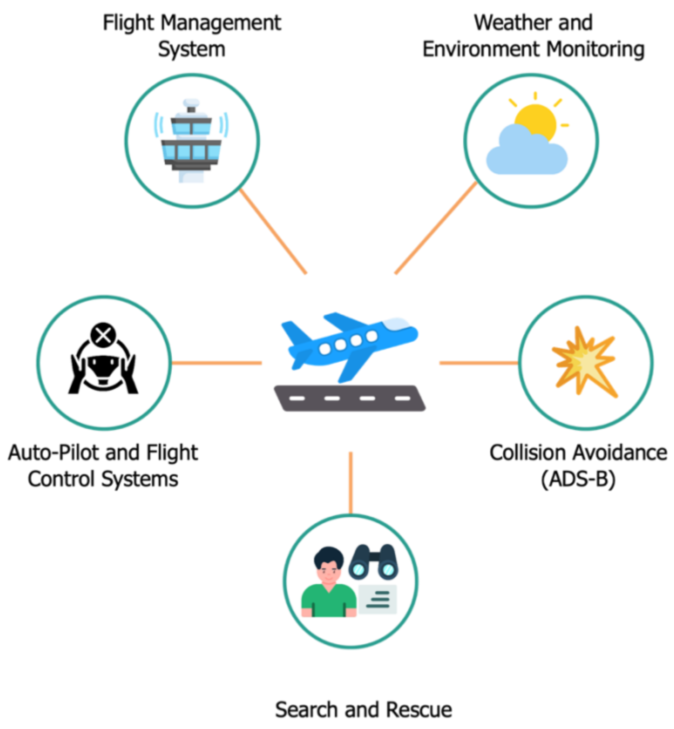

Flight Management System
Weather and Environment Monitoring
Auto-Pilot and Flight Control Systems
Collision Avoidance (ADS-B)
Search and Rescue

Figure 2. Importance of GPS in aviation.

1.2. Research Questions

The aviation industry heavily relies on the GPS and several factors contribute to data integrity issues, disrupting flight safety. Thus, this thesis focuses on hypothesis (H_0) aimed to investigate whether machine learning-based imputation and Federated Learning framework aids in solving GPS/ADS-B integrity issues. The research will primarily address two problem statements:

1. How can advanced ML-based techniques be effectively utilized to impute the missing data in GPS/ADS-B system?
2. In what ways can the Federated Learning framework enhance the privacy and improve the flight safety and navigational accuracy?

1.3. Organization of the Thesis

The outline of this thesis is as follows:

Chapter II provides an overview of the ADS-B, including the ADS-B data format and associated vulnerabilities. A broad review is conducted to assess the vulnerabilities present in the ADS-B.

Chapter III investigates the reasons for GPS integrity and the importance of imputation to avoid potential collision incidents. This chapter also highlights the importance of imputation by demonstrating the use of ML algorithms to predict the missing data at different missing rates ranging from 10% to 30% on the ADS-B/GPS dataset.

Chapter IV overviews how the aviation landscape continues to evolve and the significance of simulation in this modern air traffic system. Thus, this chapter discusses the state-of-the art review on existing architecture frameworks used in aviation and different domains. Finally introduces the design of a Federated Learning framework for aviation network data.

Chapter V provides the practical implementation of the architecture mentioned above in a simulation environment and conduct a benchmark to validate multiple quality attributes for the proposed aviation Fed-CPS framework.

Chapter VI concludes by providing future work and open research directions.

CHAPTER II

BACKGROUND AND OVERVIEW OF ADS-B

2.1. Background of ADS-B

Before ADS-B was rolled out, the aircraft depended on the Primary Surveillance Radar (PSR), which uses radio waves sent out in a specific direction. When encountering an aircraft, it bounces back if it is in operating range. The time difference between transmission and reception is considered to determine the distance. Furthermore, Secondary Surveillance Radar (SSR) is an advanced RADAR system used to collect additional information, such as the identification code of aircraft (transponder code) and altitude. This information is then sent to the Air Traffic System (ATS) to identify aircraft and manage air traffic more effectively. The transition from Radio Detection and Ranging (RADAR) to ADS-B as a primary source for surveillance has significantly improved aviation safety, real-time tracking, and effective ground operations [6]–[8].

ADS-B largely depends on the Global Navigation Satellite System (GNSS) and other sensors to determine the aircraft location. Unique aircraft id, altitude, and additional critical information are transmitted at 1Hz [9] to nearby planes and ground air traffic control (ATC). Therefore, it improves air-traffic surveillance by providing higher accuracy, enhancing the situational awareness, and decreasing the risk of mid-air collisions compared to PSR and SSR. It also aids in preventing runway incursions and enables more precision of ATC management, especially in remote regions lacking radar coverage. Consequently, the ATC system can now effectively handle large volumes of aircraft, facilitating the implementation of optimized departures and arrival procedures. The rapid advancements in artificial intelligence (AI) have led to the utilization of ML models in forecasting flight trajectories [10]–[16], and fuel and emissions evaluation [17]–[19].

2.2. ADS-B Categories

ADS-B is categorized as ADS-B Out and In, as shown in Figure 3. The Extended Squitter (ES) and the Universal Access Transceiver (UAT) are the two ways that ADS-B Out transmits data. The purpose of it is to broadcast aircraft metadata, including flight unique identification number, position information (latitude, longitude, altitude), and velocity. All the aircraft must mandatorily have ADS-B Out and transmit data either through the 1090MHz frequency spectrum for aircraft operating in Class A airspace anywhere in the world or the 978MHz frequency when aircraft operating below Class A airspace within the United States only. Meanwhile, the aircraft having optional ADS-B In, will obtain information from nearby operating aircraft in that region.

2.3. Understanding of ADS-B Message Structure

The structure of the ADS-B 1090ES message is shown in Figure 4. It consists of five fields, with a length of 112 bits, described below:

- **Downlink format (DF)** – This determines the type of message. All the ADS-B transponders will use the download format beginning with a decimal value of 17 (i.e., 10001 in binary). In contrast, the non-transponder-based ADS-B starts with a decimal value of 18 (i.e., 10010 in binary).
- **Transponder Capability (CA)** – This indicates the capabilities of the transponder level, either airborne or on-ground status.
- **International Civil Aviation Organization (ICAO)** – a distinct aircraft identification code (hex code) given to each aircraft.
- **Message Extended Squitter (ME)** – The field contains data about the aircraft, including its surface position, altitude, velocity, and status.

- **Parity Field (PE)** – It holds 24-bit cyclic redundancy check (CRC) information to identify whether messages are corrupted.

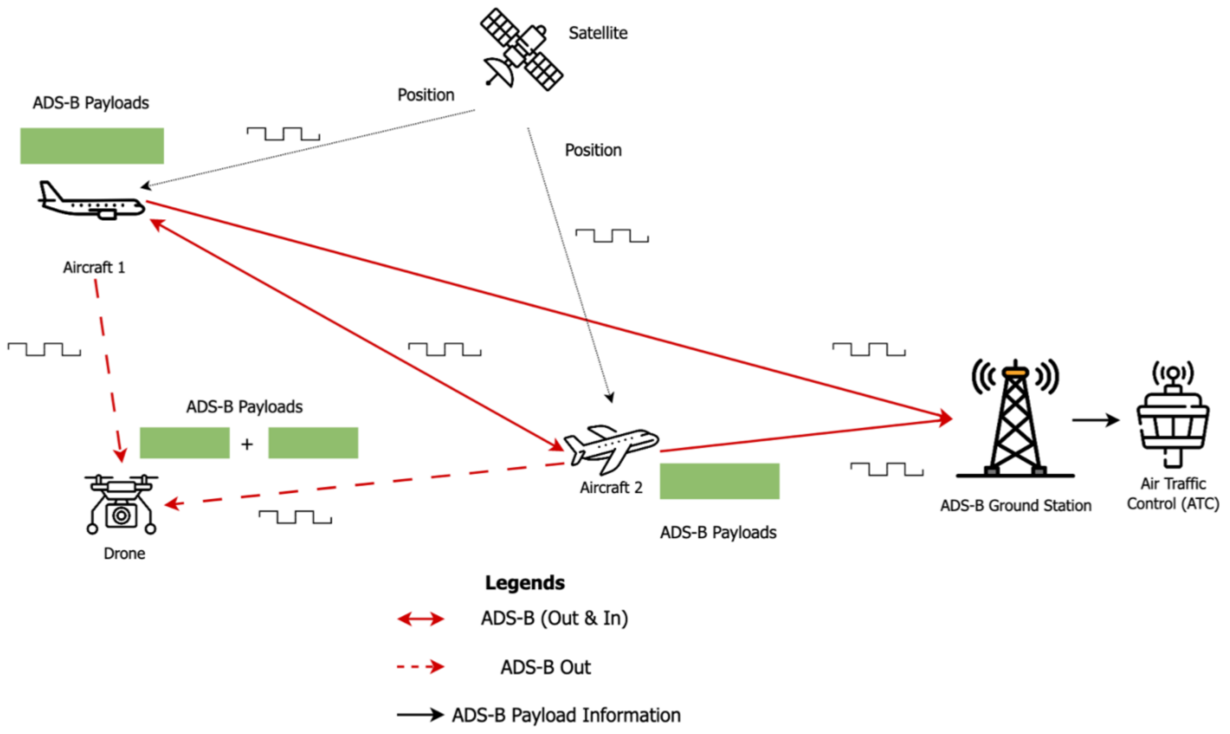


Figure 3. An Overview of the ADS-B and GPS communication between aircraft, ATC, and UAV.

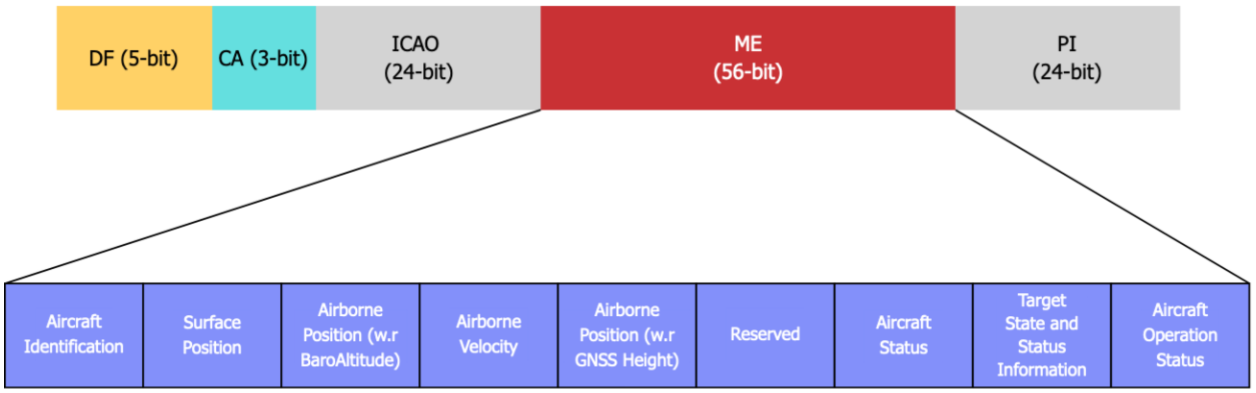


Figure 4. ADS-B data packet layout.

2.4. Vulnerabilities in ADS-B

ADS-B data packets are intended to be transmitted at 1Hz, but "dropouts" or incomplete transmissions might occur occasionally [9]. Several potential causes exist for these problems, including GPS receiver issues, faulty ADS-B transceivers, or other issues [20]. Another possible factor is the security risks associated with its open and unencrypted broadcasting within a known frequency range, exposing well-known data formats. This could potentially lead to passive or active attacks, originating from within and outside the ATC system. Active attacks include intentional or unintentional interference [21]–[23] as well as jamming, spoofing, or message deletion or modification [24]–[29]. Passive attacks, such as eavesdropping, involve attempts to listen to the ADS-B messages of an individual aircraft without disrupting the system. This Table 1 indicates how ADS-B messages can be attacked and the consequences of doing so.

Table 1. Different Attacks in the ADS-B Data Packets

Attack Types	Impacts	References
Spoofing	• Broadcasting falsified ADS-B information that can result in mid-air collision or other dangerous situations. • Impersonating the identity of a genuine aircraft. • Capturing the ADS-B data and then replaying it.	[30]–[32]
Jamming	• Disrupting the broadcasting of ADS-B data within a designated airspace.	[33]
Message Manipulation (or) Deletion	• Manipulate the ADS-B data for a single or multiple aircraft. • Removal of specific or entire ADS-B data for certain regions.	[34], [35]
Eavesdropping	• Listening to the ADS-B data on aircraft or airspace, specifically military aircraft.	[36]

Wireless technologies are heavily used in modern aviation for communication. And the fact that ADS-B communicates with ATC in an unencrypted way makes the ADS-B protocol vulnerable. Software Defined Radio (SDR) is widely adopted method for transmission and reception of RF signals. Some of the popular SDRs include the HackRF One, USRP, and BladeRF. It is critical to identify any ADS-B flaws that might lead to GPS data integrity tampering. This information is critical for developing effective countermeasures and guaranteeing flight safety. The chapters that follow will go over the importance of machine learning (ML) and federated learning (FL) approaches in ensuring the safety and precision of GPS and ADS-B technologies.

CHAPTER III

IMPUTING ADS-B/GPS DROPOUT USING MACHINE LEARNING

3.1. Introduction

The integrity of GPS signals can be susceptible to various factors, such as natural events, satellite-related issues, environmental factors, defects, regulation challenges, and cyber threats, as illustrated in Figure 5. An example of such an incident happened with Cirrus Jet (ICAO - ad564c) near Savannah, Georgia, on February 26, 2022, where geo-altitude was not transmitting for about 20 minutes, as shown in Figure 6 (highlighted in yellow). On October 18, 2022, a similar incident occurred with aircraft DAL 2439, which was en-route from Oklahoma to Austin, USA. During the flight, the barometric pressure data did not transmit for a duration of 15 minutes [37]. The ADS-B/GPS messages should broadcast at 1Hz, but if there is a discontinuation, it is called an 'ADS-B Dropout' [9].

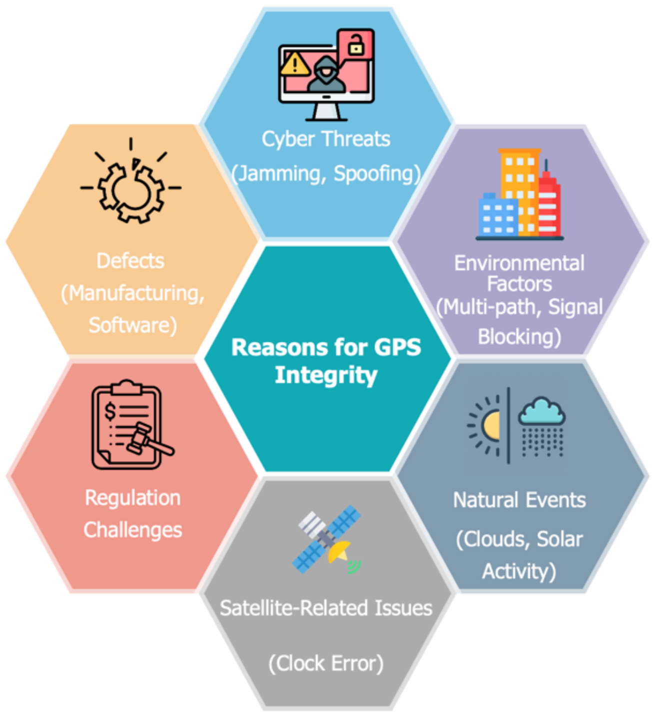
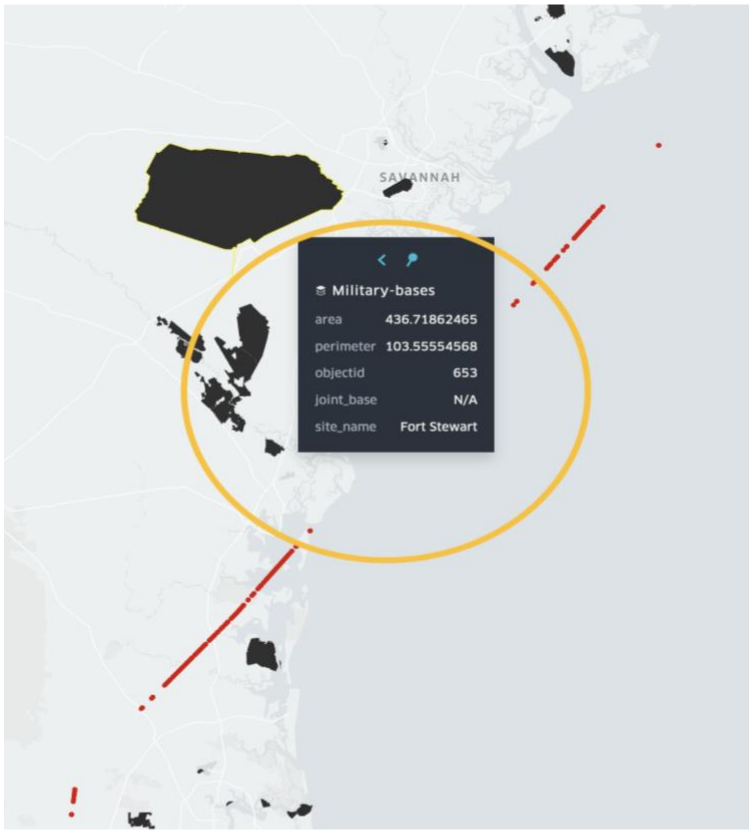
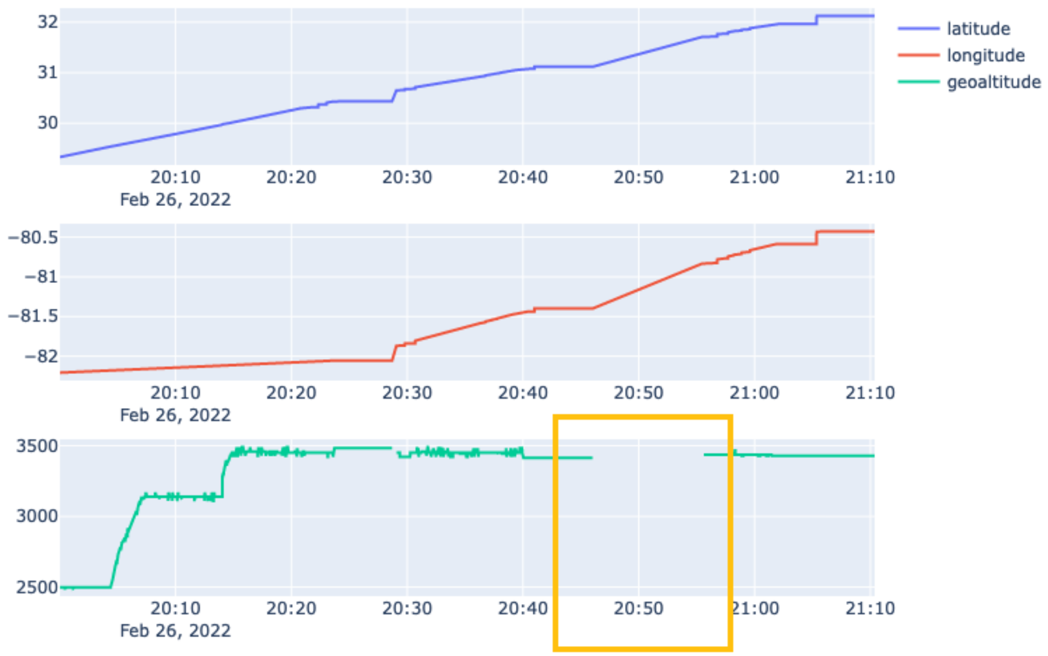


Figure 5. Categories of GPS Integrity challenges.



(a) Trajectory of the flight ICAO – ad564c



(b) Missing data packets for geo-altitude for the flight ICAO – ad564c
Figure 6. Data discrepancies captured in OpenSky Network on February 26, 2022.

The GPS positioning errors can vary significantly across different locations, resulting in incomplete or inaccurate information. To avoid potential collision incidents, it is necessary to implement robust Machine Learning (ML) algorithms that use patterns and impute trajectory points to maintain GPS integrity.

In this chapter, we will discuss the importance of imputation and how to handle missing data. Additionally, we will analyze the dataset and the various ML algorithms used for imputing GPS/ADS-B parameters. Finally, the last section will focus on the results and conclusions from the imputation.

3.2. Related Work

Every real-world application has certain incompleteness in its data points, often known as 'missing data'. Developing robust decision-making systems and providing accurate results in many study domains, such as medicine [38], engineering, and finance [39] is difficult when missing data exists. These frequently occur due to (i) measurement errors, (ii) inaccurate data entry, (iii) purposefully masking, (iv) equipment failure, or (v) cyberattacks. A thorough investigation of the patterns and characteristics is required to address these, which consumes time. Rubin [40] categorized the occurrence of missing data into (i) Missing Completely at Random (MCAR), (ii) Missing at Random (MAR), and (iii) Missing Not at Random (MNAR). The two common strategies for tackling missing information are deletion or imputation (filling in the gaps). Listwise or pairwise deletion strategies are mostly adopted, which involve the removal of the entire value or only a pair of columns that contain missing values [41]. However, these methods are feasible only when the proportion of missing entries is small [41]. Unfortunately, considering the importance of feature columns, these approaches would limit the available data, resulting in inaccurate classification or prediction [41], [42]. Alternatively, statistical, ML-based, or DL-based approaches can be used to substitute the null records through imputation. Figure 7 summarizes the potential approaches to dealing with the missing data.

3.2.1. Statistical-based Imputation

Single Imputation (SI) and Multiple Imputation (MI) are two types of imputation that fall in this Statistical-based Imputation. SI involves substituting the missing values using statistical approaches using mode [43], mean [44], median [45] or last observed carried forward (LOCF). In many studies, that is adopted only when the ratio of missing data is small numbers [46]. These methods frequently result in inaccurate distribution of the data points, which lowers the output quality. In MI, regression models are used to predict and impute missing values by considering all

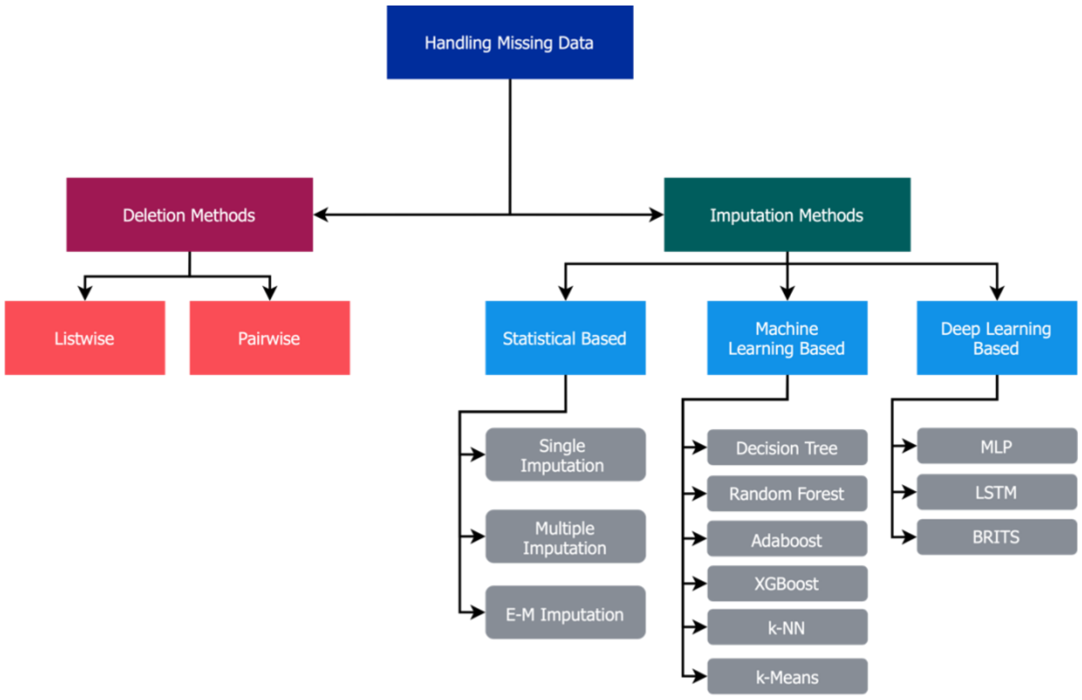


Figure 7. An Overview of handling missing data.

other variables. Multiple Imputation by Chained Equations (MICE) is an example of this category that uses this approach [47]. Expectation-Maximization (EM), an iterative two-step imputation procedure. In the first stage, the missing data is estimated by using the available information. Then, the model attempts to utilize maximum likelihood estimates on those values in the subsequent phase, and this procedure is continued until convergence is attained [48]. In the GNSS interference classification study to impute the multivariate time-series data, a combination of Autoregressive Integrated Moving Average (ARIMA) and EM was used to better estimate the missing values [49]. Most researchers initially adopted this method to do imputation [50], [51]. This imputation strategy has been observed to perform well with a smaller sample size [51].

3.2.2. Machine Learning-based Imputation

As ML gained attention, researchers started to apply iterative and regression-based approaches to impute the missing data [52], [53]. Some of the popular ML algorithms in imputation include Linear Regression (LR), Random Forest (RF), k-Nearest Neighbor (k-NN), and Support Vector (SV). RF is a well-known and popular supervised ML algorithm that aggregates the prediction results from various decision trees rather than using a single decision tree to produce accurate results [54]. A grid search was used in a study to determine the optimal combination of parameters in the RF for effectively handling missing values to achieve maximum accuracy [55]. In science and medicine, imputation techniques are applied to the liquid chromatography-mass spectrometry (LC-MS) dataset with varying percentages from 5% to 30% missing rates consisting of different imputation techniques - zero, minimum value, half of the minimum value, and mean imputation, and dimensionality reduction based-imputation - Singular Value Decomposition (SVD), Probabilistic Principal Component Analysis (PPCA), Bayesian Principal Component Analysis (BPCA), as well as supervised ML algorithms - RF and k-NN [56]. Because k-NN has a faster computation time than tree-based methods, several researchers explored the modified k-NN algorithm. For instance, Grey-Based kNN Iteration Imputation (GBKII), a modified variant of k-NN, was utilized as an alternative to the traditional Euclidean distance-based k-NN [57]. For the time-series dataset, gap-sensitive windowed kNN (GSW-kNN) was proposed for imputing on a traffic dataset [58]. On the contrary, Poulos and Valle [59] conducted a comparative analysis of LR and SV with k-NN and RF at different levels of MCAR and showcased that k-NN performed better. Additionally, Jadhav et al. carried out with only 10% to 50% missing rates, and their results demonstrated that k-NN outperforms all other ML models [60].

3.2.3. Deep Learning-based Imputation

Although ML algorithms have shown promising results, in recent years, Deep Learning (DL), a subfield of ML, has demonstrated significant promise in dealing with missing data across various fields. A Multi-Layer Perceptron (MLP) network is an example used in the energy management system and assessed with other machine learning models, showing that k-NN performed better during the peak hours and Linear Interpolation (LI) at off- and semi-peak hours [61], [62]. A Deep Neural Network (DNN) with the EM algorithm was utilized, and Root Mean Squared Error (RMSE) results showed different missing rates of 25%, 50%, and 75% [63]. Recurrent neural networks (RNNs) are also used in research when dealing with sequential time series [64], [65]. Due to challenges in the vanishing gradient points, a modified version called gated recurrent unit (GRU) was developed, which correlates well with target variables [64]. A modified version of this model, GRU-D, also demonstrated improved results but ended up with space and time complexity [64]. As a result, this algorithm works only on specific datasets based on the recent data point and the global average. To overcome this problem, Bidirectional Recurrent Imputation for Time Series (BRITS) is based on taking insights from forward and backward values was considered [66]. A modified version of Bidirectional RNN was proposed, generating synthetic data using temporal and non-temporal information. Results showed a score above 50%, even with 90% missing rates, compared to other DL models as per reference [67].

In this chapter, we will gather the dataset of different flights of ADS-B/GPS data and introduce missing data at varying rates of 10%, 20%, and 30% through random sampling and use various ML algorithms, including Bayesian Ridge (BRR), Random Forest (RFR), AdaBoost (ABR), Extra Tree (ETR), and k-Nearest Neighbors (kNNR) for imputation. Our goal is to determine the most effective algorithm, even in situations involving significant amounts of missing data.

3.3. Dataset Description

In this section, we will discuss the details of the dataset, highlighting its key features. Additionally, the necessary steps required to improve the quality and relevance of the dataset through preprocessing will be addressed.

3.3.1. Dataset Collection

Data from the OpenSky Network was downloaded using a Python script [68]. This information was arranged by day and hour in each parquet file, a columnar storage file format optimized for big-data processing. The downloaded data was then organized daily and hourly in individual parquet files from February 19 to 27, 2022, containing six flights that resulted in approximately 60,100 data points. Table 2 illustrates the features column used for this study and the corresponding sample data recorded for an aircraft in the OpenSky database.

Table 2. Dataset description of the OpenSky dataset

Field Name	Field Purpose	Sample Data Format View
time	The Unix (epoch) timestamp that OpenSky ADS-B Receivers recorded when an aircraft was nearby.	1479957078
icao24	The 24-bit ICAO transponder ID to track aircraft.	780db8
lat	Last known latitude of the aircraft in decimal degrees.	118.59931
lon	Last known longitude of the aircraft in decimal degrees.	22.916793
geo-altitude	The actual height of aircraft above sea level in meters.	8839.2

3.3.2. Dataset Preprocessing

Pre-processing the data involves transforming the raw data into a more usable and effective format suitable for further processing steps. In this scenario, each flight is divided into individual trips as a part of the pre-processing step. A threshold limit of 15 minutes was chosen to define the trips, as the data obtained had challenges in accurately distinguishing between dropouts due to ADS-B receivers being turned off for shorter durations. The segmented trips of each flight were then saved as a separate parquet data file. These parquet data files containing each flight's trips were utilized to train the models. Figure 8 provides an overview of the ML framework for imputing ADS-B/GPS dropout data. The initial phase is to import all the parquet files segmented based on trip intervals using pySpark. Pandas [69] is a popular Python data analysis and manipulation library. However, due to its single-threaded nature it faces performance-related issues, which constrain its ability to leverage multi-core processing for parallel task execution. As a result, pySpark [70] was chosen over other libraries [71]. Next, the data is cleaned to create a clean flight trip by eliminating duplicate entries. Then the missing values are introduced randomly from 10% to 30%.

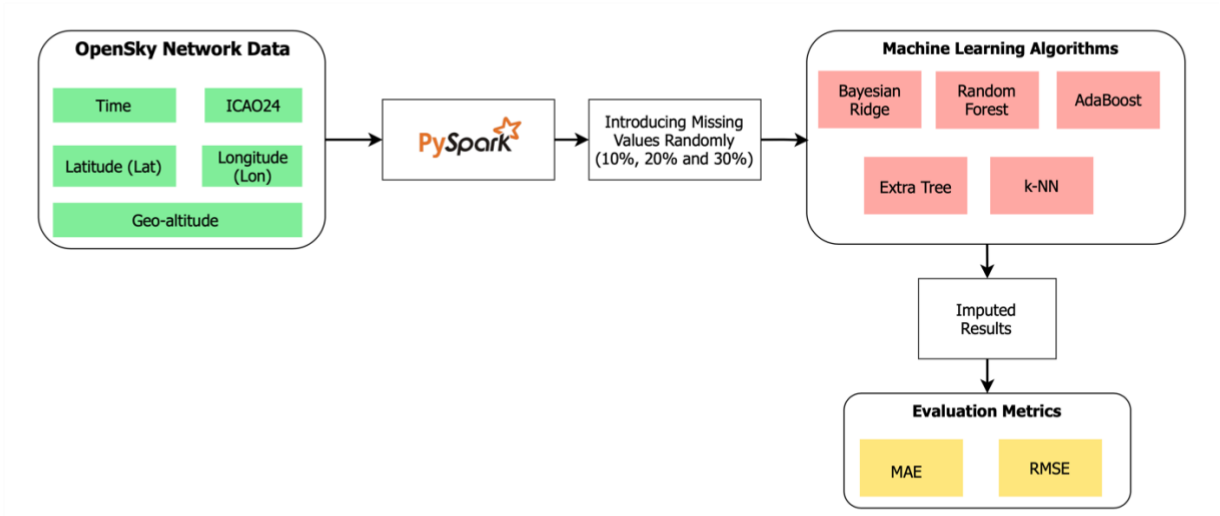


Figure 8. Overview of ML framework.

3.4. Machine Learning Models and Evaluation

In this study, imputation is performed using scikit-learn [72], a well-known ML library. Five different regression machine learning models—BRR, RFR, ABR, ETR, and kNNR — are utilized for imputation, which uses Iterative Imputer [73] and kNNImputer [74] functionality.

3.4.1. Bayesian Ridge Regression Imputation

The Ordinary Least Squares (OLS) strategy is used to fit the model, and the error is minimized by adjusting the *w* coefficients.

$$min \sum_{i=1}^{M}(y_i - w^T x_i) \qquad \text{.... Equation 1}$$

Equation 2 shows the dependent variable (y), the coefficients (w) and M instances (x_i). However, there are instances when this method overfits, thus overcoming this regularization (L2 regularization) using a Gaussian distribution to lower the least-squares integrating regularization element [75]. Equation 3 represents the formulae for Bayesian Ridge Regression, where the X is the matrix of all instances.

$$y \sim \mathcal{N}(w^T X, \sigma^2) \qquad \text{…. Equation 3}$$

3.4.2. Random Forest Regression Imputation

Random Forest for imputing missing information has gained attention in several disciplines. Several decision trees are created during training, based on the bootstrapping of the observed data samples. A random subset of features is selected during each split of individual decision tree. This is repeated until all the decision trees are generated with only observed data. These trees constructed from observed data are then used to replace these missing values [55].

3.4.3. AdaBoost Regression Imputation

AdaBoost (Adaptive Boosting) is another ensemble-based model that uses boosting to strengthen weak learners and produce more accurate regression results. Freund and Schapire introduced this first boosting ensemble model [76]. AdaBoost is sensitive to noise, but with repeated iterations, it may still strengthen poor learners. Similar to this, the performance of these weak learners is estimated after they are imputed iteratively. The final imputation stage is achieved through a weighted combination of the outputs from each weak model.

3.4.4. Extremely Randomized Trees Regression Imputation

Extremely Randomized Trees or Extra-Trees Regressor (ETR) is an ensemble approach and performs like a RF algorithm in which the trees are created based on random subsets of features that use a random threshold for each node split and finally combine to generate the output [77]. Its primary advantage over random forest is the reduction of bias and variance.

3.4.5. k-Nearest Neighbors Regression Imputation

It is often called as “lazy learner” because it does not have any function to learn from training datasets but memorizes them instead. Using distance-based approaches, it estimates the empty records based on the difference between the target and the other values [57]. Below is the pseudo-code for the k-NN imputation implementation. In our scenario, k-neighbors from are varied from 1, 2, 3, 4, 5, 10, 20, 30, and 50 with weights as uniform and distance.

Algorithm 1 K-Nearest Neighbors (KNN) Imputation

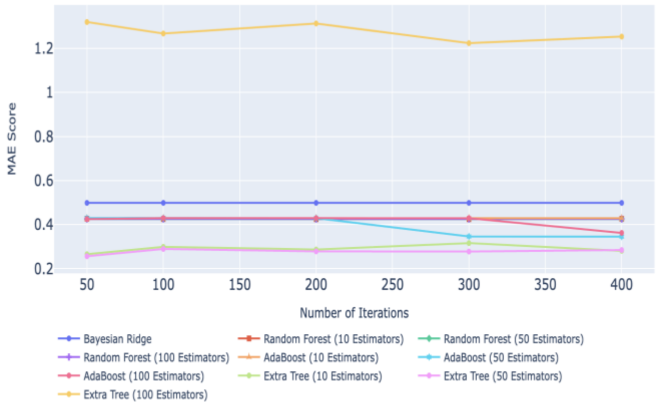
```
1: Select 'k' neighbors to be consider
2: Normalize all the feature columns
3: Select the column which needs to be imputed
4: Compute the Euclidean or Manhattan distance between the rest of the
   values of the variable and all other variables
5: Select the k variables with lower Euclidean distance
6: For the values of these variables that are in the same row as the value to
   be imputed, calculate the weighted mean and use it as the imputation
   value.
```

3.5. Results

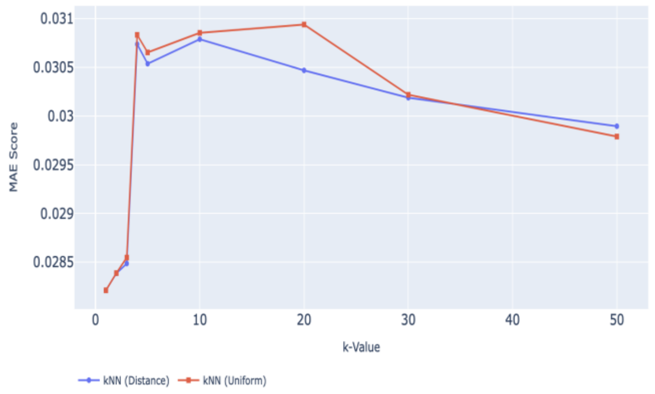
To estimate the missing values for the three feature columns—latitude, longitude, and geo-altitude; the RFR, ABR, ETR with ‘n_estimators’ set to 10, 50, and 100. Additionally, the kNNR was applied with k values ranging from 1 to 50, using both uniform and distance weights. The imputation process was conducted for different ranges of missing rates from 10% to 30%. Evaluation metrics such as the mean absolute error (MAE) and root mean square error (RMSE) were used, defined in Equations 4 and 5 respectively.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i| \qquad \text{… Equation 4}$$

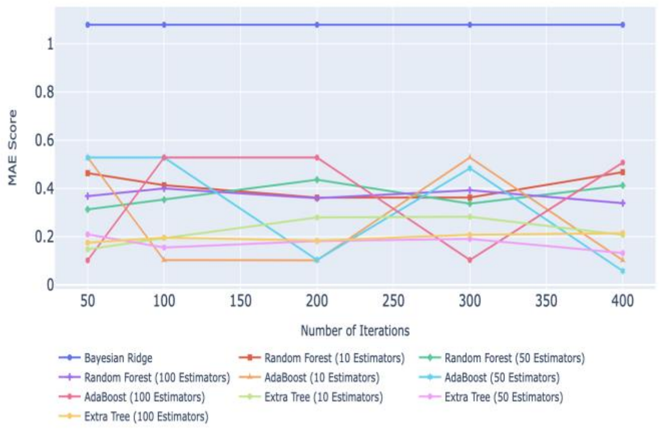
$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|x_i - \hat{x}_i|} \qquad \text{… Equation 5}$$



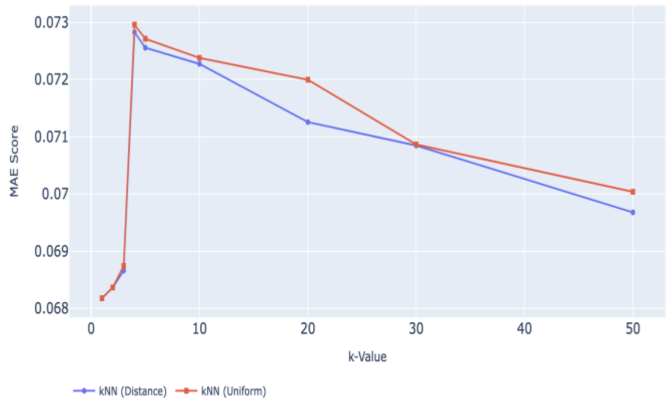
(a) For Latitude – BRR, ABR, RFR, ETR



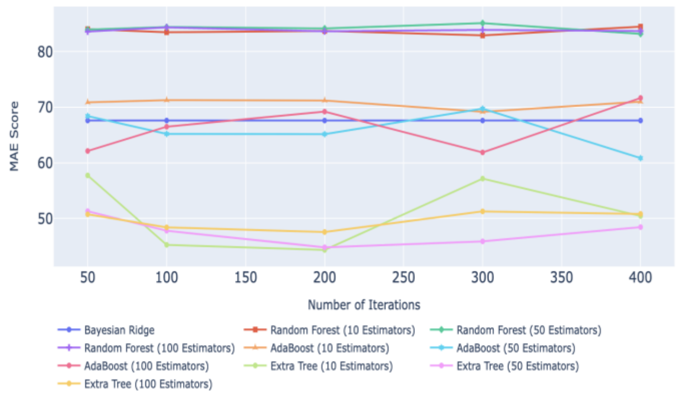
(b) For Latitude – kNNR



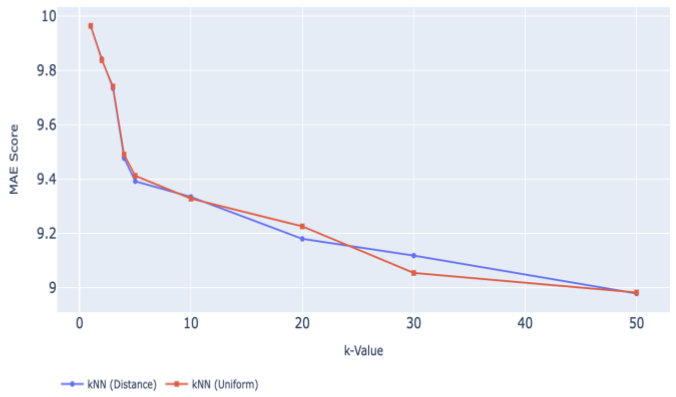
(c) For Longitude – BRR, ABR, RFR, ETR



(d) For Longitude – kNNR

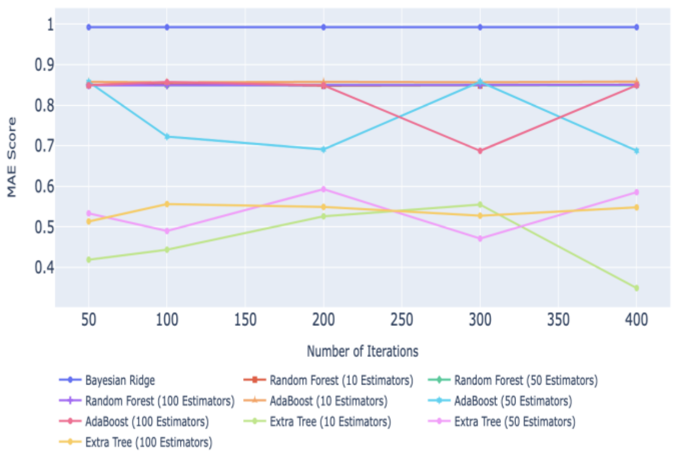


(e) For Geo-altitude – BRR, ABR, RFR, ETR

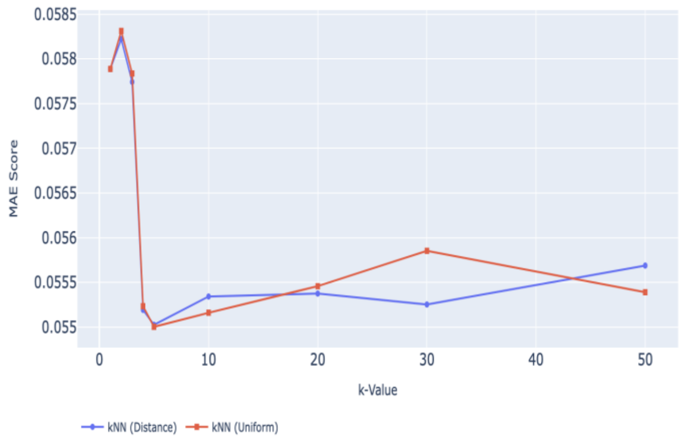


(f) For Geo-altitude – kNNR

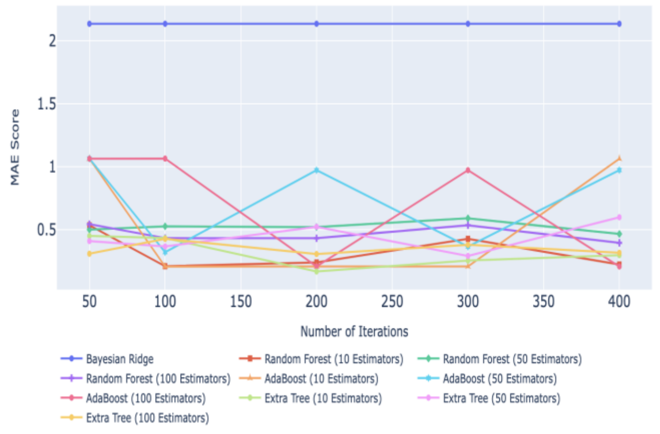
Figure 9. Comparison of MAE Score for Latitude, Longitude, and Geo-altitude at 10% missing ratio.



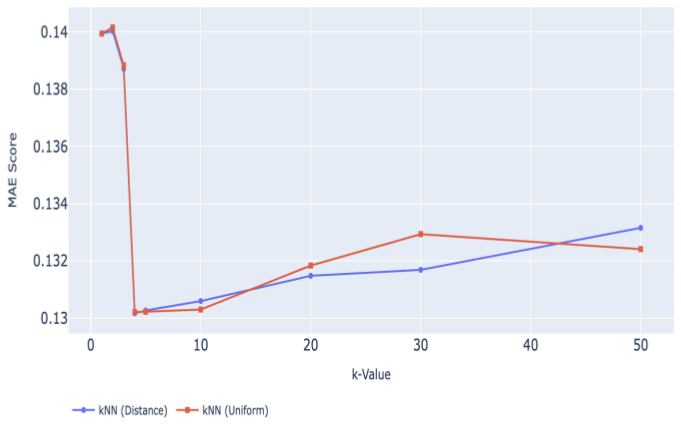
(a) For Latitude – BRR, ABR, RFR, ETR



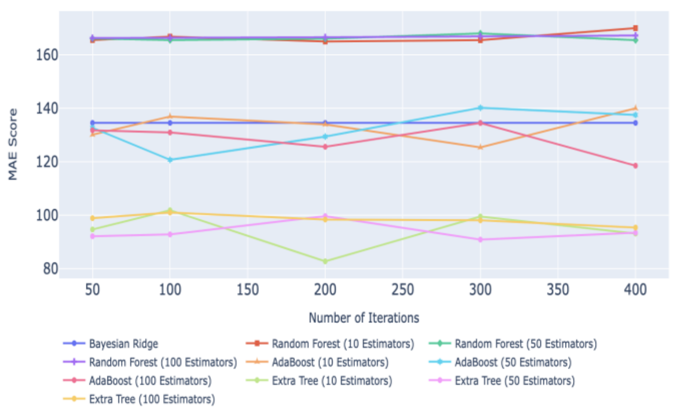
(b) For Latitude – kNNR



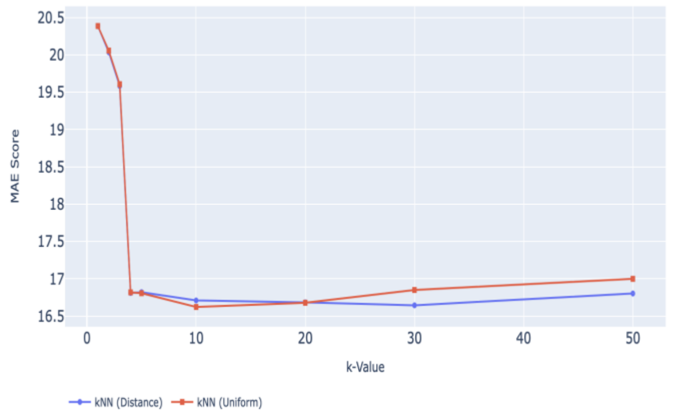
(c) For Longitude – BRR, ABR, RFR, ETR



(d) For Longitude – kNNR

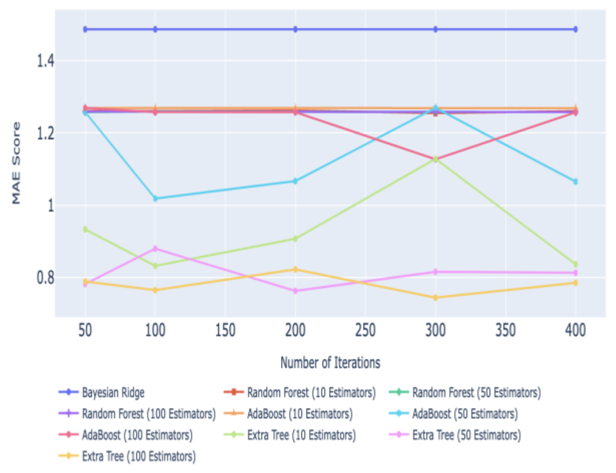


(e) For Geo-altitude – BRR, ABR, RFR, ETR

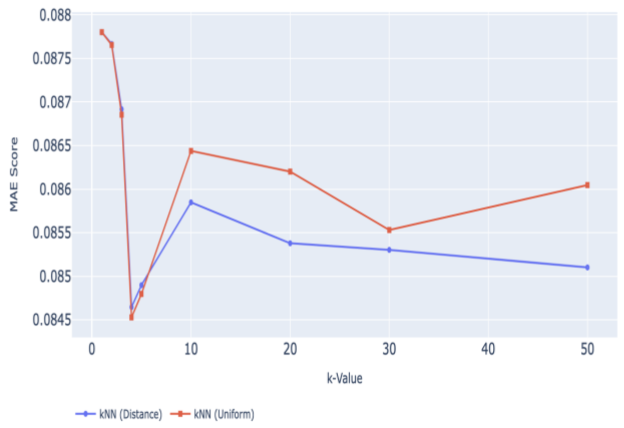


(f) For Geo-altitude – kNNR

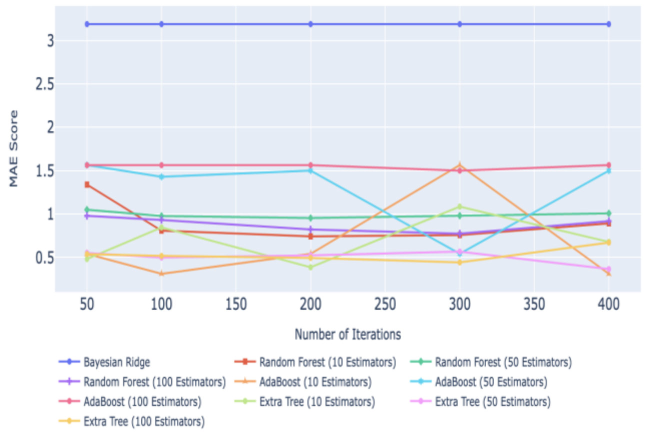
Figure 10. Comparison of MAE Score for Latitude, Longitude, and Geo-altitude at 20% missing ratio.



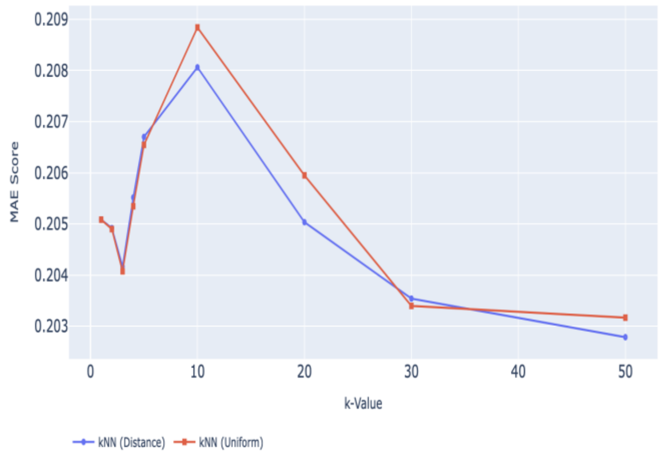
(a) For Latitude – BRR, ABR, RFR, ETR



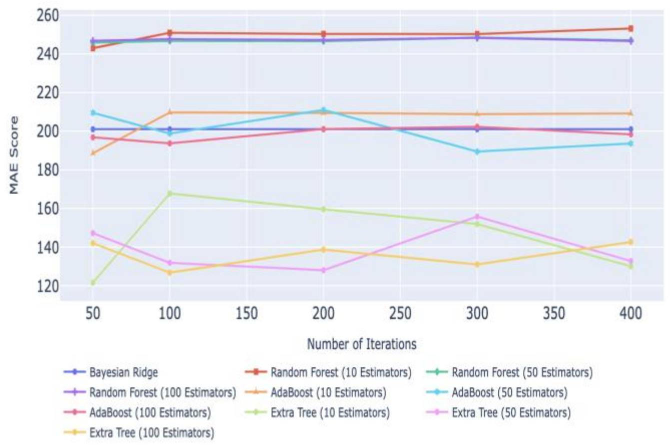
(b) For Latitude – kNNR



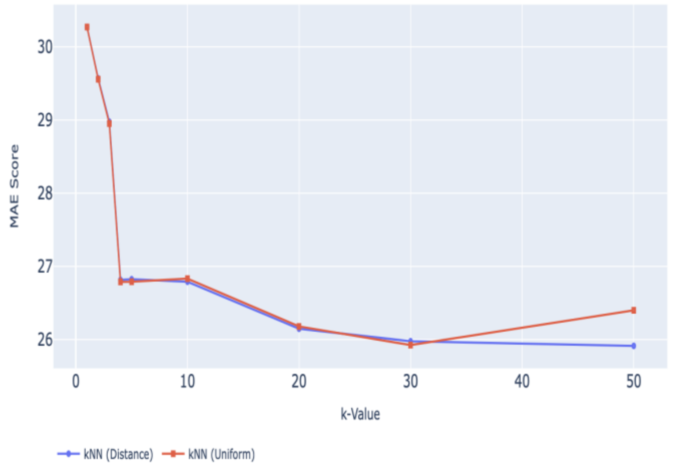
(c) For Longitude – BRR, ABR, RFR, ETR



(d) For Longitude – kNNR



(e) For Geo-altitude – BRR, ABR, RFR, ETR



(f) For Geo-altitude – kNNR

Figure 11. Comparison of MAE Score for Latitude, Longitude, and Geo-altitude at 30% missing ratio.

Figure 9, Figure 10 and Figure 11 shows the comparison chart of MAE scores vs. number of iterators or nearest neighbors, varying the imputation rates from 10% to 30% for latitude, longitude, and geo-altitude. When the dataset is randomly imputed for 10%, the MAE score of Bayesian Ridge Regressor remains consistent across the iterations. On the other hand, the Extra Tree Regressor performs better at a lower number of estimators at 10 and 50 in contrast to other tree-based models - Adaboost and Random Forest. However, increase in the number of estimators in-turn increases the MAE Score. On the contrary, k-NN consistently yielded better results irrespective of the number of k neighbors. When increasing the imputation rate to 20% and 30%, no improvement is observed in the Bayesian Ridge Regressor. The Extra Tree Regressor performs well compared to other tree-based models, requiring an average of 300 iterations to achieve lower MAE scores. However, it does not surpass the performance of k-NN imputation, where a higher number of k-nearest neighbors provides better MAE scores demonstrating effectiveness even with 30% missing data.

CHAPTER IV

FEDERATED LEARNING ENVIRONMENT FOR AVIATION NETWORK DATA

4.1. Introduction

The aviation industry is undergoing a significant technological transformation as it introduces new aircraft into the existing operational airspace through the Advanced Air Mobility (AAM) concept. This innovative approach aims to enhance the efficient movement of people and cargo between locations, particularly in underserved areas. Aircraft like Electric-Vehicle Take-Off and Landing (eVTOL), Electric Conventional Take-Off and Landing (eCTOL), and small Unmanned Aerial System (sUAS) fall within the scope of AAM. According to research conducted in 2023, the U.S. market is expected to grow by 82,000 passengers daily, with an estimated market evaluation of USD 2.5 billion yearly [78]. However, very few companies are participating and building prototypes in this complex environment. These modern avionics also rely on GPS and ADS-B to assist pilots in identifying nearby air traffic. The sharing of aircraft information through ADS-B/GPS transparently can potentially lead to various attack possibilities, including jamming and spoofing [79]–[81], false data injection [82] attacks. An example of such an incident occurred when a new 5G telecommunications system interfered with GPS signals, causing the airport to stop and halt [83], [84]. Therefore, it is crucial to effectively share the information to provide safety of aircraft [85].

Federated Learning (FL) is an active research field that maintains privacy without compromising accuracy. Unlike traditional cloud-based ML approaches, the new FL provides a better alternative to addressing the issues, enabling decentralized learning approaches on the edge and achieving privacy. Most applications have started to adopt this approach [86]–[90]. In the following sections we will discuss the existing state-of-the-art (SOTA) software architecture simulation frameworks in aviation and other domains and the current software architecture trends. Furthermore, extending this to the field of FL, the goal is to enhance privacy-preserving decentralized ML. Finally, to design a hybrid simulation framework architecture that can be easily customized and supported at a large scale, thus allowing to experimentation with various applications.

4.2. Background And Related Work

The following part outlines some of the popular software architecture styles, highlighting some of the SOTA reviews in aviation and other domains, and proposes a hybrid federated learning approach and the current trends in software architecture.

4.2.1. Software Architecture Paradigms

Software architecture refers to a conceptual framework that offers an overview of a software system design from a high level. According to ISO/IEC/IEEE 42010:2011: “the fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution” [91]. It consists of an abstract representation of software components (processing and computation), connectors (interaction), and their relationship to environmental conditions. Following are a few of the popularly used architectural design patterns:

- **Client-Server Architecture** [92] – This is the fundamental architecture style in the software paradigm, which is divided into four groups: one-tier, two-tier, three-tier, and N-tier architecture. In one-tier architecture, the business logic and data access logic are combined. In contrast, a two-tier architecture consists of two main components: the client, the graphical user interface in charge of direct user interaction and communication with the server tier to request data, and the server, which manages application data, processes client requests, and controls business logic. The three-tier architecture provides more layers of separation – presentation (interacts directly with users), application logic (applications’ business logic, processing, and functionality), and data (stores and manages application data).

- **Layered Architecture** [93] – The majority of applications use this type of architecture, sometimes it is referred as N-tiered architecture. Components associated in each layer in this architecture are connected but function independently. As this architecture uses a top-down approach to interact from one layer to another, they cannot bypass intermediary layers but must pass through all sequentially. Although this architecture is inexpensive and straightforward, it is not scalable or modularized.

- **Pipe-and-Filter** [94] – In this, the data processing is broken into multiple stages and filters. These data will flow sequentially through several phases, each of which has filters. It is responsible for transforming data and passing it on from one filter to another in a unidirectional flow. Though they are all simple to set up and can be readily modularized and changed, they are not scalable.

- **Event-Driven Architecture** [95] – This is a widely used distributed asynchronous architecture, with each component separated and executing a particular process asynchronously. It comprises of two elements – the broker and the mediator. Once an event is started in the broker, it is routed to the relevant event's channel for processing, with all these operations performed asynchronously. A mediator will coordinate several event processors to control and manage the events. Although implementing this into practice can be difficult, the performance and scalability are advantages over other architecture.

- **Microservice Architecture** [96] – This architecture pattern rapidly captured industries attention, as the individual components of a single application are broken into smaller services. Each of these services operates independently and communicates with others. In addition, each of these microservices has its presentation, application, and database service, which may be shared with other microservices.

- **Microkernel Architecture** [97], [98] – This architecture, sometimes called plug-in architecture, enables adding new functionality on top of the core modules. In other words, it divides the application logic between standalone plug-in parts and the core system. Similar to layered design, this architecture is easy to implement, but scalability and modularity are not very advantageous.

Recently, modern software architecture involving a combination of High-Level Architecture (HLA) and Data Distributed Services (DDS) [99] to connect different components of the system. These connectors act as a facilitators to exchange information between these components. This type of architecture was designed to support various applications, including FAA Simulator aircraft, and Urban Air Mobility (UAM). The existing simulation tools for Air Traffic Monitoring (ATM) concepts did not incorporate the cyber element, essential for effectively addressing cyber threats and mitigating potentially catastrophic consequences. Some of the architectures tested for their performance in terms of latency, update rate, throughput, and probability density [100] in the Air

Traffic Generator (ATG) and Multi Aircraft Control System (MACS). The results indicated that MACS performance degraded when handling 200-300 aircraft with a throughput of 37.5 KB/s, in contrast with 225 KB/s in the ATG and can handle 1200 aircraft. However, the known fact is that as the number of aircraft increases, the latency and update rate also increase, which can impact the effectiveness of the simulation. As a part of this architecture, an additional centralized message-oriented middleware was added to the ATM simulation of NASA for communication among different components [101]. Performance was evaluated to determine the effectiveness of this intermediate broker, since it reduces the need to develop separate simulator functionality. Results show that throughput and duration can further be improvised with the help of compression settings. Many applications have adopted a microservices-based architecture to improve their scalability. An example was the Space Traffic Management System, based on a NASA low-altitude UAS traffic management system [102]. In this architecture, each service communicates via Application Programming Interface (API) on a containerized platform, providing significant benefits in scalability, resiliency, and flexibility. However, the designs fail to incorporate the trade-off between maintaining resiliency and efficiency.

Utilizing the layered architecture, simulation experiments were conducted by the Aviation Security Lab [103] focusing on Avionics Full Switched Duplex Ethernet (AFDX), a high-speed data communication system used in aircraft for flight control, navigation, and communication. This testbed was built with Graphical Network Simulator-3 (GNS-3) to simulate network topologies and tested on different attack vectors, including sniffing, MITM, DoS, and message replay. This setup has limitations, as it cannot be used with HITL and does not incorporate lightweight Intrusion Detection and Prevention (IDPS) technology. Another study focused on GPS attacks [104] on

NASA's ATM testbed, which utilizes service-oriented architecture with components that communicate through message-oriented middleware. However, this testbed simulation did not assess in a multi-aircraft environment nor explore the detection and mitigation of attack models beyond false data injection. Furthermore, these simulations did not evaluate the performance under varying environmental conditions.

Developing a testbed architecture involves integrating Artificial Intelligence (AI) components, requiring a flexible, scalable, and portable solution. A reinforcement learning-based testbed architecture is a good example used in the Advanced Air Mobility (AAM) [105], to predict live traffic predictions designed using layered architecture to maximize throughput and minimize delay. This architecture allows for scalable solutions and is also suitable for different use cases but lacks standardization, which can create interoperability issues with no common framework. Another study [106] proposed a microkernel with microservice architecture on the UAV to address the lost-link problem where the microkernel architecture can extend plug-in operation on top of the core modules. Using this, the sensor data were extracted, and through message middleware, these were pushed to microservice-based applications such as Mission Planning Services. The findings show this architecture style can provide robustness, quality, and performance but lacks scalability to support multiple use cases or seamless communication and collaboration between systems.

Table 3 and Table 4 shows some existing simulation architecture in aviation and other domains. Some of them have used layered architecture system testbed with attacks includes DoS, MITM, Sniffing, and Replay attacks on the Aviation Full-Duplex Ethernet protocol [30], on the contrary, others have adopted the combination of microservices and microkernel architecture to improve the performance and re-usability [33] [34]. In some research, there were limited API-based approaches for designing aircraft operations and suggested using bidirectional state services like publish-subscribe communication patterns [40] and lightweight detection systems [29].

Table 3. Taxonomy of existing simulation architecture in aviation

Year Reference (HITL/SITL)	**Study Highlights**	**Architectural Styles**	**QA**
2023 [107] (HITL + SITL)	Explore decentralized architecture showcasing AAM's full potential, encompassing advanced U-space services, platforms for future scenarios (e.g., air cargo delivery, air taxi operations), in a co-simulation setting and study in complex traffic scenarios, examining interactions among the operator, U-space service provider (USSP), and ATC	Centralized - Decentralized	-
2023 [108] (SITL)	A simulation environment to replicate aircraft operations and test autonomous vehicles on a larger scale of hundreds of flights over areas like San Francisco-Oakland Bay.	-	Cost
2022 [109] (HITL)	Enablement of robust, flexible monitoring framework that can easily integrate with the legacy system using a distributed self-adaptive system	MAPE-K	Feasibility, Efficiency, Scalability
2022 [103] (SITL)	Conducted attacks on the AFDX protocol, which is primarily used for communicating between systems	Layered	-
2022 [110] (HITL)	demonstrating an autonomous UAV system for emergency rescue operations that identified gaps in Human and Autonomous system factors and incorporated in the existing MAPE-K	MAPE-K	Observability, Adaptability, Detectability, Trust

Table 3. Taxonomy of existing simulation architecture in aviation (Contd.)

Year Reference (HITL/SITL)	**Study Highlights**	**Architectural Styles**	**QA**
2022 [101] (SITL)	Addition of Message Oriented Middleware would help to address eliminating the reconstruction the simulation for each use case	MOM (or) PF	Latencies, Run-time Durations, Throughputs
2022 [111] (HITL + SITL)	To address unreliability in communication and the need for cost-effective solutions in swarm	L	Cost-effective
2021 [112] (SITL)	To perform false data injection (FDI) on communication sensor data and develop a rule-based machine learning system to detect and mitigate	CS	Performance
2021 [106] (SITL)	Solves the problem of lost-link communication between UAV and remote control	MS + MK	Fault Tolerance, Availability, Performance and Safety
2021 [113] (SITL)	To simulate Unmanned Traffic Management (UTM) systems for flight planning and tracking	MS	Reusable, Extensibility
2021 [114] (HITL + SITL)	A multi-drone surveillance coordination system to overcome ineffective coordination and redundant searching	H	Redundancy, Scalability
2020 [115] (SITL)	To develop an application via API with external factors (wind and pressure) and mimic the obstacles	L	Lightweight, Performance

Note: MOM–Message Oriented Middleware, PF–Pipe and Filter, L–Layered, CS–Client-Server, MS – Microservice, MK–Microkernel, H–Hierarchical

Table 3. Taxonomy of existing simulation architecture in aviation (Contd.)

Year Reference (HITL/SITL)	**Study Highlights**	**Architectural Styles**	**QA**
2020 [116] (SITL)	The containerized approach of designing UAV fleet systems and orchestration to analyze the network functions and the potential impact on wireless communication networks	SO	Scalability, Low Latency, High Reliability
2020 [117] (SITL)	To design for monitoring mission operation	SO	Low Latency, Scalability
2019 [118] (SITL)	Federated testing of aircraft electronic systems for spoofing of ACARS messages and evaluating software patches electronics	POM	Flexibility
2019 [119] (HITL + SITL)	To perform autonomous decision-making of fixed-wing UAV in a distributed fashion	L	Scalability
2019 [120] (SITL)	Visualization of robotic and drone swarms and CPS systems to better understand decision-making, and control	L	-
2019 [121] (SITL)	Simulation of FDI to detect and alert in distributed systems to reduce the simulation time	H	Fidelity / Accuracy

Note: POM–Process Oriented Model, SO–Service Oriented, L–Layered, H–Hierarchical

As part of a broader research initiative, this study also involved collecting and analyzing additional research articles from different domains beyond aviation. The aim was to expand the scope of the investigation, incorporating diverse areas that have focused on expandability [122] and performance [123] quality attributes for simulation purposes. Table 4 provides a detailed summary of the study highlights and features architectural designs incorporating the quality attributes (QA) adopted in each research work.

Table 4. Taxonomy of existing simulation architecture in different domains

Domain	**Year Reference (HITL/SITL)**	**Study Highlights**	**Architectural Styles**	**QA**
Energy	2022 [123] (HITL)	substation simulation to perform various cyberattacks (at the device, firmware, and software levels)	H	Maintainability, Scalability
	2022 [124] (HITL + SITL)	To test cyberattacks on individual DER and combined with the grid	PP + H + C	-
	2022 [125] (HITL + SITL)	Simulating the communication networks of the power systems in the IoT devices	L	Flexibility, Scalability
	2021 [126] (HITL + SITL)	To test and defend against FDI, Command Injection, MITM, and DoS multi-stage cyberattacks	-	Scalability, Automation, Maintenance
	2020 [127] (HITL)	Distributed Intrusion Detection System for the DERs built to detect attacks - DoS, ARP spoofing, and TCP SYN flood	-	Less Computational Time, Less Latency
	2019 [122] (HITL)	To simulate the power system application on Federated Environment and perform load-tripping and voltage sag attacks	H	Secure, Reliability
Other	2022 [128] (HITL)	A semi-realistic testbed of miniature vehicles enabling from designing to testing of autonomous vehicles (AVs)	L	Low Cost, Usability, Safety, Flexibility
	2022 [129] (HITL)	Digital Twin of Palfinger Crane was designed to track the Automatic Adaptation System	SO	Interoperability
	2022 [130] (HITL + SITL)	Prototype of developing and evaluating internet-enabled CPS applications	M	Extensible

Note: SO–Service Oriented, M–Modular, L–Layered, H–Hierarchical, PP–Peer to Peer, C–Centralized

4.2.2. Implementation of Federated Learning Framework

As the airspace activities and technologies continue to grow, it is important to efficiently share sensitive information, such as flight ICAO Id, latitudes, and longitudes. When such detailed information is transmitted over a network, malicious individuals may intercept and exploit it. To deal with this data confidentiality [131], [132], a new edge-based distributed learning solution known as Federated Learning (FL) [133] has started to gain attention. Each device (client) exchanges its trained model parameters without sharing the source information with the central server. Note that the main server may or may not initiate the model. All the devices participating in this will send updates of their trained parameters to the central server to perform computations and send back the updated attributes to clients. This procedure will be repeated until the model becomes stable. Figure 12 shows the diagrammatic representation of FL.

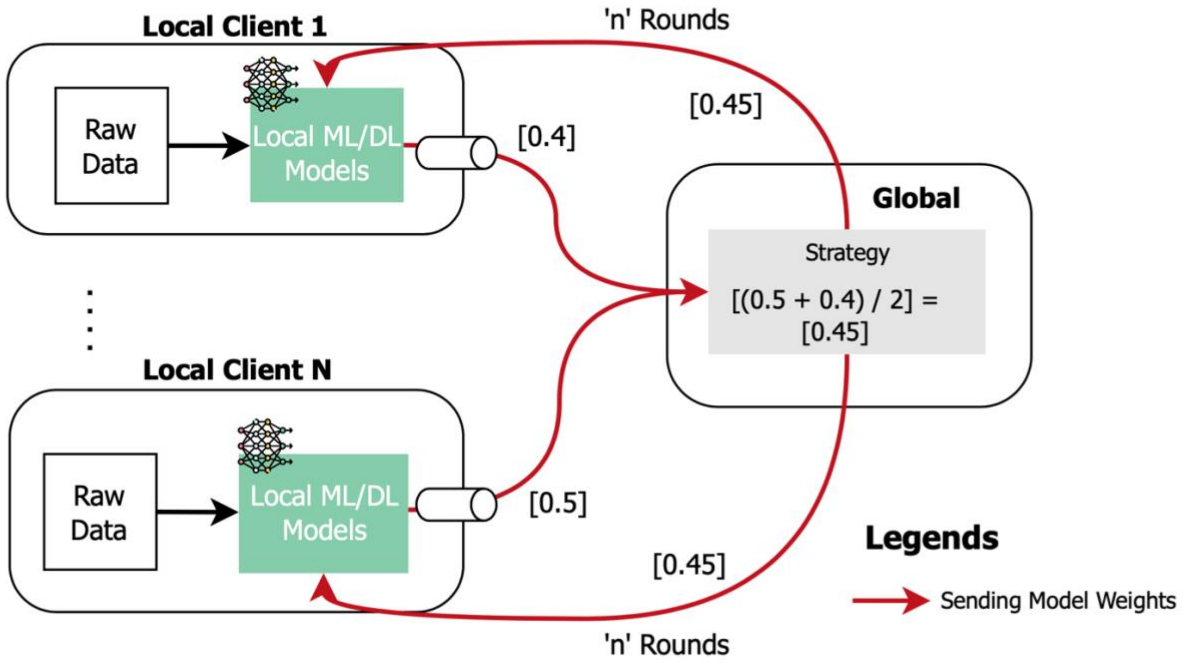


Figure 12. Diagrammatic representation of FL.

This decentralized approach runs directly on edge devices, allowing for real-time anomaly detection and forecasting with reduced latency while enhancing data security. To achieve this, creating a hybrid simulation environment that enables researchers to design and assess multiple scenarios is essential.

4.2.3. Recent Trends in Software Architecture

It is essential also to consider recent trends in software architecture that aimed to enhance the software application design, development, and deployment. Figure 13 shows the categories of individuals and organizations interested in adapting these frameworks [134]. The categories are as follows [135][136]:

- **Innovators**: This group are the first willing to take risks and implement new innovative solutions.
- **Early Adopters**: This group are willing to take the risks and experiment the solutions before others by exploring the benefits and values of the solutions provided by the Innovators.
- **Early Majority**: This group is more cautious while adopting new technological solutions. They will adopt only if these are widely adopted.
- **Late Majority**: They are reluctant to adopt innovations unless the proposed solution is well-established and widely accepted by the broader community.

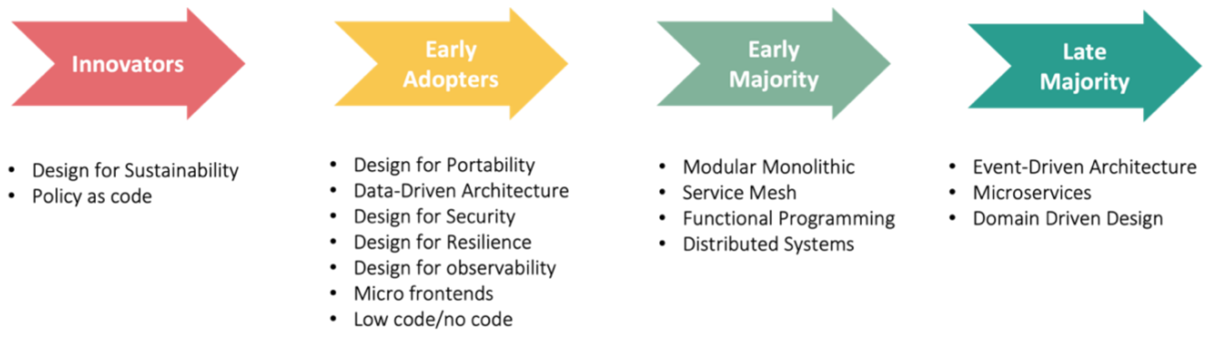


Figure 13. Recent trends in Software Architecture.

4.3. Proposed Architecture

The goal is to design a decentralized privacy-preserved approach of running the ML models with reduced latency, adaptable to different use cases (e.g. anomaly detection, forecasting) in the aviation field. Furthermore, considering the diverse airspace operational environment, a hybrid architecture referred to as Federated Cyber Physical System (Fed-CPS) is proposed that leverages the advantages of multiple architectural patterns, including microkernel, pipe-and-filter, event-based, and micro-frontends as illustrated in Figure 14.

4.3.1. Client-Side Simulation Module

The first step in client-side simulation, is the use of software like AirSim [137], customized based on each preference. AirSim is commonly used in Software-in-the-Loop (SITL) or Hardware-in-the-Loop (HITL) to simulate real-world scenarios and generate data. This data is accessed through core sensing modules, which allow the software to function and interact with various plug-in functionalities such as GPS data, altimeter, pressure, and gyroscope readings. This module utilizes a microkernel-based pattern [138] to create key functionalities as a common core; other features are developed as plug-ins to allow for modularity. Furthermore, it enhances the efficiency and performance of the system by parallel processing of core sensing modules. For our specific use-case, each client acts as individual flight, including the ADS-B data obtained such as icao24, latitude, longitude, and altitude obtained from OpenSky Network [139]. This data is temporarily stored as a file for further processing. Each of the data parameters can be broken to perform filtering operations depending on the use-case, making the ‘pipe and filter’ pattern [140], [141], particularly useful. In this scenario, the raw data are transformed using multiple filtering criteria, including removal of missing data, outlier handling, and scaling.

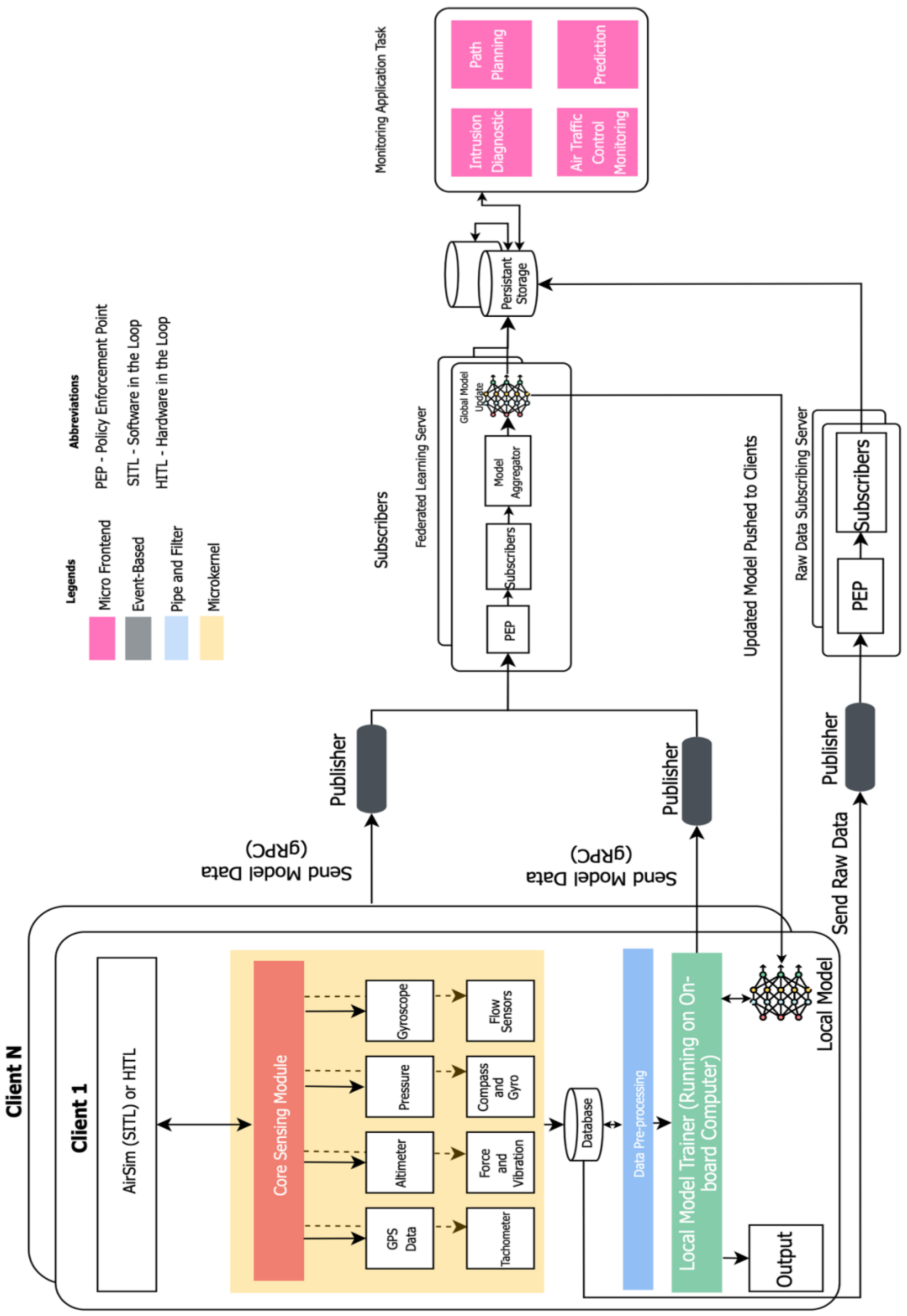


Figure 14. Proposed Fed-CPS framework for aviation networks.

After this stage, it will push the model parameters to server-side model aggregation through the communication module and train on the masked data, subsequently returning the updated model to the client-side. This iterative process continues until convergence of these models. Numerous applications in aviation, including optimization of path planning [142], [143], forecasting [144], anomaly detection [145], [146] and image-based power-line inspection [147]. In a real-world scenario, this process will run on various on-board computers, such as Raspberry Pi or NVIDIA Jetson Nano. However, for simulation purpose, Docker [148], a containerized platform, will be used to carry out these activities.

4.3.2. Communication Module

Each client is individually trained on its own data and shares only the model weights on which data was trained instead of sharing the raw information. This information can be exchanged between the clients and server using various protocols such as (i) Hypertext Transfer Protocol (HTTP) or (ii) Remote Procedure Call (RPC).

4.3.2.1. Understanding of HTTP and RESTful Architecture

The Hypertext Transfer Protocol (HTTP) is the fundamental building block of the World Wide Web. It is an application layer protocol designed to transfer all the contents. The version 1.1 was introduced in 1997 [149], was superseded by HTTP/2.0 in 2015 [150]. HTTP is generally implemented in a server, where the clients communicate with server through HTTP messages. It uses a request–response model (Figure 15) approach. To illustrate, when a web browser functions as a client and carries out various activities such as fetching data, by sending an HTTP request to a server that runs any application. The server then consolidates all relevant resources in formats - HTML, JSON, or XML, along with status information related to the requested URI. This compiled information is then transmitted to the client as a response message. For instance, assume that the client here would be the web browser which performs tasks such as fetching data by sending an HTTP request to the application server. Followed by that the server encapsulates the resources in HTML, JSON, or XML, along with status information for the requested URI, forming a response message sent back to the client.

Application Programming Interface (API) is set of definitions that adhere when interacting with other systems. Representational State Transfer (REST) commonly referred to as RESTful API is an architectural constraint on the functionality of the API. The server uses a Uniform Resource Locator (URL) to identify each resource. This URL (or request endpoint) is the path the client needs to access. Developers commonly implement RESTful APIs via HTTP to manage resources. There are five popular HTTP methods: GET, POST, PUT, DELETE, and UPDATE.

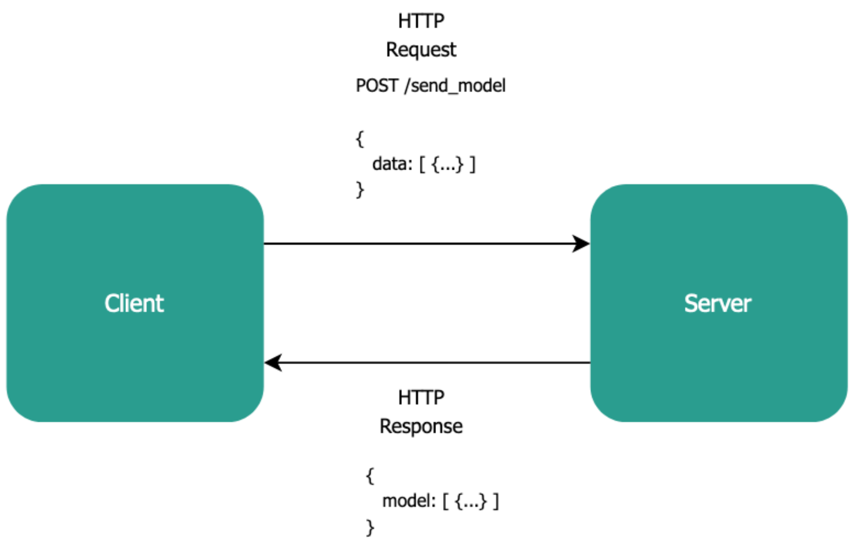


Figure 15. Representation of request-response model with REST.

4.3.2.2. RPC and gRPC

Remote procedure call (RPC) is a protocol for effective communication in early distributed applications. The fundamental idea is to separate functions within distributed systems in a server and allow other services to invoke them directly. Figure 16 illustrates the RPC functionality, comprising five components - Client, Client stub, Network, Server stub, and Server. This process begins with the client sending a request to the server with the converting messages; on the other end, the server decodes this received data and performs the corresponding operation.

gRPC is a RPC framework designed to exchange messages seamlessly, which utilizes protocol buffers (or protobuf) for serializing and deserializing messages. Messages in the protobuf format are encoded in binary, ensuring efficiency. gRPC uses HTTP/2 faster than HTTP/1.1 because of its streamlined message definitions. While traditional RPC involves a single request and response, gRPC extends the capabilities by providing three additional modes for data exchange: response-streaming, request-streaming, and bidirectional streaming RPC [151]. Therefore, gRPC based Pub-Sub [152] based pattern would be the best approach as it provides a sophisticated way to transmit and receive messages from multiple clients and to different sources.

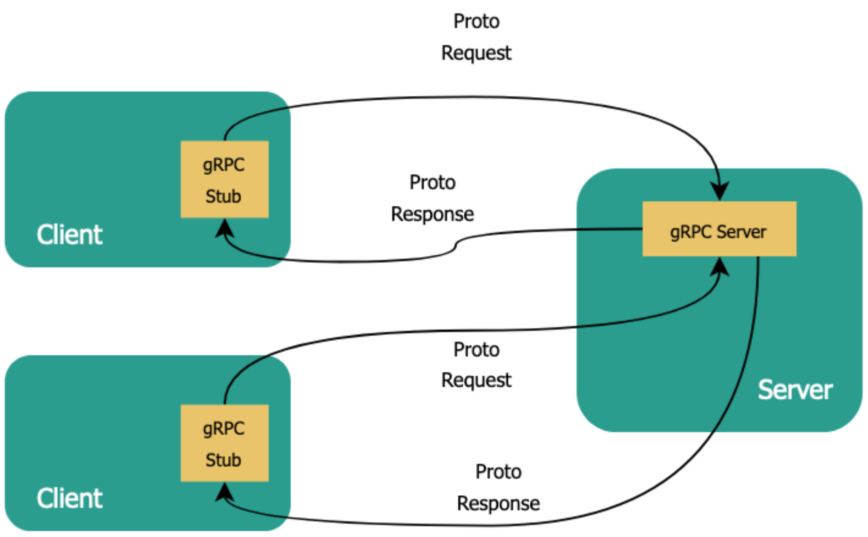


Figure 16. Representation of bi-directional communication gRPC.

4.3.3. Server-Side Computational Module

In the FL server, the model parameters were obtained through zero trust components [153], [154]: policy enforcement point (PEP), policy administrator (PA), and policy engine (PE). These components enforce security policies and access controls and act as a gatekeeper to allow or deny the requests depending on the policies defined. The PEP ensures that all access requests are authenticated, authorized, and validated before granting access to the requested resources. Observing and monitoring various aspects of the network, applications, and user behavior are collected and analyzed data to detect anomalies, potential security threats, or policy violations.

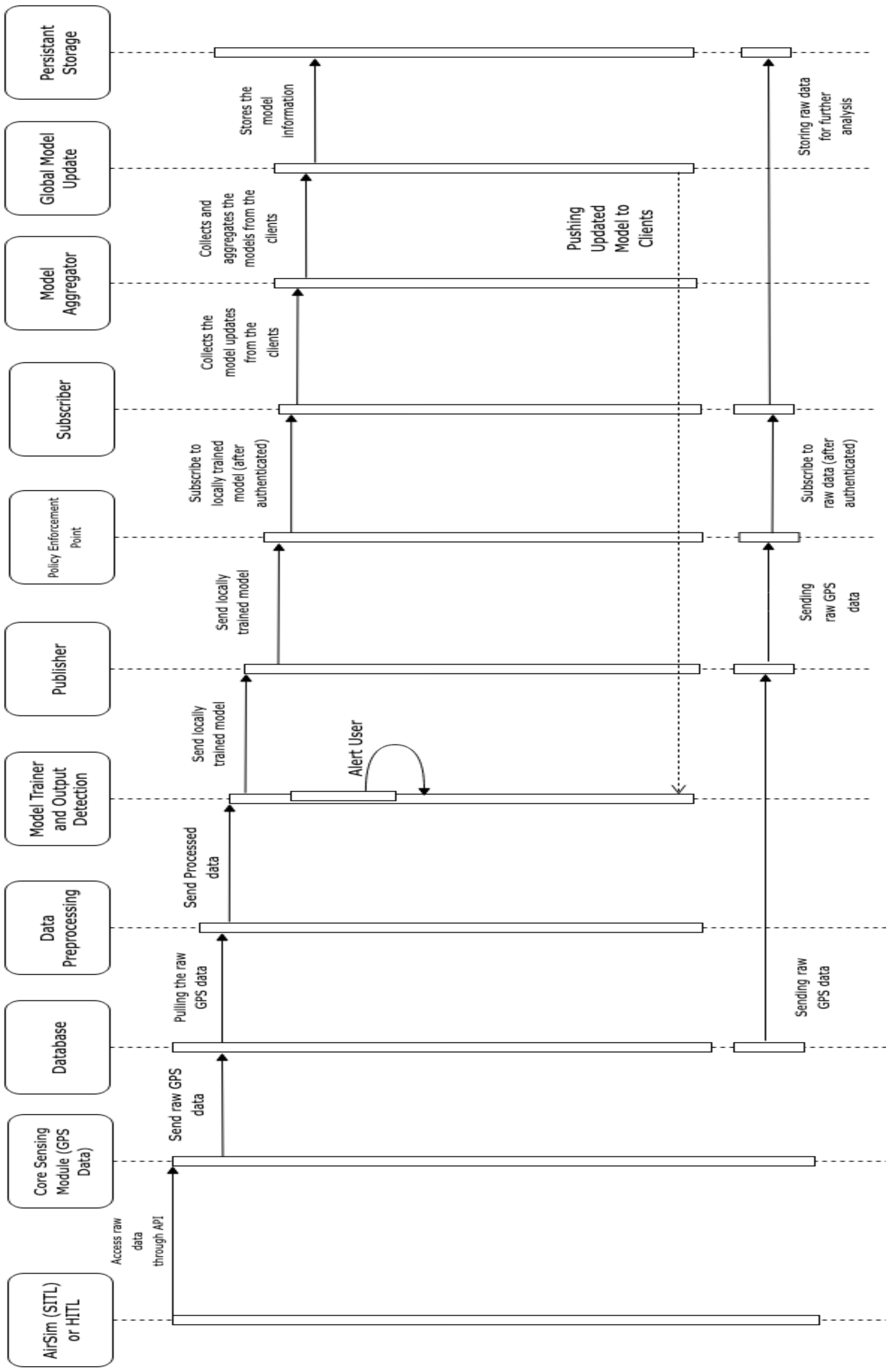
After validating and approving requests, the model aggregator (central server) collects weights from decentralized devices, ensuring user privacy and aggregating the weights to generate a new model, which in turn is sent back to corresponding participants for training. This iterative process ensures that clients learn patterns from different participating clients, and the model converges across all participants. FL is a promising paradigm for developing robust and privacy-preserving

ML models in distributed environments. Simultaneously, the model is saved to persistent storage, allowing researchers to utilize it for hyperparameter tuning [155]–[157].

4.3.4. Monitoring Application Task

To accomplish specific tasks with FL, each application team adopts a micro-frontend architectural pattern [158], [159]. This pattern divides the components into smaller, self-contained, and independent frontend modules. Each module represents a distinct User Interface (UI) section that is capable of independently developed, deployed, and scaled. This configuration reduces costs and allows teams to focus on specific tasks effectively.

The Unified Modeling Language (UML) sequence diagram in Figure 17 illustrates the interactions between components in this FL application. All the CPS core sensing modules will stream real-time data from AirSim or other simulators, and this data will be stored in the database. In this scenario, these data are already stored in the form of files, each holding individual GPS data. Each of these data will then be sent to the preprocessing stage, undergo data wrangling, and, depending on the use case, ML algorithms will be trained on this data to generate model weights. These locally trained models are sent to the centralized server, which acts as the model aggregator. This is done through a Publish-Subscribe (pub-sub) mechanism and passes through a PEP gateway for validation, verification, and authorization. Once authorized, the model aggregator collects and combines these weights to produce new weights. Then these weights are subsequently sent back to clients for local training.



Persistant Storage
Global Model Update
Model Aggregator
Subscriber
Policy Enforcement Point
Publisher
Model Trainer and Output Detection
Data Preprocessing
Database
Core Sensing Module (GPS Data)
AirSim (SITL) or HITL
Access raw data through API
Send raw GPS data
Pulling the raw GPS data
Send Processed data
Send locally trained model
Alert User
Send locally trained model
Subscribe to locally trained model (after authenticated)
Collects the model updates from the clients
Collects and aggregates the models from the clients
Stores the model information
Pushing Updated Model to Clients
Sending raw GPS data
Sending raw GPS data
Subscribe to raw data (after authenticated)
Storing raw data for further analysis

Figure 17. UML sequence diagram (from simulator to storing model).

Figure 18 illustrates a UML sequence diagram depicting how end-user can develop and access applications tailored to their specific use cases. The process begins with the user the application front-end dashboard UI through the browser. Behind the scenes, the UI application interacts with the database to retrieve additional GPS data required for running and testing the models.

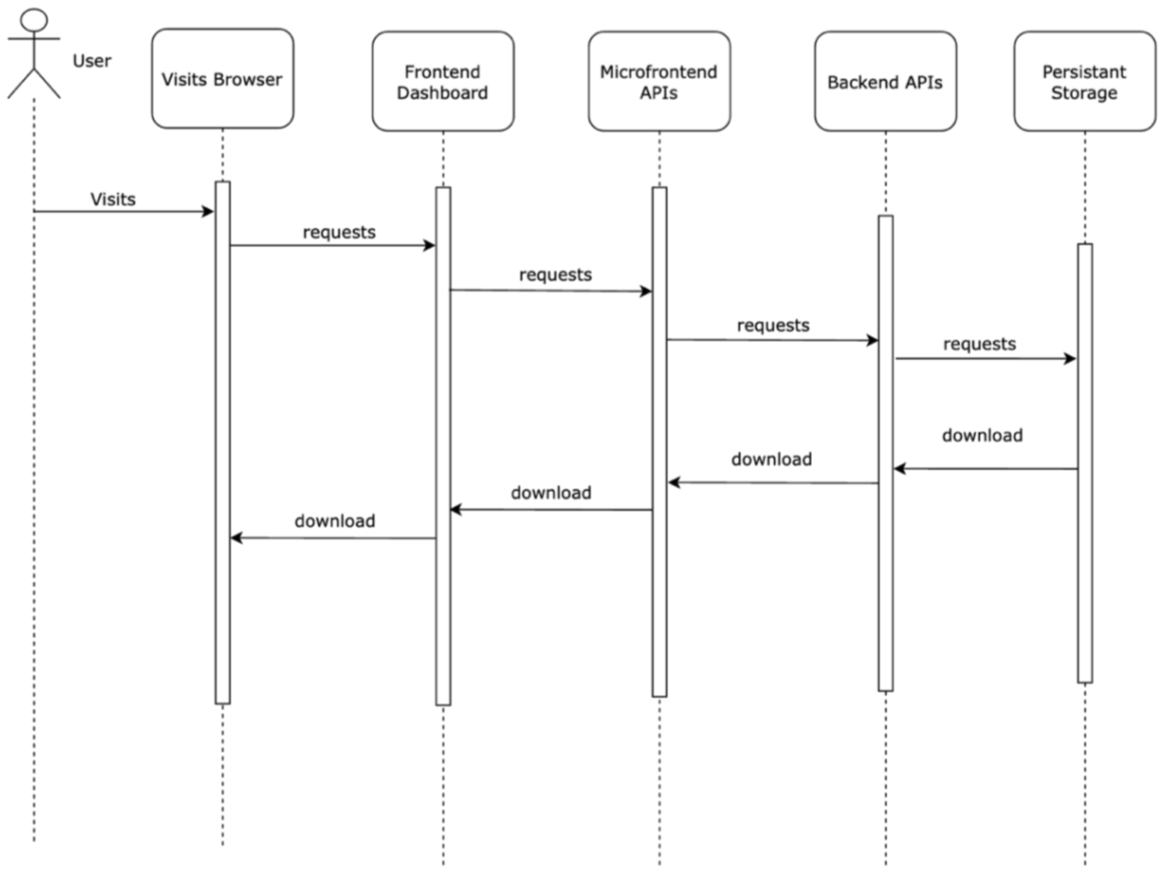


Figure 18. UML sequence diagram (User pulls the model information).

4.4. Results

This study investigated the SOTA review of Software Frameworks in aviation and other domains, each of which primarily focuses on the quality attributes such as reusability, flexibility, and maintainability. The insights gained from these domain-specific lays the ground for our proposed approach, named ‘Fed-CPS’, a hybrid architecture. Fed-CPS integrates the four architectural styles

- microkernel, pipe-and-filter, event-based, and a micro front-end-based architecture style. This style will leverage FL to address privacy concerns and enable de-centralized distributed learning.

Within this proposed architecture, the microkernel plays an important role. It is designed for its modularity and extensibility, making it well-suited for smaller-scale applications. This architecture style aligned with our use case, where clients can implement their personalized modular design. Each microkernel-based client will perform data wrangling through pipe-and-filter, making it self-contained, and the processed data will be trained locally with ML algorithms. These trained ML parameters will then stream in an event-based fashion, enhancing performance and scalability while maintaining a modular structure. The micro front-end-based style further adds strengths on reusability and adaptability by allowing end-user applications to be designed with flexible development, and scalable and resilient making it well-suited for a wide array of use-cases.

Implementing privacy settings and adherence to Zero Trust policies provides the FL architecture with resilience against potential security threats. Furthermore, containerizing all applications within this architecture enhances portability, concurrently reducing costs and minimizing maintenance requirements, promoting a more sustainable and economical computing environment. In summary, the proposed Fed-CPS approach strategically integrating microkernel, event-based systems, and micro front-end-based structures and leveraging FL not only addresses key quality attributes but also sets the stage for a modular, scalable, and privacy-conscious framework.

CHAPTER V

VALIDATION OF FEDERATED LEARNING FRAMEWORK

5.1. Introduction

Federated Learning (FL) represents a novel distributed approach to ML, redefining the conventional methods of ML application. Instead of training in a centralized location, FL enables models to undergo training on individual edge devices, achieving the privacy of sensitive data. In this procedure, the decentralized training of models on these edge devices aggregates their updates on a central server. This central server performs computations and returns the updated weights to the edge devices, this process will be iteratively performed until convergence is achieved on the edge devices.

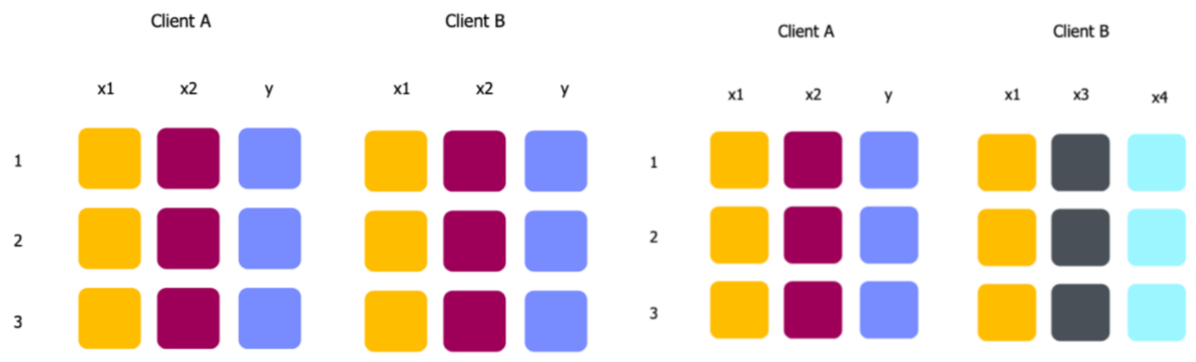
This chapter will provide an overview of FL and its implementation for predicting the geo-altitude of the aircraft. Furthermore, the study will assess the accuracy of predictions and quality attributes such as cost, performance, and energy efficiency that are yet to be explored. Simulation and monitoring tools, including Docker (simulating client), Flower (developing FL), and Grafana and Prometheus (system monitoring), are utilized for implementing the testbed environment. These tools improve the visibility and management of FL systems, providing a robust foundation for real-world applications.

5.2. Essentials of Federated Learning

As the demand for computational power rises the data storage capacity also increases. In the past, ML and DL applications were typically done centrally by training the data, with some popular use cases including fuel efficiency and trajectory prediction. As the aviation industry expands, the demand for substantial data to improve performance increases, leading to a greater need for investment in server infrastructure. FL is a promising approach to tackle the issues with fragmented data stored in silos for training diverse applications and customizing learning models for specific user groups. FL also facilitates modular approaches to demand prediction, allowing for the evaluation of these models across a broad airspace environment.

FL was initially introduced by Google in 2016 [160]. ML models run on each mobile client in this framework, resembling distributed learning. The key feature is the ability to facilitate collaborative learning by sharing only the prediction model weights among devices while retaining all the training data locally. This eliminates the need to store data in the cloud for performing ML. While this distributed processing primarily aims at speeding up the processing stage, FL also focuses on constructing a collaborative model without compromising privacy.

As illustrated in Figure 19, FL can be categorized into two types: horizontal FL and vertical FL. In a horizontal FL, there exists a slight overlap in the characteristics of data across various clients. In a specific situation where the number of feature columns was limited, research was conducted, introducing hierarchical heterogeneous horizontal Federated Learning (HHHFL) as one of the approaches and this was applied it in the classification of Electroencephalography (EEG). In this approach each client was iteratively swapping repeatedly as the targeted feature to generate additional data [161]. In vertical FL, data may or may not have partial overlap in features (e.g., reference ID) but differ in data samples. For example, collaboration among different entities allows them to work together and learn, providing a better understanding of the model.



(a) Horizontal (or Homogeneous) FL with same features across clients

(b) Vertical (or Heterogeneous) FL with diverse features across clients (features may or may not be shared between clients)

Figure 19. Categories of FL.

One of the primary issues in FL is with inconsistent data distribution. It is assumed that the data is independently and identically distributed (IID) across all participating devices in an ideal scenario. However, real-world situations the data is always in Non-Independently and Identically Distributed (Non-IID) format. This non-IID characteristic is attributed to various factors, including configuration and shifts in user preferences. In practical settings, data distribution across devices frequently impacts the degradation of performance and generalization capabilities of machine learning models, as observed in studies documented in sources such as [162], [163].

5.3. Dataset Description

In this section, information about the origin of the collected dataset will be provided. Additionally, the dataset was separated into distinct batches, with preprocessing steps focusing only on the geo-altitude feature.

5.3.1. Dataset Collection and Preprocessing

The dataset is collected from the OpenSky Network [68] in a series of parquet files organized by day and hour from February 18 to 27, 2022, which captured a diverse range of aviation events during this timeframe. To ensure the quality dataset, datasets were preprocessed using pySpark. The primary focus of this chapter is to predict geo-altitude. For each dataset, data wrangling (including removing duplicate entries and addressing missing values) will be performed for every flight record. Subsequently, only the geo-altitude information is extracted, creating a new dataset with a time window of size 4. This focus is directed on Horizontal FL, utilizing the same feature column with varying data samples. Different batches of datasets are generated; their categorization is detailed in Table 5 provides a comprehensive overview of how the data is segmented, providing valuable insights into the distribution strategy across different clients in the FL framework.

Table 5. Batch data details

Batch Name	Train-Test Sample Size	Comments / Categories
Dataset-1	C1: ~4900 / ~11000 C2: ~7300/ ~4600	Considered each client as each flight with multiple trips
Dataset-2	C1: ~62000/~29000 C2: ~57000 / ~33000	Combined multiple flights and grouped into two clients
Dataset-3	C1-C4: ~20000/~7000-10000	Combined multiple flights and grouped into four clients
Dataset-4	C1-C8: ~2000-7000 / 1000-7000	Considered each flight as each flight with multiple trips

5.4. Implementation of Federated Learning Framework

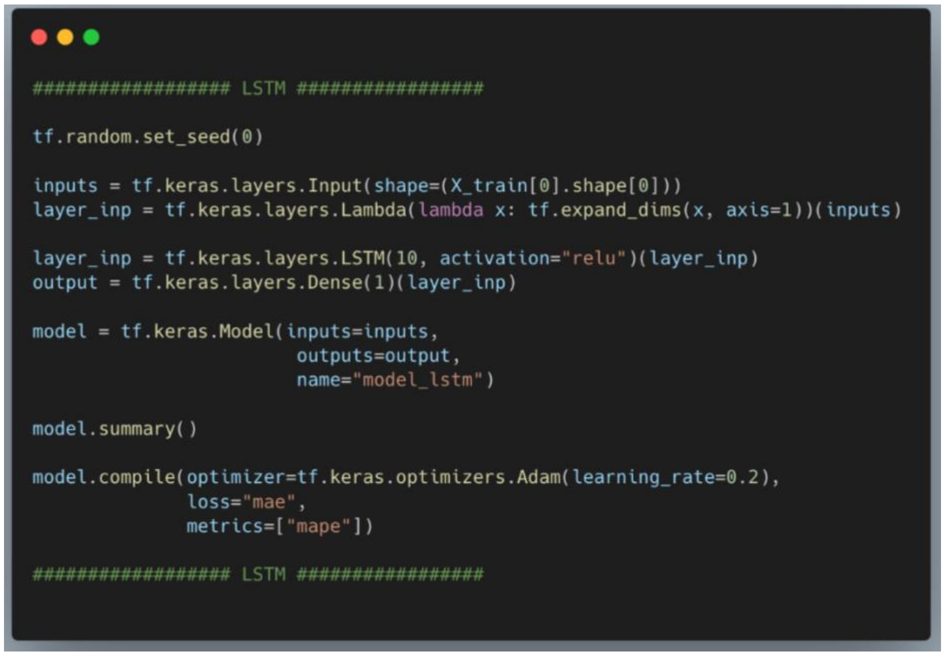
Many FL libraries are available on the Internet, but they are still in the developmental stage and not yet suitable for production use. Flower [164] stands out as a user-friendly FL framework in our specific scenario. Its intuitive design enables easy implementation and deployment on any infrastructure, particularly useful for large-scale testing.

On the client side, the architecture is responsible for training the dataset associated with that client, utilizing different ML or DL models. The server consists of the - FL loop, RPC server, and a customizable strategy chosen by the user. Various strategies are available for aggregating the model weights from each client. Clients connect to the RPC server, which monitors the connection requests of clients. The FL loop is the central component of the federated learning process, orchestrating continuous learning.

Two ML algorithms: LSTM and k-Means were considered both running on the clients trained with different batches of dataset, while the central server will perform Federated Average Aggregation on the global model. The source code for this project is available on GitHub [165].

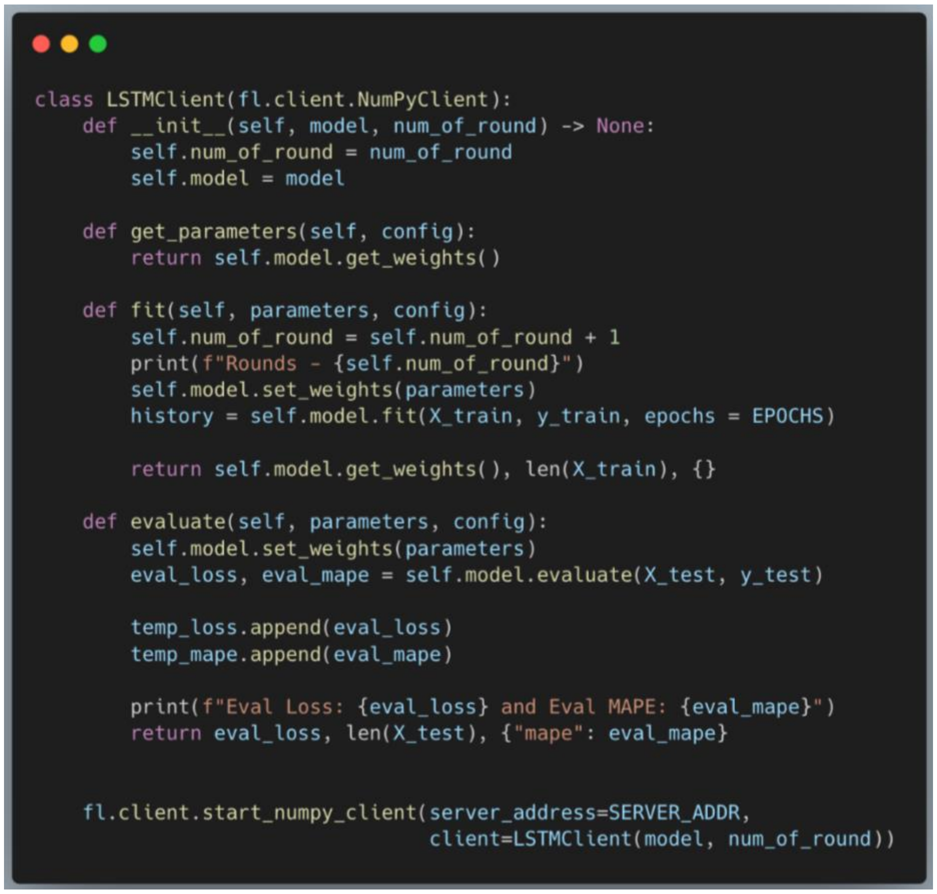
5.4.1. Long Short-Term Memory (LSTM)

LSTM is an enhancement of traditional recurrent neural network (RNN), proposed by Hochreiter and Schmidhuber [166] uses temporal correlations between past and the current data points. As a result, LSTM addresses the common issues of vanishing gradient in RNNs. This algorithm is generally applied to time-series data, in which each sequence of data points reflects the interdependence between current and historical data. Our focus is on predicting geo-altitude, and Figure 20 illustrates the code snippet of this algorithm on the client side. Utilizing the LSTM model from the Keras library [167] configured with 10 layers and the 'ReLu' activation function, the output is the predicted geo-altitude. To optimize the model, Adam optimization algorithm with a low learning rate of 0.2 is configured. Each client uses the abstract base class implemented using NumPy, and the predefined functions for fitting and evaluation facilitate the exchange of model weights. This process, illustrated in Figure 20 is iteratively performed to make the model better by retraining it.



```
################## LSTM ##################

tf.random.set_seed(0)

inputs = tf.keras.layers.Input(shape=(X_train[0].shape[0]))
layer_inp = tf.keras.layers.Lambda(lambda x: tf.expand_dims(x, axis=1))(inputs)

layer_inp = tf.keras.layers.LSTM(10, activation="relu")(layer_inp)
output = tf.keras.layers.Dense(1)(layer_inp)

model = tf.keras.Model(inputs=inputs,
                        outputs=output,
                        name="model_lstm")

model.summary()

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.2),
              loss="mae",
              metrics=["mape"])

################## LSTM ##################
```

Figure 20. Implementation of LSTM model in the Federated Client.

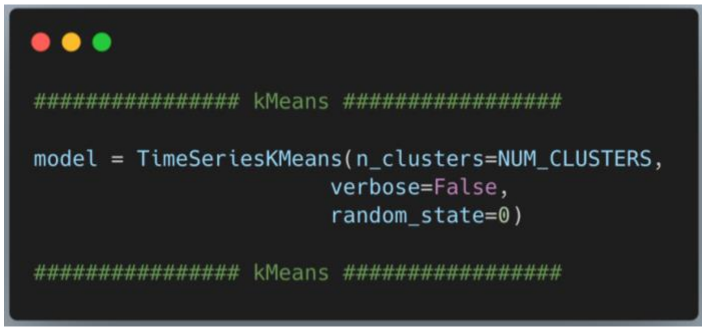


```
class LSTMClient(fl.client.NumPyClient):
    def __init__(self, model, num_of_round) -> None:
        self.num_of_round = num_of_round
        self.model = model

    def get_parameters(self, config):
        return self.model.get_weights()

    def fit(self, parameters, config):
        self.num_of_round = self.num_of_round + 1
        print(f"Rounds - {self.num_of_round}")
        self.model.set_weights(parameters)
        history = self.model.fit(X_train, y_train, epochs = EPOCHS)

        return self.model.get_weights(), len(X_train), {}

    def evaluate(self, parameters, config):
        self.model.set_weights(parameters)
        eval_loss, eval_mape = self.model.evaluate(X_test, y_test)

        temp_loss.append(eval_loss)
        temp_mape.append(eval_mape)

        print(f"Eval Loss: {eval_loss} and Eval MAPE: {eval_mape}")
        return eval_loss, len(X_test), {"mape": eval_mape}


    fl.client.start_numpy_client(server_address=SERVER_ADDR,
                                 client=LSTMClient(model, num_of_round))
```

Figure 21. Implementation of LSTM-based Abstract Class in the Federated Client.

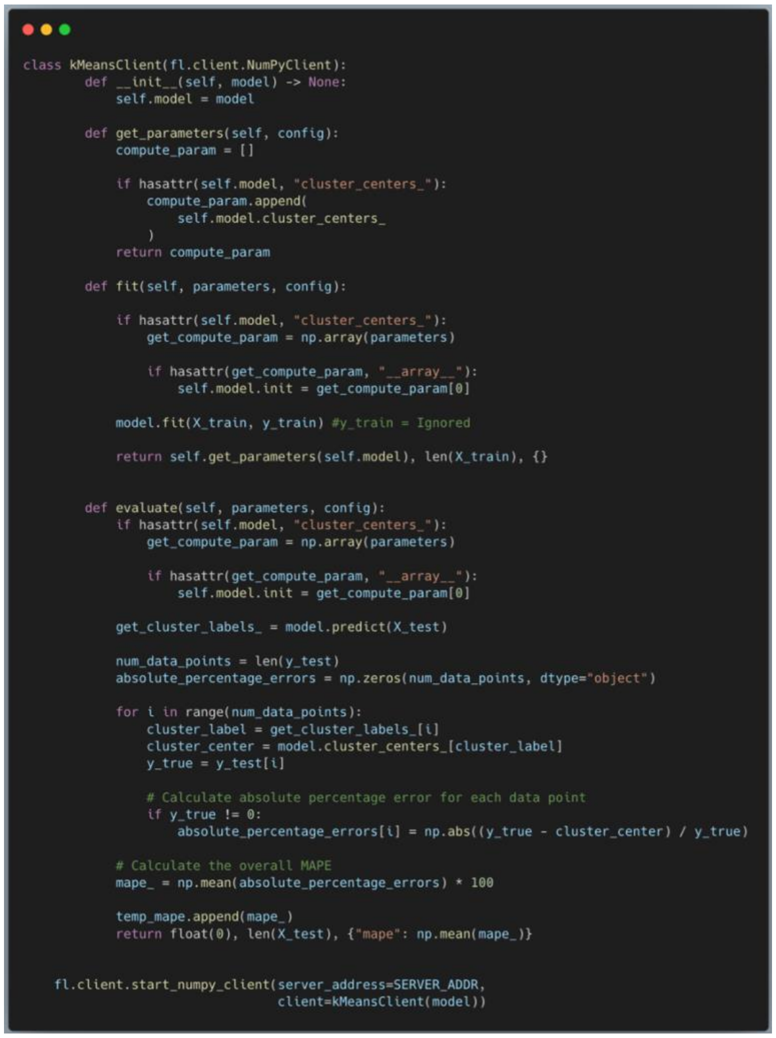
5.4.2. k-Means Clustering

It is an unsupervised clustering-based algorithm uses grouping technique to form clusters of similar points. The objective of this algorithm is to find the ‘k’ cluster centroids by comparing the features of each data point, the algorithm assigns them to a cluster to reduce the overall distance between the data points and their respective cluster centroid. The algorithm aims to ensure that each data point is correctly assigned to its corresponding group [168]. Generally, the Euclidean distance determines the features' similarity and forms clusters with data points. Figure 22 illustrates the code snippet of this algorithm and assigned with number of clusters ranging from 1 to 3 implemented using tslearn library [169]. Like LSTM, each client on k-Means also uses the abstract base class implemented via NumPy, and the predefined functions for fitting and evaluation facilitate the exchange of cluster centroids. This process, illustrated in Figure 23 is iteratively performed.



```
############### kMeans ################

model = TimeSeriesKMeans(n_clusters=NUM_CLUSTERS,
                         verbose=False,
                         random_state=0)

############### kMeans ################
```

Figure 22. Implementation of k-Means model in the Federated Client.



```
class kMeansClient(fl.client.NumPyClient):
        def __init__(self, model) -> None:
            self.model = model

        def get_parameters(self, config):
            compute_param = []

            if hasattr(self.model, "cluster_centers_"):
                compute_param.append(
                    self.model.cluster_centers_
                )
            return compute_param

        def fit(self, parameters, config):

            if hasattr(self.model, "cluster_centers_"):
                get_compute_param = np.array(parameters)

                if hasattr(get_compute_param, "__array__"):
                    self.model.init = get_compute_param[0]

            model.fit(X_train, y_train) #y_train = Ignored

            return self.get_parameters(self.model), len(X_train), {}


        def evaluate(self, parameters, config):
            if hasattr(self.model, "cluster_centers_"):
                get_compute_param = np.array(parameters)

                if hasattr(get_compute_param, "__array__"):
                    self.model.init = get_compute_param[0]

            get_cluster_labels_ = model.predict(X_test)

            num_data_points = len(y_test)
            absolute_percentage_errors = np.zeros(num_data_points, dtype="object")

            for i in range(num_data_points):
                cluster_label = get_cluster_labels_[i]
                cluster_center = model.cluster_centers_[cluster_label]
                y_true = y_test[i]

                # Calculate absolute percentage error for each data point
                if y_true != 0:
                    absolute_percentage_errors[i] = np.abs((y_true - cluster_center) / y_true)

            # Calculate the overall MAPE
            mape_ = np.mean(absolute_percentage_errors) * 100

            temp_mape.append(mape_)
            return float(0), len(X_test), {"mape": np.mean(mape_)}


    fl.client.start_numpy_client(server_address=SERVER_ADDR,
                                 client=kMeansClient(model))
```

Figure 23. Implementation of k-Means based Abstract Class in the Federated Client.

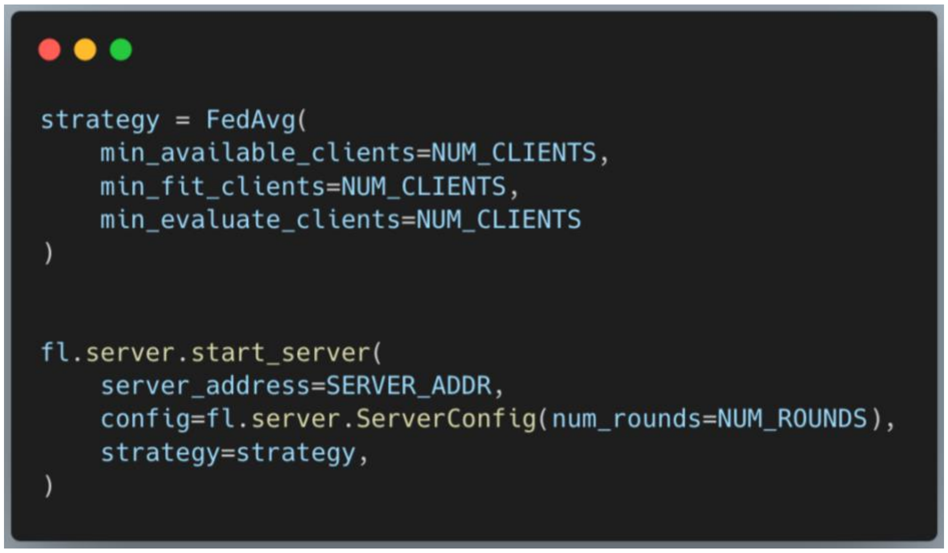
5.4.3. Federated Averaging (FedAvg)

Gradient descent is an optimization algorithm used in ML to minimize the cost function associated with each data point. Models learn over time by iteratively adjusting their parameters. Stochastic gradient descent (SGD) is another optimization algorithm that refines the objective function by computing the gradient for each data point. However, for large datasets, the computation burden of calculating the gradient across the entire dataset becomes unsuitable. The Federated Stochastic Gradient Descent (FedSGD) algorithm works similarly but addresses the dataset size by selecting only a fraction (C) of clients from the entire set (K). Each participating client computes and shares the gradient with the centralized server. The Federated Averaging (FedAvg) algorithm involves sending weights to the server, which then aggregates and averages them. Each client then performs iterations of updates on these updated weights. The pseudocode for the FedAvg algorithm is shown below [160].

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

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

ClientUpdate$(k, w)$:   // Run on client $k$
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server
```

This strategy is executed with multiple clients ranging from 2 to 8 in different batches (
Table 5). The code snippet for this algorithm is depicted in Figure 24. FedAvg involves three key parameters: min_available_clients, min_fit_clients, and min_evaluate_clients. These parameters set the minimum number of clients required to start the training and evaluation process. Table 6 displays the overall parameter settings applied across various dataset batches.



```
strategy = FedAvg(
    min_available_clients=NUM_CLIENTS,
    min_fit_clients=NUM_CLIENTS,
    min_evaluate_clients=NUM_CLIENTS
)


fl.server.start_server(
    server_address=SERVER_ADDR,
    config=fl.server.ServerConfig(num_rounds=NUM_ROUNDS),
    strategy=strategy,
)
```

Figure 24. Implementation of FedAvg in Federated Server.

Table 6. Parameter Settings for Federated Learning

Algorithms / Strategy	**Hyperparameter Settings**	**Comments**
LSTM	Epochs: 100, 200	Client-Side
k-Means	Clusters: 1, 2, 3	
FedAvg	Rounds: 1, 2, 3	Server-Side Aggregator

This FL Framework is containerized using Docker, as shown in Figure 25 offering several advantages beyond deployment and scalability. This containerization ensures a standardized and isolated environment, simplifying the management of dependencies and configurations. With Docker, the framework becomes highly portable, allowing seamless execution across diverse computing environments.

Grafana [170] and Prometheus [171] are used for monitoring, as a powerful combination that provides real-time insights into the performance of containers. Grafana's intuitive dashboard interface makes visualizing and interpreting metrics easy, enhancing the ability to detect and promptly address potential bottlenecks or issues. The integration of Cadvisor [172], a popular plugin from Google, further refines our monitoring strategy by capturing essential Docker health metrics at a granularity of 5 seconds. Prometheus plays a pivotal role in extracting and storing these metrics, exposing them through the PromQL API for easy retrieval and analysis. This streamlines the monitoring process and enables users to query specific performance parameters dynamically. Grafana, in turn, leverages the data provided by Prometheus to create intuitive dashboard. This visualization layer enhances our ability to gain insights into the dynamic behavior of the FL Framework.

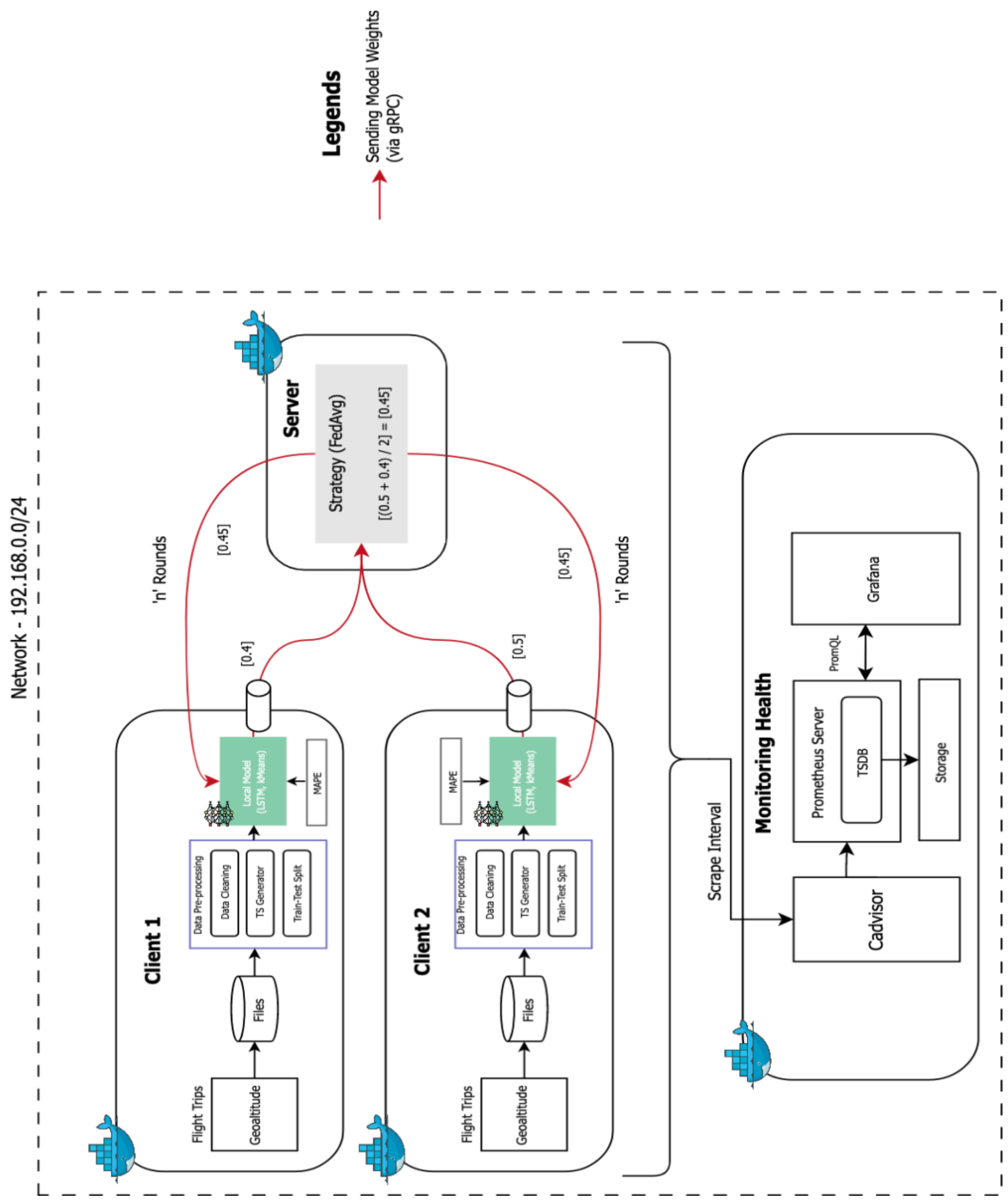


Figure 25. Architecture of Federated framework.

5.5. Results

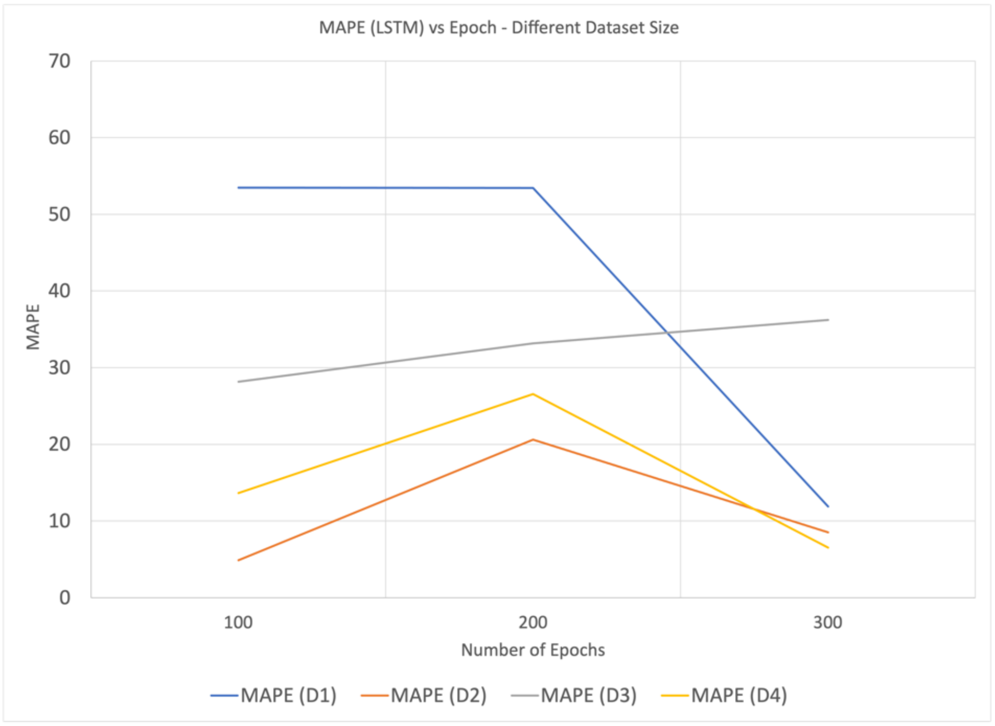
Mean Absolute Percentage Error (MAPE) is used to evaluate prediction accuracy, along with additional metrics such as CPU, memory, and network usage. This evaluation aims to understand performance in terms of cost (Performance-per-Dollar) and energy efficiency (Energy- Efficiency-per-Watt) to assess how well the FL framework predicts the geo-altitude under different settings with respect to Table 5 and Table 6.

5.5.1. Mean Absolute Percentage Error (MAPE)

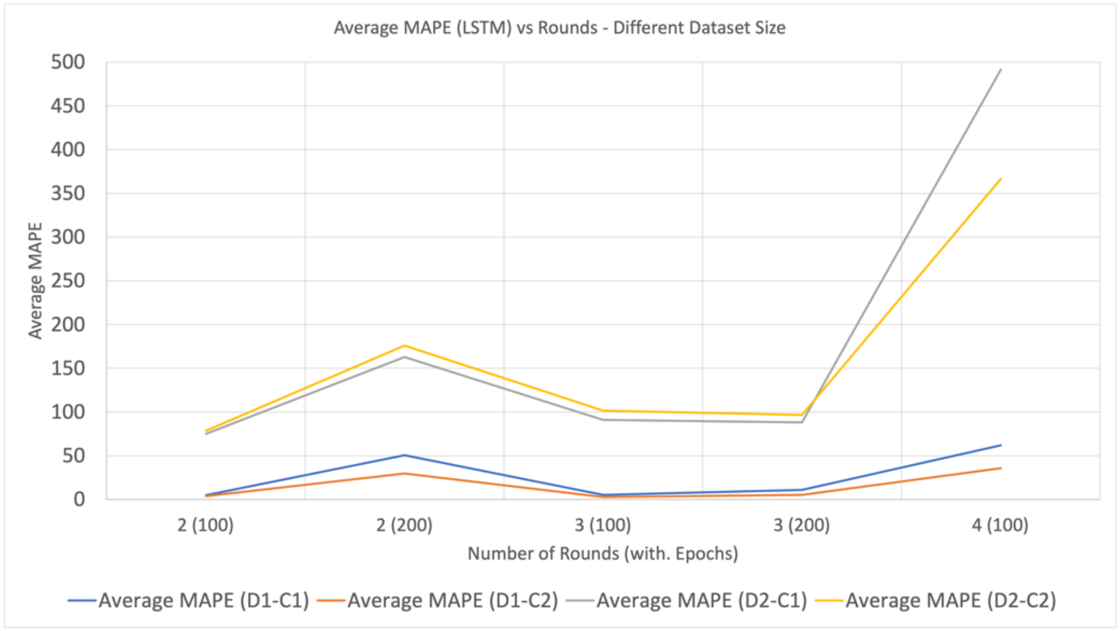
It is widely used metrics for evaluating the accuracy of the prediction or forecasting model. It measures the percentage difference between predicted and observed values, as shown in Equation 6.

$$\text{MAPE} = \frac{100\%}{N}\sum_{i=0}^{N-1}\frac{y_i-\hat{y_i}}{y_i} \quad \ldots \text{ Equation 6}$$

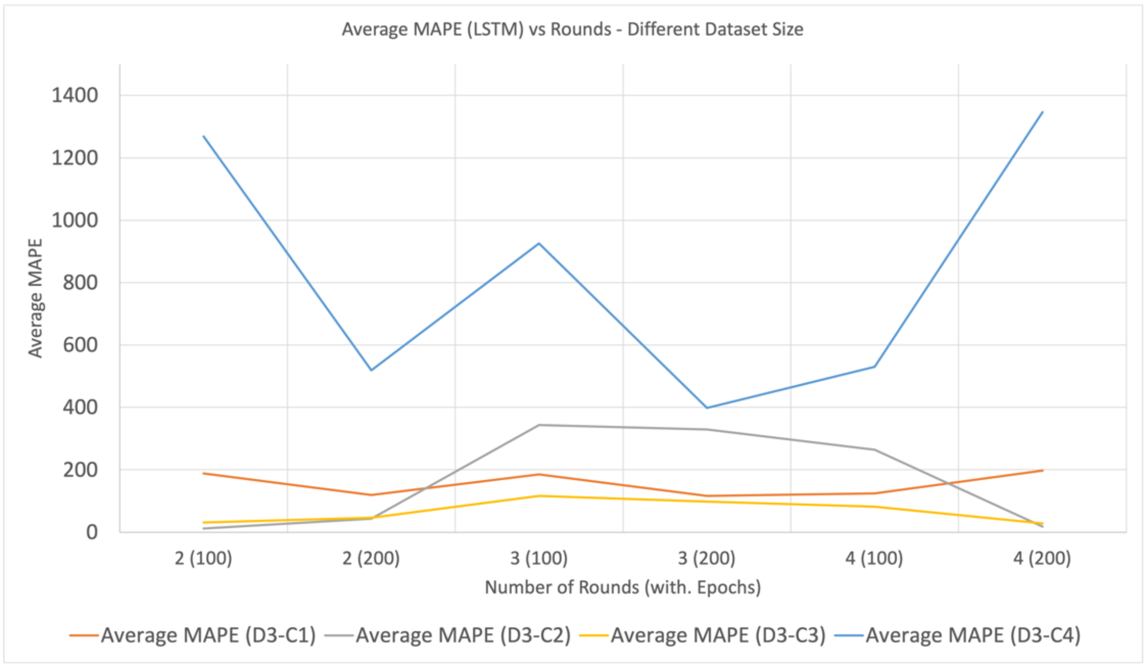
Figure 26 (a) compares MAPE against the number of epochs for various batches of datasets without utilizing FL. Regardless of the dataset, the MAPE score remains consistently below 60. Additionally, the convergence of the model is achieved within 200 epochs. Contrastingly, when the same performed in FL (Figure 26 (b), (c), (d)), each of the clients participating in the FL undergoes multiple epochs and multiple rounds of FedAvg. The overall average MAPE score is significantly reduced below 10% showing good prediction accuracy. Moreover, convergence of this FL model after multiple rounds, showcases the effectiveness of collaborative learning across clients in refining the predictive model.



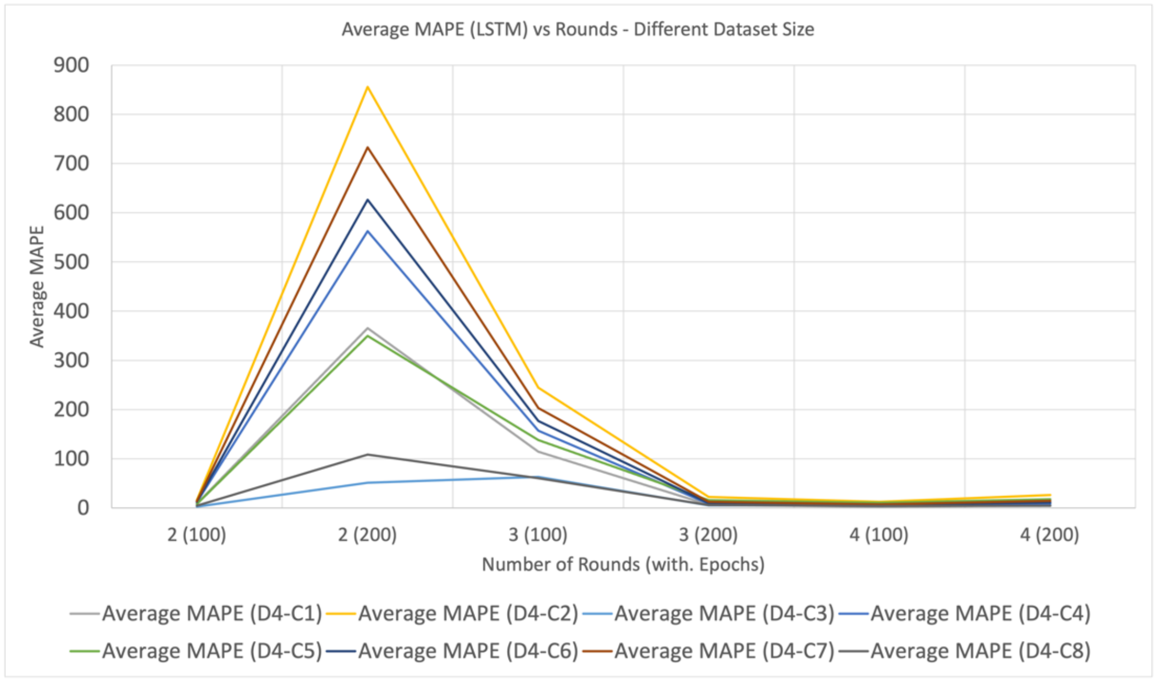
(a) Comparison of MAPE vs Epochs (without FL)



(b) Comparison of Average MAPE vs Rounds (with Epochs) for Two Clients on Dataset-1, and 2 (with FL)



(c) Comparison of Average MAPE vs Rounds (with Epochs) for Four Clients on Dataset-3 (with FL)

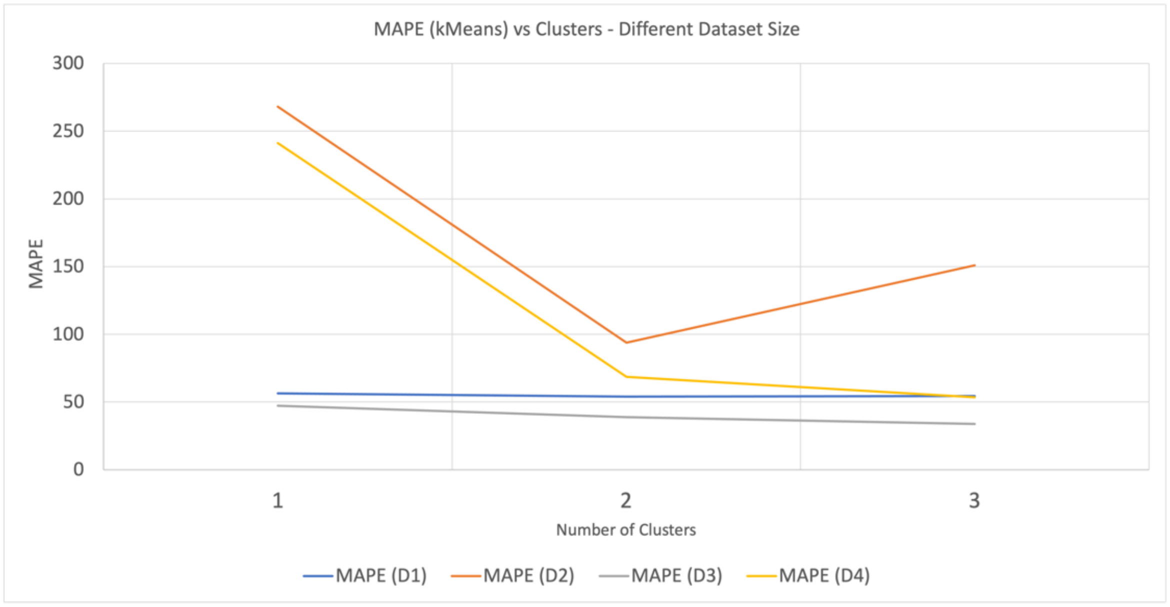


(d) Comparison of Average MAPE vs Rounds (with Epochs) for Eight Clients on Dataset-4 (with FL)

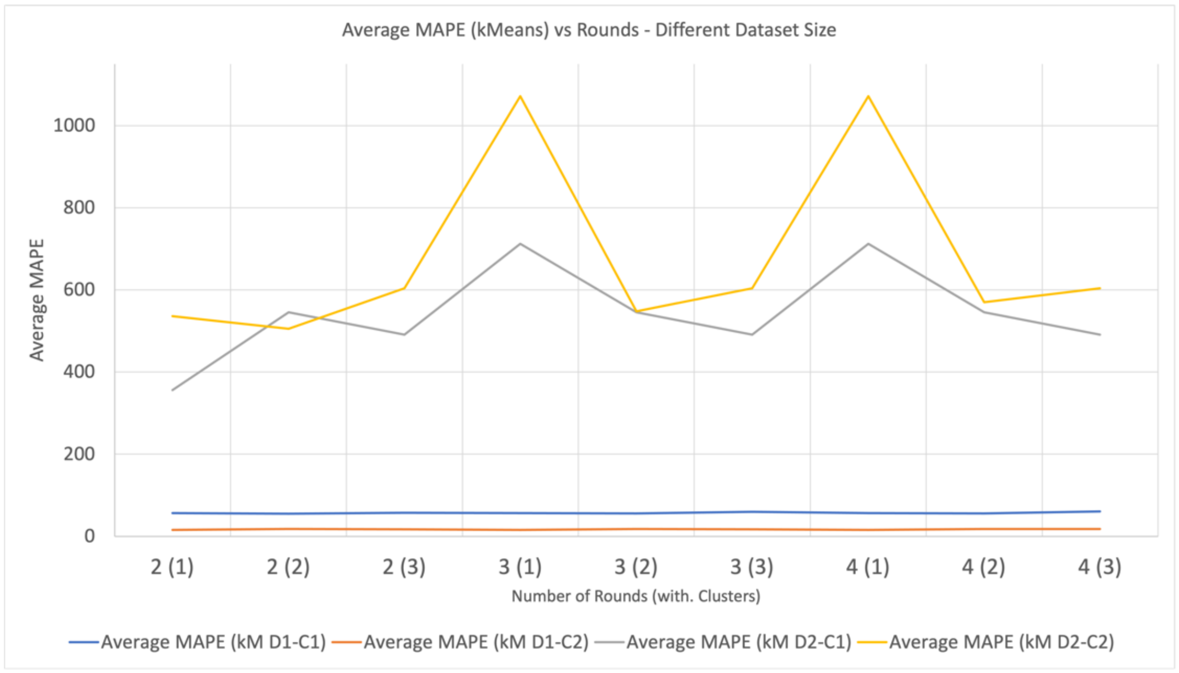
Figure 26. Comparison of MAPE Score with and without FL for LSTM.

Similarly, the same approach was applied in the k-Means algorithm across various batch datasets, without utilizing FL, to compare MAPE against the number of clusters (refer to Figure 27 (a)). The MAPE score remains consistent at 50 for the datasets 1 and 2. However, for larger datasets, the MAPE score increases, showing a higher level of error. Additionally, the MAPE score tends to decrease as the cluster increases.

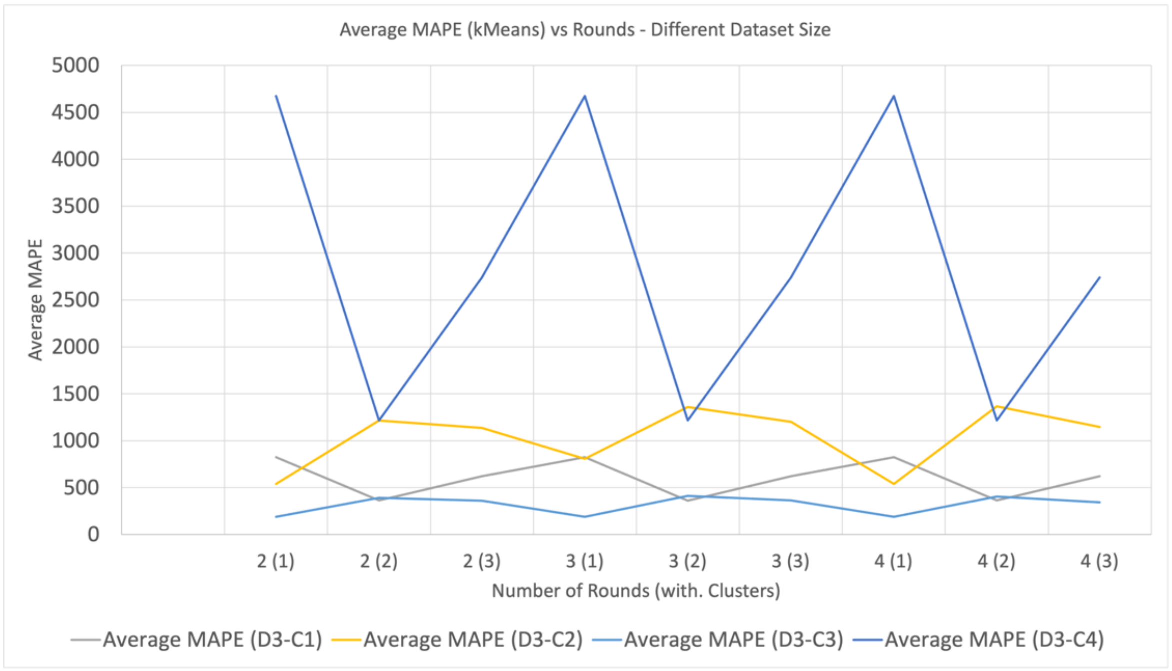
In contrast, the same were assessed with the FL (refer to the Figure 27 (b), (c), (d)), the overall average MAPE score for each client results in a reasonable reduction, mainly when there are additional rounds of FedAvg compared to the LSTM. A minimum of four rounds of computation is required to yield improved average MAPE results.



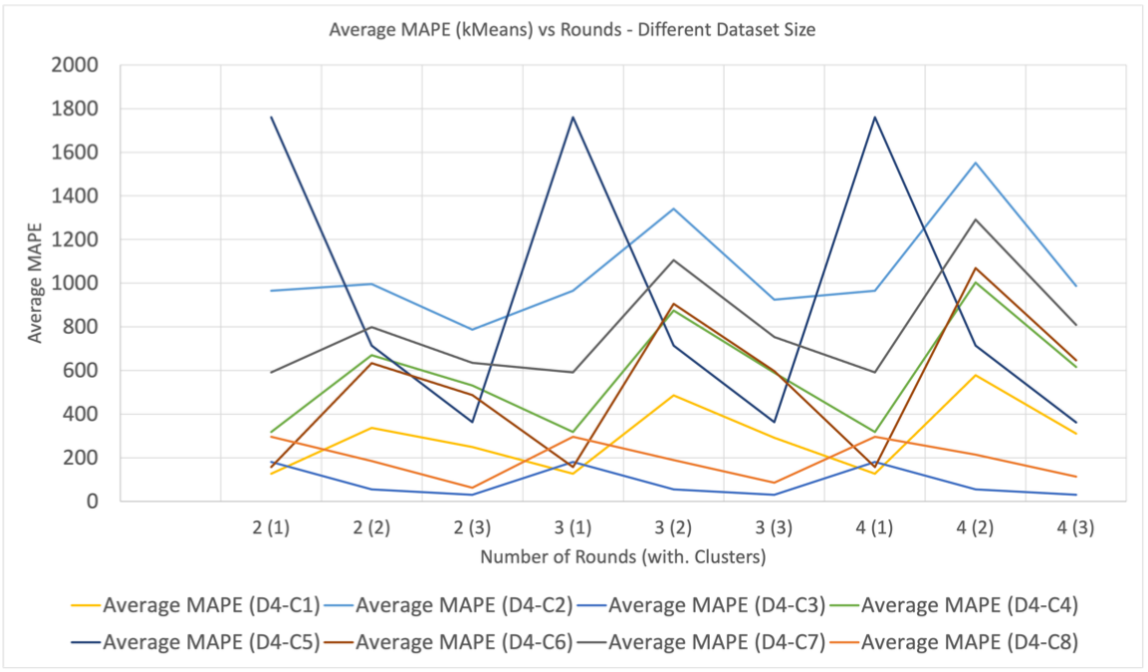
(a) Comparison of MAPE vs Clusters (without FL)



(b) Comparison of Average MAPE vs Rounds (with Clusters) for Two Clients on Dataset-1, and 2 (with FL)



(c) Comparison of Average MAPE vs Rounds (with Clusters) for Four Clients on Dataset-3 (with FL)



(d) Comparison of Average MAPE vs Rounds (with Clusters) for Eight Clients on Dataset-4 (with FL)

Figure 27. Comparison of MAPE Score with and without FL for k-Means.

5.5.2. Performance-per-dollar (PD) and Energy Efficiency-per-Watt (EEW)

Two key metrics were used to evaluate the FL's cost-effectiveness and energy efficiency, based on the [173]: Performance-per-dollar and Energy Efficiency-per-Watt. These algorithms ran on Intel Xeon Processor with eight cores and 16GB of memory. It is estimated that this processor costs $8000 and is designed with a Thermal Design Power of 165 watts. The formulas used to calculate these metrics are listed below:

PPD = CPU performance (in percentage) / Cost of CPU …Equation 7

EEW = CPU performance (in percentage) / Thermal Density Power … Equation 8

Developed a Grafana visualization dashboard to analyze and display peak CPU usage as shown in Figure 28 and Figure 29. The CPU usage reaches 200%, implying each core is operating at maximum capacity (100%). Specifically, during the computation of LSTM, the CPU usage remains at 200% with relatively low memory usage. In the case of k-Means, the CPU usage is 100% but utilizes more than 10GB of memory. When translating these values into metrics, it becomes apparent that LSTM yields a PD ratio 0.2 times better than k-Means (without FL). In the FL, LSTM utilizes only 150% of the CPU, whereas k-Means operates at 100%. Interestingly, during LSTM computations, the server demonstrates a CPU usage of only 5%, contrasting with the 10% observed during k-Means computations. This results in LSTM exhibiting a 1.5x better PD ratio on the client side and a 0.5x improvement on the server side compared to k-Means.

Furthermore, to estimate Energy Efficiency, thermal density power is considered, which refers to the power consumed when the system operates at its maximum capacity. For the CPU, the power consumption is measured at 165 W. Based on the observations, LSTM consumes more than twice the power compared to k-Means (without FL). In Federated Learning, LSTM consumes 1.5 times more power per client ratio than k-Means. However, when considering power consumption on the server side, LSTM yields less than 0.5 times the power consumption ratio compared to k-Means.

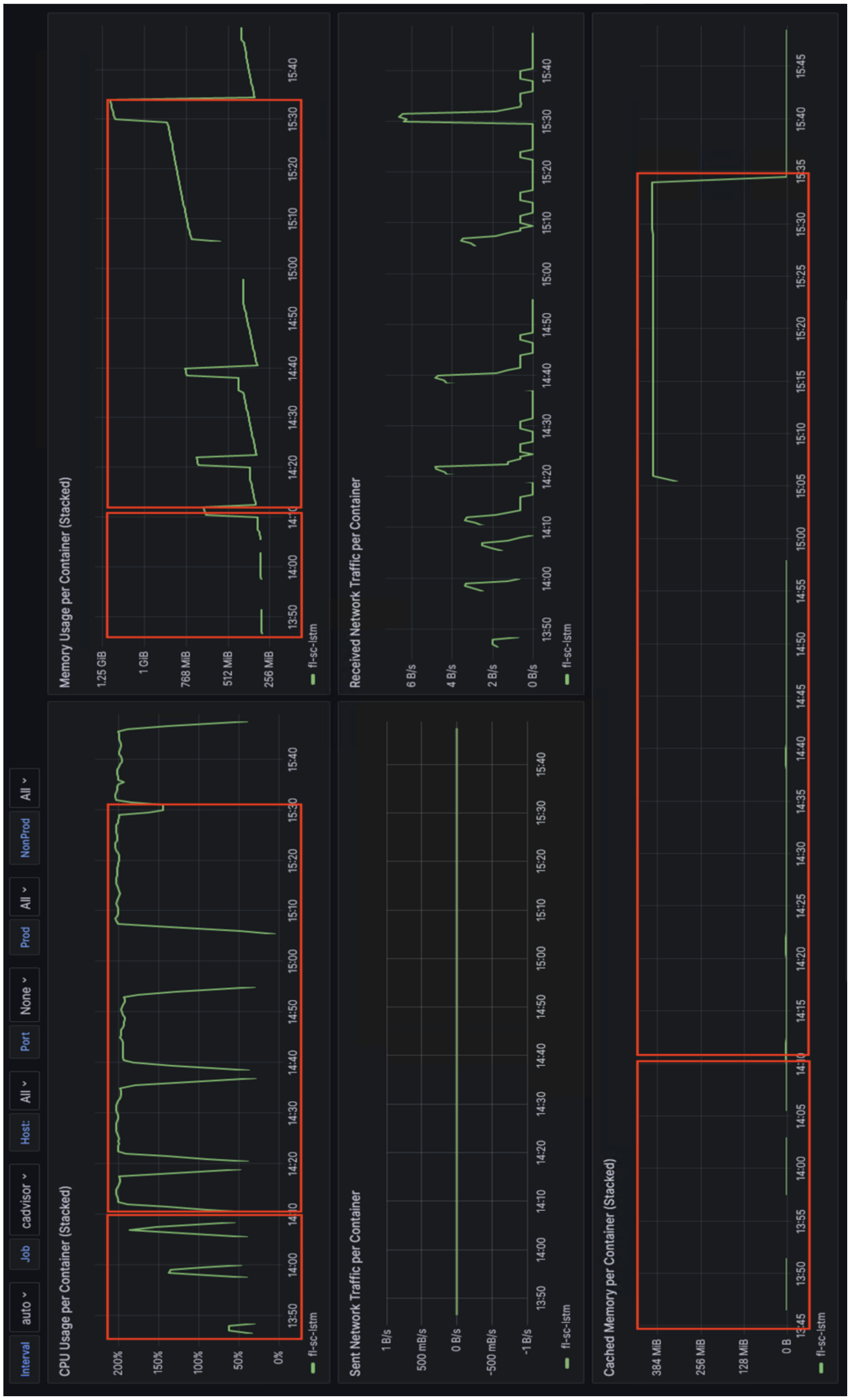


Figure 28. Dashboard metrics for non-Federated Setting (LSTM).

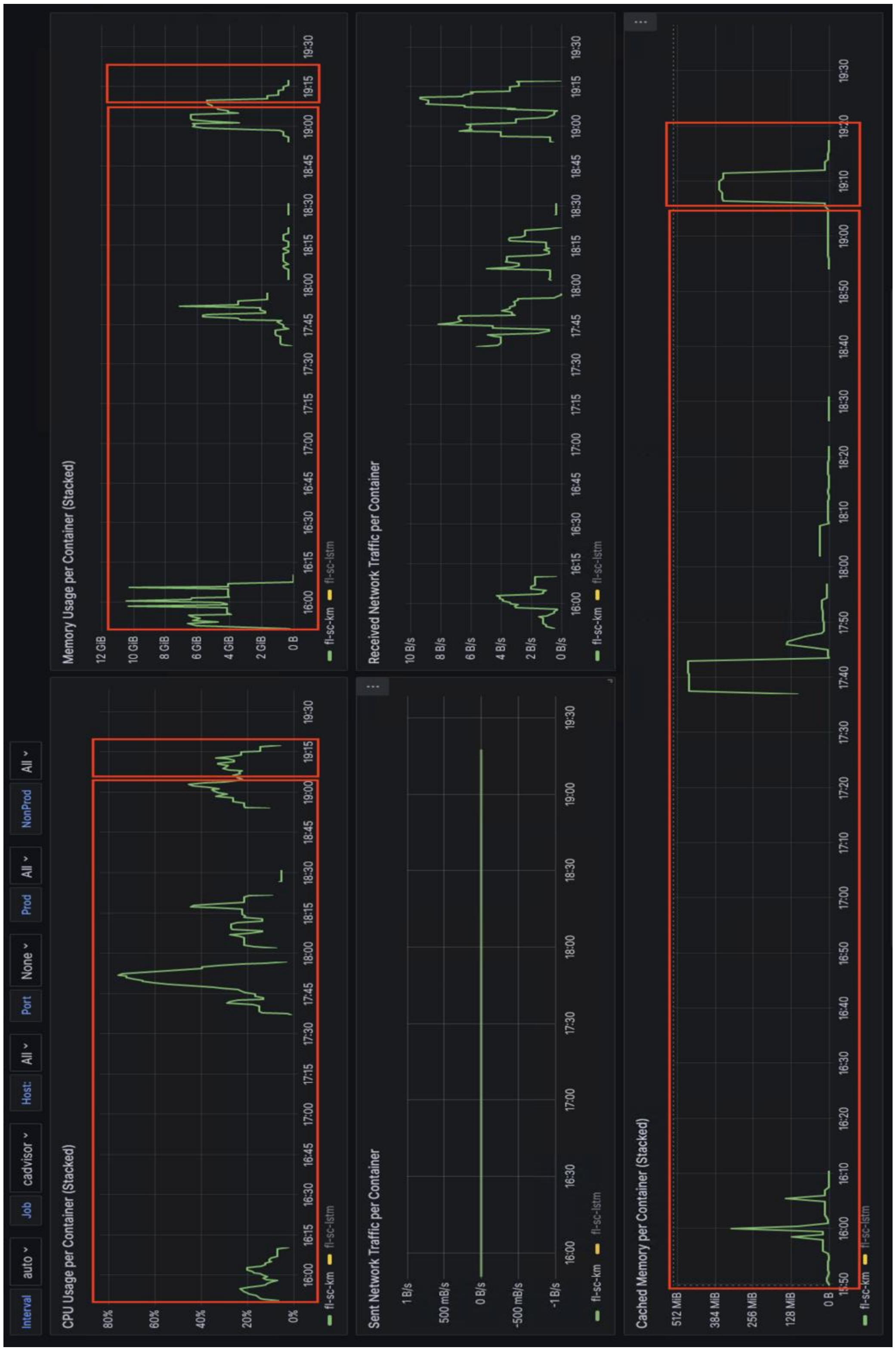

Interval auto
Job cadvisor
Host: All
Port None
Prod All
NonProd All
CPU Usage per Container (Stacked)
Memory Usage per Container (Stacked)
Sent Network Traffic per Container
Received Network Traffic per Container
Cached Memory per Container (Stacked)
fl-sc-km
fl-sc-lstm

Figure 29. Dashboard metrics for non-Federated Setting (k-Means).

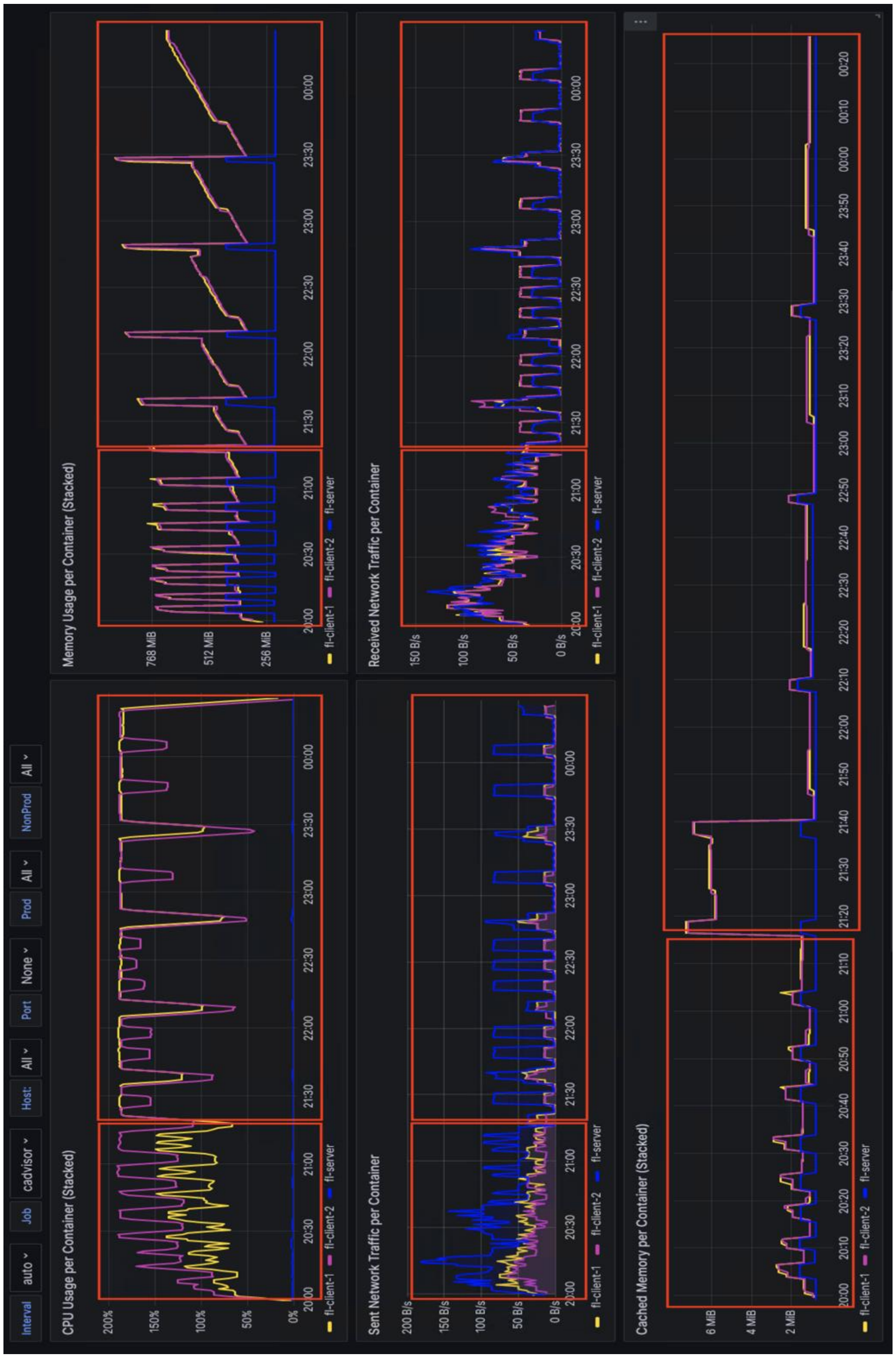


Figure 30. Dashboard metrics for Federated Setting (LSTM).

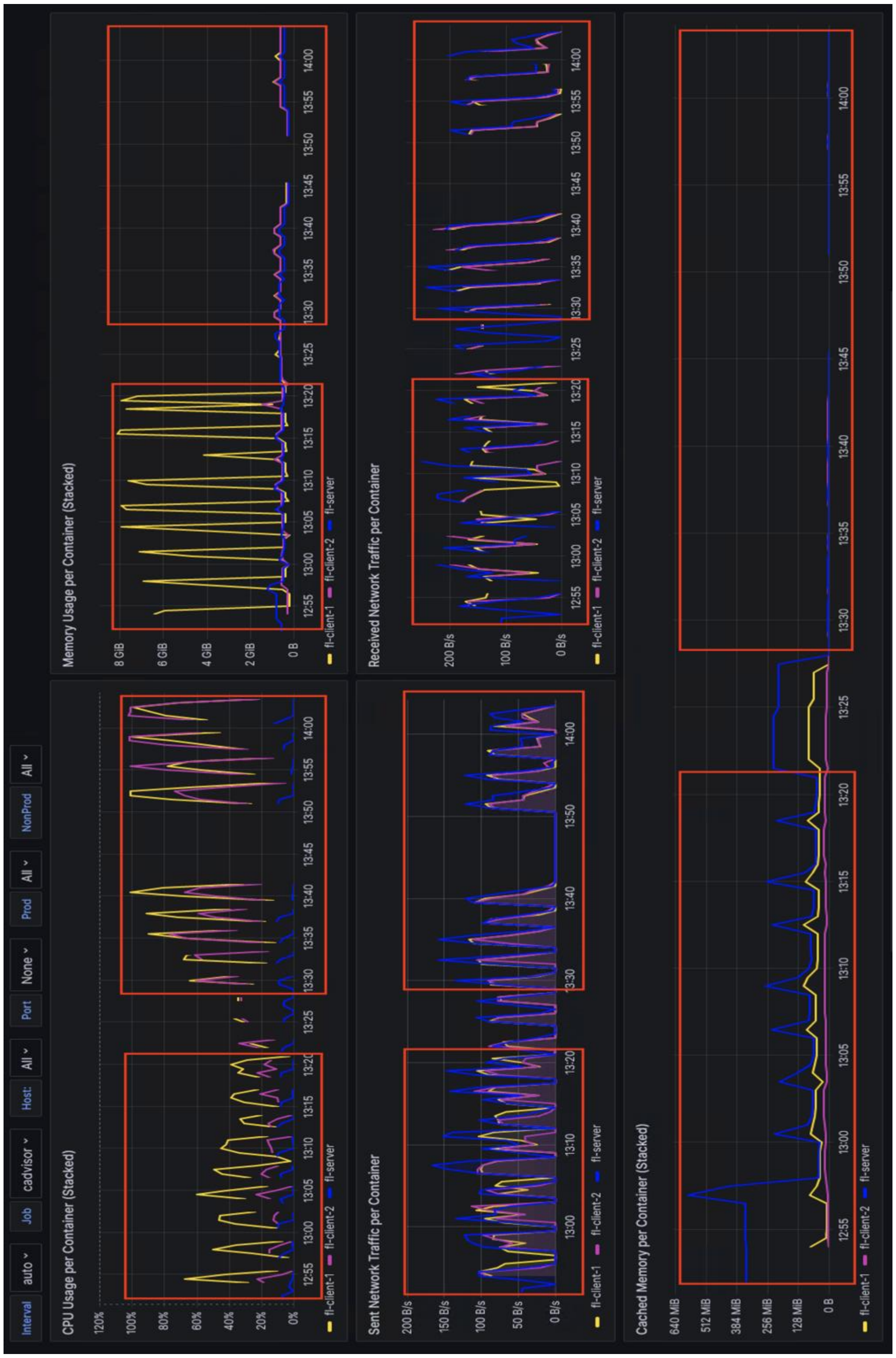


Figure 31. Dashboard metrics for Federated Setting (k-Means).

CHAPTER VI

CONCLUSION AND FUTURE WORK

As part of the NextGen initiative, several efforts are being taken to modernize the National Airspace System (NAS) to increase the safety and resiliency of current airspace operations and reach urban and rural areas. The GPS data integrity and accuracy are important in determining the location of the aircraft. Any compromise in the integrity of GPS data can lead to disruption in flight safety and navigational accuracy. To mitigate potential GPS dropout-related incidents, there is a need for robust data-driven models. Below are the main contributions of this thesis study:

1. **Selection of Robust Imputation Method under uncertain GPS Integrity Scenarios**: The first study explored applying ML algorithms for imputing missing data points in ADS-B / GPS data obtained from the OpenSky Network using five different ML models: Bayesian Ridge, Random Forest, AdaBoost, Extra Tree Regressor, and k-Nearest Neighbor. These models were experimented with various missing ratios, ranging from 10% to 30%, for the parameters—latitude, longitude, and geo-altitude. The results were assessed using MAE and RMSE, showing k-NN as a robust imputation method demonstrating effectiveness even when dealing with a high rate of missing data (30%). This significantly contributes to the identification of an optimal imputation approach for datasets with substantial missing values.
2. **Conceptualized a Hybrid Framework to benchmark performance with and without FL for the GPS network:** The ADS-B/GPS exchange information between aircraft transparently, which opens to various attack vectors such as spoofing, jamming, MITM. Thus, this study explored the possibility of privacy-preserving solutions through a decentralized learning approach using FL, proposing a conceptualized hybrid framework named 'Fed-CPS,' which combines the advantages of a microkernel, event-driven, pipe and filter, and micro-frontend

design paradigms. Specifically designed for CPS in aviation networks, ‘Fed-CPS’ incorporates both SITL and HITL testing approaches.

3. **Validated the Hybrid Framework to quantify quality attributes**: Cost, Performance, and Efficiency - Additionally, the study identifies three quality attributes—cost, performance, and efficiency that have not been explored in existing research papers for benchmarking in an FL setting for GPS data. Utilizing the Flower, and Docker software frameworks, the FL was implemented to predict the geo-altitude. The results demonstrate that LSTM outperforms k-Means in the FL regarding MAPE and PD but consumes more EEW.

There are certain limitations that exist in this study one of them is limited number of clients (i.e., 8 clients) were considered to evaluate the quality attributes. Furthermore, there is a need to consider features other than geo-altitude. Additionally, when new clients attempt to join an ongoing learning process, they must wait until the current learning process is completed. Currently, there are no tools available to support concurrent operations. Moreover, if a client leaves the ongoing operation, the entire process comes to a halt, and as of now, there is no existing solution to address this interruption issue. Nevertheless, there are several unanswered questions that still need further research. This involves implementing and testing it extensively using various combinations of data sources through Vertical FL and selecting customized ML models [174] to improve the performance of the participating clients. However, this also raises questions about the fairness and bias of these models. This also brings up the need for an appropriate selection of participating clients to address the stragglers [175], [176], as each participant runs at different computing speeds. Finally, this FL setting needs to be incorporated in the real-time continuous streaming data.

PUBLICATIONS

- B. S. Chandar, P. Ranganathan and W. Semke, “Imputing ADS-B/GPS Dropouts Using Machine Learning”, 2024 IEEE Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2024.
- B. S. Chandar, P. Ranganathan and H. Reza, “Benchmarking Federated Learning Framework for Aviation Network Data”, 2023 White Paper, Center for Cyber Security Research [*Submitted for Review*].

FUNDING ACKNOWLEDGMENT

Funding: The FAA has sponsored this project through the Center of Excellence for Unmanned Aircraft Systems. However, the agency neither endorses nor rejects the findings of this research. The presentation of this information is in the interest of invoking technical community comment on the results and conclusions of the research.

Award: 15-C-UAS-UND-030

Award Name: Mitigating GPS and ADS-B Risks for UAS 267

Program: ASSURE UAS COE

Data Availability Statement: Data is available upon request.

Conflicts of Interest: The authors declare no conflict of interest.

REFERENCES

[1] R. Baheti and H. Gill, “Cyber-physical systems,” *The impact of control technology*, vol. 12, no. 1, pp. 161–166, 2011.

[2] R. S. Nandhini and R. Lakshmanan, “A Review of the Integration of Cyber-Physical System and Internet of Things A Cyber-Physical Systems Perception of Internet of Things,” *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, p. 2022, Accessed: Mar. 24, 2023. [Online]. Available: www.ijacsa.thesai.org

[3] A. Napoleone, M. Macchi, and A. Pozzetti, “A review on the characteristics of cyber-physical systems for the future smart factories,” *J Manuf Syst*, vol. 54, pp. 305–335, Jan. 2020, doi: 10.1016/J.JMSY.2020.01.007.

[4] “Satellite Navigation - GPS - How It Works | Federal Aviation Administration.” Accessed: Nov. 17, 2023. [Online]. Available: https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/howitworks

[5] “Technology - AOPA.” Accessed: Nov. 17, 2023. [Online]. Available: https://www.aopa.org/training-and-safety/online-learning/safety-spotlights/collision-avoidance/technology

[6] “ADS-B FAQ | Federal Aviation Administration.” Accessed: Apr. 21, 2023. [Online]. Available: https://www.faa.gov/air_traffic/technology/adsb/faq#g1

[7] “AERO - New Air Traffic Surveillance Technology.” Accessed: Apr. 21, 2023. [Online]. Available: https://www.boeing.com/commercial/aeromagazine/articles/qtr_02_10/2/

[8] “SESAR Joint Undertaking | ADS-B surveillance of aircraft in flight and on the surface.” Accessed: Apr. 21, 2023. [Online]. Available: https://www.sesarju.eu/sesar-solutions/ads-b-surveillance-aircraft-flight-and-surface

[9] A. Tabassum and W. Semke, “UAT ADS-B data anomalies and the effect of flight parameters on dropout occurrences,” *Data (Basel)*, vol. 3, no. 2, 2018, doi: 10.3390/data3020019.

[10] Y. Lin, L. Deng, Z. Chen, X. Wu, J. Zhang, and B. Yang, “A Real-Time ATC Safety Monitoring Framework Using a Deep Learning Approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4572–4581, Nov. 2020, doi: 10.1109/TITS.2019.2940992.

[11] D. Wesely, A. Churchill, J. Slough, and W. J. Coupe, “A Machine Learning Approach to Predict Aircraft Landing Times using Mediated Predictions from Existing Systems.”

[12] R. Perrichon, X. Gendre, and T. Klein, “A Geometric Approach to Study Aircraft Trajectories: The Benefits of OpenSky Network ADS-B Data,” *Engineering Proceedings 2022, Vol. 28, Page 6*, vol. 28, no. 1, p. 6, Dec. 2022, doi: 10.3390/ENGPROC2022028006.

[13] H. Shafienya and A. C. Regan, “4D flight trajectory prediction using a hybrid Deep Learning prediction method based on ADS-B technology: A case study of Hartsfield-Jackson Atlanta International Airport (ATL),” 2022, doi: 10.1016/j.trc.2022.103878.

[14] R. Perrichon, X. Gendre, and T. Klein, “A Geometric Approach to Study Aircraft Trajectories: The Benefits of OpenSky Network ADS-B Data,” *Engineering Proceedings 2022, Vol. 28, Page 6*, vol. 28, no. 1, p. 6, Dec. 2022, doi: 10.3390/ENGPROC2022028006.

[15] P. N. Tran, H. Q. V. Nguyen, D. T. Pham, and S. Alam, “Aircraft Trajectory Prediction with Enriched Intent Using Encoder-Decoder Architecture,” *IEEE Access*, vol. 10, pp. 17881–17896, 2022, doi: 10.1109/ACCESS.2022.3149231.

[16] Y. Wu, H. Yu, J. Du, B. Liu, and W. Yu, “An Aircraft Trajectory Prediction Method Based on Trajectory Clustering and a Spatiotemporal Feature Network,” *Electronics 2022, Vol. 11, Page 3453*, vol. 11, no. 21, p. 3453, Oct. 2022, doi: 10.3390/ELECTRONICS11213453.

[17] C. Huang and X. Cheng, “Estimation of aircraft fuel consumption by modeling flight data from avionics systems,” *J Air Transp Manag*, vol. 99, p. 102181, 2022, doi: 10.1016/j.jairtraman.2022.102181.

[18] A. Filippone, N. Bojdo, S. Mehta, and B. Parkes, “Using the OpenSky ADS-B Data to Estimate Aircraft Emissions,” *Engineering Proceedings 2021, Vol. 13, Page 11*, vol. 13, no. 1, p. 11, Jan. 2022, doi: 10.3390/ENGPROC2021013011.

[19] F. D. A. Quadros, M. Snellen, J. Sun, and I. C. Dedoussi, “Global Civil Aviation Emissions Estimates for 2017–2020 Using ADS-B Data,” *J Aircr*, vol. 59, no. 6, pp. 1394–1405, Nov. 2022, doi: 10.2514/1.C036763/ASSET/IMAGES/LARGE/FIGURE6.JPEG.

[20] B. S. Ali, W. Ochieng, A. Majumdar, W. Schuster, and T. Kian Chiew, “ADS-B system failure modes and models,” *Journal of Navigation*, vol. 67, no. 6, pp. 995–1017, Nov. 2014, doi: 10.1017/S037346331400037X.

[21] “Mysterious interference causes planes to reroute in Texas | The Independent.” Accessed: Apr. 21, 2023. [Online]. Available: https://www.independent.co.uk/tech/gps-interference-plane-airport-texas-b2207806.html

[22] “FAA Files Reveal a Surprising Threat to Airline Safety: the U.S. Military’s GPS Tests - IEEE Spectrum.” Accessed: Apr. 21, 2023. [Online]. Available: https://spectrum.ieee.org/faa-files-reveal-a-surprising-threat-to-airline-safety-the-us-militarys-gps-tests

[23] “N.J. Man In A Jam, After Illegal GPS Device Interferes With Newark Liberty Operations - CBS New York.” Accessed: Apr. 21, 2023. [Online]. Available: https://www.cbsnews.com/newyork/news/n-j-man-in-a-jam-after-illegal-gps-device-interferes-with-newark-liberty-operations/

[24] C. Clay, M. Khan, and B. Bajracharya, “A Look into the Vulnerabilities of Automatic Dependent Surveillance-Broadcast,” in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, Mar. 2023, pp. 0933–0938. doi: 10.1109/CCWC57344.2023.10099369.

[25] S. Khandker, H. Turtiainen, A. Costin, and T. Hamalainen, “Cybersecurity Attacks on Software Logic and Error Handling Within ADS-B Implementations: Systematic Testing of Resilience and Countermeasures,” *IEEE Trans Aerosp Electron Syst*, vol. 58, no. 4, pp. 2702–2719, Aug. 2022, doi: 10.1109/TAES.2021.3139559.

[26] H. Yang, Q. Zhou, M. Yao, R. Lu, H. Li, and X. Zhang, “A practical and compatible cryptographic solution to ADS-B security,” *IEEE Internet Things J*, vol. 6, no. 2, pp. 3322–3334, Apr. 2019, doi: 10.1109/JIOT.2018.2882633.

[27] M. Yue, H. Zheng, H. Cui, and Z. Wu, “GAN-LSTM-Based ADS-B Attack Detection in the Context of Air Traffic Control,” *IEEE Internet Things J*, 2023, doi: 10.1109/JIOT.2023.3252809.

[28] P. Mykytyn, M. Brzozowski, Z. Dyka, and P. Langendoerfer, “GPS-Spoofing Attack Detection Mechanism for UAV Swarms”.

[29] M. TajDini, V. Sokolov, and P. Skladannyi, “Performing Sniffing and Spoofing Attack Against ADS-B and Mode S using Software Define Radio,” *UkrMiCo 2021 - 2021 IEEE International Conference on Information and Telecommunication Technologies and Radio Electronics, Proceedings*, pp. 7–11, 2021, doi: 10.1109/UKRMICO52950.2021.9716665.

[30] J. Wang, Y. Zou, and J. Ding, "ADS-B spoofing attack detection method based on LSTM," *EURASIP J Wirel Commun Netw*, vol. 2020, no. 1, pp. 1–12, Dec. 2020, doi: 10.1186/S13638-020-01756-8/FIGURES/11.

[31] X. Ying, J. Mazer, G. Bernieri, M. Conti, L. Bushnell, and R. Poovendran, "Detecting ADS-B Spoofing Attacks Using Deep Neural Networks," *2019 IEEE Conference on Communications and Network Security, CNS 2019*, pp. 187–195, Jun. 2019, doi: 10.1109/CNS.2019.8802732.

[32] T. Kacem, A. Kaya, A. Seydi Keceli, C. Catal, D. Wijsekera, and P. Costa, "ADS-B Attack Classification using Machine Learning Techniques," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 7–12, 2021, doi: 10.1109/IVWORKSHOPS54471.2021.9669212.

[33] D. Zuo, C. Shi, K. Jin, P. Zhao, W. Zou, and K. Cai, "A Machine Learning GNSS Interference Detection Method based on ADS-B Multi-index Features," *Integrated Communications, Navigation and Surveillance Conference, ICNS*, vol. 2023-April, 2023, doi: 10.1109/ICNS58246.2023.10124266.

[34] "ADS-B Reception Error Correction Based on the LSTM Neural-Network Model | Enhanced Reader."

[35] J. Price, H. O. Slimane, K. Al Shamaileh, V. Devabhaktuni, and N. Kaabouch, "A Machine Learning Approach for the Detection of Injection Attacks on ADS-B Messaging Systems," *2023 International Conference on Computing, Networking and Communications, ICNC 2023*, pp. 293–297, 2023, doi: 10.1109/ICNC57223.2023.10074232.

[36] D. McCallie, J. Butts, and R. Mills, "Security analysis of the ADS-B implementation in the next generation air transportation system," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 2, pp. 78–87, Aug. 2011, doi: 10.1016/j.ijcip.2011.06.001.

[37] "The Unsolved Mystery of the 2022 Texas Interference - Inside GNSS - Global Navigation Satellite Systems Engineering, Policy, and Design." Accessed: Oct. 31, 2023. [Online]. Available: https://insidegnss.com/the-unsolved-mystery-of-the-2022-texas-interference/

[38] A. R. Ismail, N. Z. Abidin, and M. K. Maen, "Systematic Review on Missing Data Imputation Techniques with Machine Learning Algorithms for Healthcare," *Journal of Robotics and Control (JRC)*, vol. 3, no. 2, pp. 143–152, Feb. 2022, doi: 10.18196/JRC.V3I2.13133.

[39] T. De Wolff, A. Cuevas, and F. Tobar, "Gaussian Process Imputation of Multiple Financial Series," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 8444–8448, May 2020, doi: 10.1109/ICASSP40776.2020.9054102.

[40] D. B. Rubin, "Biometrika Trust Inference and Missing Data," *Source: Biometrika*, vol. 63, no. 3, pp. 581–592, 1976, Accessed: Dec. 17, 2022. [Online]. Available: https://about.jstor.org/terms

[41] S. van Buuren, "Flexible imputation of missing data," p. 415.

[42] J. R. van Ginkel, M. Linting, R. C. A. Rippe, and A. van der Voort, "Rebutting Existing Misconceptions About Multiple Imputation as a Method for Handling Missing Data," *J Pers Assess*, vol. 102, no. 3, pp. 297–308, May 2020, doi: 10.1080/00223891.2018.1530680.

[43] R. J. A. Little and D. B. Rubin, "Statistical analysis with missing data".

[44] G. F. N. Berkelmans *et al.*, "Population median imputation was noninferior to complex approaches for imputing missing values in cardiovascular prediction models in clinical practice," *J Clin Epidemiol*, vol. 145, pp. 70–80, 2022, doi: 10.1016/j.jclinepi.2022.01.011.

[45] A. Skoglund and A. Westergren, “Intrusion Detection For The Con-troller Pilot Data Link Communi-cation-Detecting CPDLC attacks using machine learning Intrångsdetektering för CPDLC”, Accessed: Nov. 19, 2023. [Online]. Available: www.liu.se

[46] Q. Lan, X. Xu, H. Ma, and G. Li, “Multivariable data imputation for the analysis of incomplete credit data,” *Expert Syst Appl*, vol. 141, Mar. 2020, doi: 10.1016/j.eswa.2019.112926.

[47] S. van Buuren and K. Groothuis-Oudshoorn, “mice: Multivariate Imputation by Chained Equations in R,” *J Stat Softw*, vol. 45, no. 3, pp. 1–67, Dec. 2011, doi: 10.18637/JSS.V045.I03.

[48] A. P. Dempster, N. M. Laird, and D. B. Rubin, “ Maximum Likelihood from Incomplete Data Via the EM Algorithm ,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, Sep. 1977, doi: 10.1111/J.2517-6161.1977.TB01600.X.

[49] D. Zuo, C. Shi, K. Jin, P. Zhao, W. Zou, and K. Cai, “A Machine Learning GNSS Interference Detection Method based on ADS-B Multi-index Features,” *Integrated Communications, Navigation and Surveillance Conference, ICNS*, vol. 2023-April, 2023, doi: 10.1109/ICNS58246.2023.10124266.

[50] F. Fazzer SUKATIS, N. Mohamed NOOR, N. Afiqah ZAKARIA, A. Zia UL-SAUFIE, and A. Suwardi, “INTERNATIONAL JOURNAL OF CONSERVATION SCIENCE ESTIMATION OF MISSING VALUES IN AIR POLLUTION DATASET BY USING VARIOUS IMPUTATION METHODS”, Accessed: Apr. 22, 2023. [Online]. Available: www.ijcs.uaic.ro

[51] L. Malan, C. M. Smuts, J. Baumgartner, and C. Ricci, “Missing data imputation via the expectation-maximization algorithm can improve principal component analysis aimed at deriving biomarker profiles and dietary patterns,” *Nutrition Research*, vol. 75, pp. 67–76, Mar. 2020, doi: 10.1016/j.nutres.2020.01.001.

[52] C. Velasco-Gallego and I. Lazakis, “Real-time data-driven missing data imputation for short-term sensor data of marine systems. A comparative study,” *Ocean Engineering*, vol. 218, Dec. 2020, doi: 10.1016/j.oceaneng.2020.108261.

[53] M. T. Sattari, K. Falsafian, A. Irvem, S. S, and S. N. Qasem, “Potential of kernel and tree-based machine-learning models for estimating missing data of rainfall,” *Engineering Applications of Computational Fluid Mechanics*, vol. 14, no. 1, pp. 1078–1094, Jan. 2020, doi: 10.1080/19942060.2020.1803971.

[54] L. Breiman, “Random Forests,” vol. 45, pp. 5–32, 2001.

[55] R. Feng, D. Grana, and N. Balling, “Imputation of missing well log data by random forest and its uncertainty analysis,” *Comput Geosci*, vol. 152, Jul. 2021, doi: 10.1016/j.cageo.2021.104763.

[56] M. Kokla, J. Virtanen, M. Kolehmainen, J. Paananen, and K. Hanhineva, “Random forest-based imputation outperforms other methods for imputing LC-MS metabolomics data: A comparative study,” *BMC Bioinformatics*, vol. 20, no. 1, Oct. 2019, doi: 10.1186/s12859-019-3110-0.

[57] S. Zhang, D. Cheng, Z. Deng, M. Zong, and X. Deng, “A novel k NN algorithm with data-driven k parameter computation,” *Pattern Recognit Lett*, vol. 109, pp. 44–54, 2018, doi: 10.1016/j.patrec.2017.09.036.

[58] B. Sun, L. Ma, W. Cheng, W. Wen, P. Goswami, and G. Bai, “An improved k-nearest neighbours method for traffic time series imputation,” in *Proceedings - 2017 Chinese*

Automation Congress, CAC 2017, Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 7346–7351. doi: 10.1109/CAC.2017.8244105.

[59] J. Poulos and R. Valle, “Missing Data Imputation for Supervised Learning,” Oct. 2016, doi: 10.1080/08839514.2018.1448143.

[60] A. Jadhav, D. Pramod, and K. Ramanathan, “Comparison of Performance of Data Imputation Methods for Numeric Dataset,” *Applied Artificial Intelligence*, vol. 33, no. 10, pp. 913–933, Aug. 2019, doi: 10.1080/08839514.2019.1637138.

[61] M. C. Wang, C. F. Tsai, and W. C. Lin, “Towards missing electric power data imputation for energy management systems,” *Expert Syst Appl*, vol. 174, Jul. 2021, doi: 10.1016/j.eswa.2021.114743.

[62] C. Yan, J. Yuan, Z. Ye, and Z. Yang, “A Discrete Missing Data Imputation Method Based on Improved Multi-layer Perceptron,” *Proceedings of the 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021*, vol. 1, pp. 480–484, 2021, doi: 10.1109/IDAACS53288.2021.9661028.

[63] H. Zhang, P. Xie, and E. Xing, “Missing Value Imputation Based on Deep Generative Models,” Aug. 2018, [Online]. Available: http://arxiv.org/abs/1808.01684

[64] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, “Recurrent Neural Networks for Multivariate Time Series with Missing Values,” *Sci Rep*, vol. 8, no. 1, Dec. 2018, doi: 10.1038/s41598-018-24271-9.

[65] Z. C. Lipton, D. C. Kale, R. Wetzel, and L. K. Whittier, “Modeling Missing Data in Clinical Time Series with RNNs,” Jun. 2016, doi: 10.48550/arxiv.1606.04130.

[66] W. Cao *et al.*, “BRITS: Bidirectional Recurrent Imputation for Time Series.”

[67] S. Yang, M. Dong, Y. Wang, and C. Xu, “Adversarial Recurrent Time Series Imputation,” *IEEE Trans Neural Netw Learn Syst*, pp. 1–12, Aug. 2020, doi: 10.1109/tnnls.2020.3010524.

[68] “The OpenSky Network - Free ADS-B and Mode S data for Research.” Accessed: Dec. 17, 2022. [Online]. Available: https://opensky-network.org/

[69] “pandas - Python Data Analysis Library.” Accessed: Nov. 07, 2023. [Online]. Available: https://pandas.pydata.org/

[70] “PySpark Overview — PySpark 3.5.0 documentation.” Accessed: Nov. 07, 2023. [Online]. Available: https://spark.apache.org/docs/latest/api/python/index.html

[71] V. Abeykoon *et al.*, “Data Engineering for HPC with Python,” *Proceedings of PYHPC 2020: 9th Workshop on Python for High-Performance and Scientific Computing, Held in conjunction with SC 2020: The International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 13–21, Nov. 2020, doi: 10.1109/PYHPC51966.2020.00007.

[72] “scikit-learn: machine learning in Python — scikit-learn 1.3.2 documentation.” Accessed: Nov. 01, 2023. [Online]. Available: https://scikit-learn.org/stable/index.html

[73] “sklearn.impute.IterativeImputer — scikit-learn 1.3.2 documentation.” Accessed: Nov. 01, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html#sklearn.impute.IterativeImputer

[74] “sklearn.impute.KNNImputer — scikit-learn 1.3.2 documentation.” Accessed: Nov. 01, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

[75] X. Liu, B. Children', W. Li, W. Xu, and F. Leng, "Blood-based multi-tissue gene expression inference with Bayesian ridge regression", doi: 10.1093/bioinformatics/btaa239/5819142.

[76] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999, Accessed: Nov. 01, 2023. [Online]. Available: www.research.att.com/

[77] D. H. Lee, S. E. Woo, M. W. Jung, and T. Y. Heo, "Evaluation of Odor Prediction Model Performance and Variable Importance according to Various Missing Imputation Methods," *Applied Sciences (Switzerland)*, vol. 12, no. 6, Mar. 2022, doi: 10.3390/app12062826.

[78] "FAA Aerospace Forecast".

[79] M. R. Manesh, J. Kenney, W. C. Hu, V. K. Devabhaktuni, and N. Kaabouch, "Detection of GPS Spoofing Attacks on Unmanned Aerial Systems," *2019 16th IEEE Annual Consumer Communications and Networking Conference, CCNC 2019*, Feb. 2019, doi: 10.1109/CCNC.2019.8651804.

[80] M. P. Arthur, "Detecting signal spoofing and jamming attacks in UAV networks using a lightweight IDS," *CITS 2019 - Proceeding of the 2019 International Conference on Computer, Information and Telecommunication Systems*, Aug. 2019, doi: 10.1109/CITS.2019.8862148.

[81] G. Aissou, H. O. Slimane, S. Benouadah, and N. Kaabouch, "Tree-based Supervised Machine Learning Models for Detecting GPS Spoofing Attacks on UAS," *2021 IEEE 12th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2021*, pp. 649–653, 2021, doi: 10.1109/UEMCON53757.2021.9666744.

[82] Y. Haddad, E. Orye, and O. Maennel, "Ghost Injection Attack on Automatic Dependent Surveillance-Broadcast Equipped Drones Impact on Human Behaviour," *Proceedings - 2021 IEEE International Conference on Cognitive and Computational Aspects of Situation Management, CogSIMA 2021*, pp. 161–166, May 2021, doi: 10.1109/COGSIMA51574.2021.9475928.

[83] "What happened to GPS in Denver? - GPS World : GPS World." Accessed: Mar. 24, 2023. [Online]. Available: https://www.gpsworld.com/what-happened-to-gps-in-denver/

[84] "CISA Insights GPS Interference Event | Enhanced Reader."

[85] W. N. Chan *et al.*, "Overview of NASA's Air Traffic Management-eXploration (ATM-X) Project", Accessed: May 09, 2023. [Online]. Available: https://ntrs.nasa.gov/search.jsp?R=20180005224

[86] Y. Zhu, S. Zhang, Y. Liu, D. Niyato, and J. J. Q. Yu, "Robust federated learning approach for travel mode identification from non-IID GPS trajectories," *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, vol. 2020-December, pp. 585–592, Dec. 2020, doi: 10.1109/ICPADS51040.2020.00081.

[87] K. Guo, T. Chen, S. Ren, N. Li, M. Hu, and J. Kang, "Federated Learning Empowered Real-Time Medical Data Processing Method for Smart Healthcare," *IEEE/ACM Trans Comput Biol Bioinform*, 2022, doi: 10.1109/TCBB.2022.3185395.

[88] L. Sun and J. Wu, "A Scalable and Transferable Federated Learning System for Classifying Healthcare Sensor Data," *IEEE J Biomed Health Inform*, vol. 27, no. 2, pp. 866–877, Feb. 2023, doi: 10.1109/JBHI.2022.3171402.

[89] X. Hou, J. Wang, C. Jiang, X. Zhang, Y. Ren, and M. Debbah, "UAV-Enabled Covert Federated Learning," *IEEE Trans Wirel Commun*, Oct. 2023, doi: 10.1109/TWC.2023.3245621.

[90] J. Tursunboev, Y. S. Kang, S. B. Huh, D. W. Lim, J. M. Kang, and H. Jung, "Hierarchical Federated Learning for Edge-Aided Unmanned Aerial Vehicle Networks," *Applied Sciences 2022, Vol. 12, Page 670*, vol. 12, no. 2, p. 670, Jan. 2022, doi: 10.3390/APP12020670.

[91] "ISO/IEC/IEEE 42010:2011(en), Systems and software engineering — Architecture description." Accessed: Nov. 28, 2023. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:42010:ed-1:v1:en

[92] M. Amini Valashani and A. M. Abukari, "ERP Systems Architecture For The Modern Age: A Review of The State of The Art Technologies," *Journal of Applied Intelligent Systems and Information Sciences*, vol. 1, no. 2, pp. 70–90, Jul. 2020, doi: 10.22034/JAISIS.2020.232506.1009.

[93] G. Merugu and A. Akepogu, "Four Layered Approach to Non-Functional Requirements Analysis," Jan. 2012, Accessed: Nov. 05, 2023. [Online]. Available: https://arxiv.org/abs/1201.6141v2

[94] H. Jin, G. Liu, D. Hwang, S. Kumar, Y. Agarwal, and J. I. Hong, "Peekaboo: A Hub-Based Approach to Enable Transparency in Data Processing within Smart Homes," *Proc IEEE Symp Secur Priv*, vol. 2022-May, pp. 303–320, 2022, doi: 10.1109/SP46214.2022.9833629.

[95] R. Laigner *et al.*, "From a Monolithic Big Data System to a Microservices Event-Driven Architecture," *Proceedings - 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020*, pp. 213–220, Aug. 2020, doi: 10.1109/SEAA51224.2020.00045.

[96] T. Cerny, A. S. Abdelfattah, V. Bushong, A. Al Maruf, and D. Taibi, "Microservice Architecture Reconstruction and Visualization Techniques: A Review," *Proceedings - 16th IEEE International Conference on Service-Oriented System Engineering, SOSE 2022*, pp. 39–48, 2022, doi: 10.1109/SOSE55356.2022.00011.

[97] R. Xu, L. Zhang, and N. Ge, "Modeling and Timing Analysis for Microkernel-Based Real-Time Embedded System," *IEEE Access*, vol. 7, pp. 39547–39563, 2019, doi: 10.1109/ACCESS.2019.2906011.

[98] D. Ji, Q. Zhang, S. Zhao, Z. Shi, and Y. Guan, "MicroTEE: Designing TEE OS based on the microkernel architecture," *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*, pp. 26–33, Aug. 2019, doi: 10.1109/TRUSTCOM/BIGDATASE.2019.00014.

[99] E. M. Blount, R. Maddox, N. O'connor, and D. Wood, "Distributed Simulation Infrastructure for Researching Implementation of Evolving Concepts in the NAS", Accessed: May 03, 2023. [Online]. Available: http://www.sti.nasa.gov

[100] S. Jovic and W. Harper, "UAS in the NAS Characterization Study of TestBed Infrastructure Performance in a Distributed Simulation Environment: Baseline Analysis," 2020. [Online]. Available: http://www.sti.nasa.gov

[101] C. Fung Lai and P. V Huynh, "Air Traffic Management TestBed: Messaging Performance," 2022, Accessed: Mar. 12, 2023. [Online]. Available: http://www.sti.nasa.gov

[102] S. Nag, D. D. Murakami, N. A. Marker, M. T. Lifson, and P. H. Kopardekar, "Prototyping operational autonomy for Space Traffic Management," *Acta Astronaut*, vol. 180, pp. 489–506, Mar. 2021, doi: 10.1016/J.ACTAASTRO.2020.11.056.

[103] A.-V. Predescu and T. H. Stelkens-Kobsch, “Aviation Security Lab: A testbed for security testing of current and future aviation technologies,” Institute of Electrical and Electronics Engineers (IEEE), Nov. 2022, pp. 1–5. doi: 10.1109/dasc55683.2022.9925750.

[104] D. Zeng, A. Mahmud, and N. Wendt, “Cyber Physical Security (CPS) Extension to Air Traffic Management (ATM) Testbed”.

[105] M. Brittain, L. E. Alvarez, K. Breeden, and I. Jessen, “AAM-Gym: Artificial Intelligence Testbed for Advanced Air Mobility,” *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 2022-September, 2022, doi: 10.1109/DASC55683.2022.9925762.

[106] G. Airlangga and A. Liu, “A Novel Architectural Design for Solving Lost-Link Problems in UAV Collaboration,” *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, vol. 2021-December, pp. 380–389, 2021, doi: 10.1109/APSEC53868.2021.00045.

[107] A. T. Altun *et al.*, “The Development of an Advanced Air Mobility Flight Testing and Simulation Infrastructure,” *Aerospace 2023, Vol. 10, Page 712*, vol. 10, no. 8, p. 712, Aug. 2023, doi: 10.3390/AEROSPACE10080712.

[108] K. Kannan *et al.*, “A Simulation Architecture for Air Traffic Over Urban Environments Supporting Autonomy Research in Advanced Air Mobility.” 2023.

[109] M. Vierhauser, R. Wohlrab, M. Stadler, and J. Cleland-Huang, “AMon: A domain-specific language and framework for adaptive monitoring of Cyber-Physical Systems ☆,” *J Syst Softw*, vol. 195, p. 111507, 2023, doi: 10.1016/j.jss.2022.111507.

[110] J. Cleland-Huang, A. Agrawal, M. Vierhauser, M. Murphy, and M. Prieto, “Extending MAPE-K to support Human-Machine Teaming,” *Proceedings - 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2022*, pp. 120–131, 2022, doi: 10.1145/3524844.3528054.

[111] Z. Cai, X. Chang, and M. Li, “A Cost-Efficient Platform Design for Distributed UAV Swarm Research,” *ACM International Conference Proceeding Series*, Nov. 2021, doi: 10.1145/3503047.3503070.

[112] A. A. Alsulami and S. Zein-Sabatto, “Resilient Cyber-Security Approach for Aviation Cyber-Physical Systems Protection against Sensor Spoofing Attacks,” in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference, CCWC 2021*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021, pp. 565–571. doi: 10.1109/CCWC51732.2021.9376158.

[113] D. Carramiñana, I. Campaña, L. Bergesio, A. M. Bernardos, and J. A. Besada, “Sensors and communication simulation for unmanned traffic management,” *Sensors (Switzerland)*, vol. 21, no. 3, pp. 1–29, Feb. 2021, doi: 10.3390/s21030927.

[114] S. M. Yusuf, I. Abdullahi, A. Bappi, A. Aliyu, B. Modi, and U. Y. Ibrahim, “Towards Autonomous Multi-UAVs Surveillance Mission: A Study of Nigerian Telecommunication Masts Surveillance,” *2021 1st International Conference on Multidisciplinary Engineering and Applied Science, ICMEAS 2021*, 2021, doi: 10.1109/ICMEAS52683.2021.9692377.

[115] M. Afanasov Politecnico di Milano, I. Luca Mottola Politecnico di Milano, R. Sweden Editor, and S. Ko, “THE FlyZone TESTBED ARCHITECTURE FOR AERIAL DRONE APPLICATIONS [EXPERIMENTAL METHODS],” vol. 24, no. 1, 2020.

[116] O. Bekkouche, K. Samdanis, M. Bagaa, and T. Taleb, “A service-based architecture for enabling UAV enhanced network services,” *IEEE Netw*, vol. 34, no. 4, pp. 328–335, Jul. 2020, doi: 10.1109/MNET.001.1900556.

[117] P. T. Grogan and J. L. Stern, “Coordinating Observation at Global and Local Scales: Service-Oriented Platform to Evaluate Mission Architectures,” *International Geoscience*

and Remote Sensing Symposium (IGARSS), pp. 3837–3840, Sep. 2020, doi: 10.1109/IGARSS39084.2020.9323712.

[118] S. Crow *et al.*, “Triton: A Software-Reconfigurable Federated Avionics Testbed”.

[119] Z. Liu *et al.*, “Mission Oriented Miniature Fixed-wing UAV Swarms: A Multi-layered and Distributed Architecture,” *IEEE Trans Syst Man Cybern Syst*, vol. 52, no. 3, pp. 1588–1602, Dec. 2019, doi: 10.1109/TSMC.2020.3033935.

[120] X. Zhang, H. Wang, J. Liu, and H. Li, “CyberEarth: A virtual simulation platform for robotics and cyber-physical systems,” *IEEE International Conference on Robotics and Biomimetics, ROBIO 2019*, pp. 858–863, Dec. 2019, doi: 10.1109/ROBIO49542.2019.8961433.

[121] J. Wen, H. Ji, H. Wang, M. Zhang, D. Li, and J. Wu, “Design of a Real-Time UAV Fault Injection Simulation System,” *Proceedings of the 2019 IEEE International Conference on Unmanned Systems, ICUS 2019*, pp. 767–772, Oct. 2019, doi: 10.1109/ICUS48101.2019.8995942.

[122] V. K. Singh, M. Govindarasu, D. Porschet, E. Shaffer, and M. Berman, “Distributed Power System Simulation using Cyber-Physical Testbed Federation: Architecture, Modeling, and Evaluation,” *Proceedings - 2019 Resilience Week, RWS 2019*, pp. 26–32, Nov. 2019, doi: 10.1109/RWS47064.2019.8971970.

[123] P. A. Jorgensen, A. Waltoft-Olsen, S. H. Houmb, A. L. Toppe, T. G. Soltvedt, and H. K. Muggerud, “Building a Hardware-in-the-Loop (HiL) Digital Energy Station Infrastructure for Cyber Operation Resiliency Testing,” *Proceedings - 3rd International Workshop on Engineering and Cybersecurity of Critical Systems, EnCyCriS 2022*, pp. 9–16, 2022, doi: 10.1145/3524489.3527299.

[124] J. Kleissl *et al.*, “DERConnect-A Distributed Energy Resources Testbed for Solar Power Integration,” *e-Energy 2022 - Proceedings of the 2022 13th ACM International Conference on Future Energy Systems*, pp. 587–589, Jun. 2022, doi: 10.1145/3538637.3539633.

[125] G. Chen, Y. Qu, and D. Jin, “Cyber-Physical Simulation Testbed for MadIoT Attack Detection and Mitigation”, doi: 10.1145/3518997.3534995.

[126] A. Sahu *et al.*, “Design and evaluation of a cyber-physical testbed for improving attack resilience of power systems,” *IET Cyber-Physical Systems: Theory and Applications*, vol. 6, no. 4, pp. 208–227, Dec. 2021, doi: 10.1049/cps2.12018.

[127] G. Ravikumar, A. Singh, J. R. Babu, A. Moataz A, and M. Govindarasu, “D-IDS for cyber-physical der modbus system - Architecture, modeling, testbed-based evaluation,” in *2020 Resilience Week, RWS 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 153–159. doi: 10.1109/RWS50334.2020.9241259.

[128] J. Tabor, S. Dai, V. Sreenivasan, and S. Banerjee, “µCity: A Miniatured Autonomous Vehicle Testbed,” *Proceedings of the 17th ACM Workshop on Mobility in the Evolving Internet Architecture, MobiArch 2022*, pp. 25–30, Oct. 2022, doi: 10.1145/3556548.3559631.

[129] H. T. Nguyen and O. Haugen, “Building Experimental Laboratory for Digital Twin in Service Oriented Architecture,” *Proceedings - 2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems, ICPS 2022*, 2022, doi: 10.1109/ICPS51978.2022.9816926.

[130] K. Polachan, J. Pal, C. Singh, T. V. Prabhakar, and F. A. Kuipers, “TCPSbed: A Modular Testbed for Tactile Internet-Based Cyber-Physical Systems,” *IEEE/ACM Transactions on Networking*, vol. 30, no. 2, pp. 796–811, Apr. 2022, doi: 10.1109/TNET.2021.3124767.

[131] Y. Ye, S. Li, F. Liu, Y. Tang, and W. Hu, “EdgeFed: Optimized Federated Learning Based on Edge Computing,” *IEEE Access*, vol. 8, pp. 209191–209198, 2020, doi: 10.1109/ACCESS.2020.3038287.

[132] S. Wang *et al.*, “Adaptive Federated Learning in Resource Constrained Edge Computing Systems,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019, doi: 10.1109/JSAC.2019.2904348.

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

[134] “Software Architecture and Design InfoQ Trends Report - April 2023.” Accessed: May 04, 2023. [Online]. Available: https://www.infoq.com/articles/architecture-trends-2023/

[135] “Diffusion of Innovation Theory.” Accessed: May 10, 2023. [Online]. Available: https://www.ou.edu/deptcomm/dodjcc/groups/99A2/theories.htm

[136] S. L. P¯eeger, “Understanding and improving technology transfer in software engineering”, Accessed: May 10, 2023. [Online]. Available: www.elsevier.com/locate/jss

[137] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” *Springer Proceedings in Advanced Robotics*, vol. 5, pp. 621–635, 2018, doi: 10.1007/978-3-319-67361-5_40.

[138] A. Konttinen, “Architecture of Industrial Device Interfaces,” Nov. 2021, Accessed: May 11, 2023. [Online]. Available: https://trepo.tuni.fi/handle/10024/134839

[139] “OpenSky Network - Dataset.” Accessed: Nov. 07, 2023. [Online]. Available: https://opensky-network.org/datasets/

[140] I. Mathieson, S. Dance, L. Padgham, M. Gorman, and M. Winikoff, “An open meteorological alerting system: Issues and solutions,” *Computer Science 2004 - Proceedings of the 27th Australasian Computer Science Conference*, vol. 26, no. December, pp. 287–294, 2004, Accessed: May 11, 2023. [Online]. Available: https://researchrepository.rmit.edu.au/esploro/outputs/conferenceProceeding/An-open-meteorological-alerting-system-Issues-and-solutions/9921863652601341

[141] R. Kazman, P. Bianco, J. Ivers, and J. Klein, “Maintainability Software Solutions Division,” 2020, doi: 10.1184/R1/12954908.

[142] X. Hou, J. Wang, C. Jiang, X. Zhang, Y. Ren, and M. Debbah, “UAV-Enabled Covert Federated Learning,” *IEEE Trans Wirel Commun*, 2023, doi: 10.1109/TWC.2023.3245621.

[143] M. Fu, Y. Shi, S. Member, and Y. Zhou, “Federated Learning via Unmanned Aerial Vehicle,” Oct. 2022, Accessed: May 11, 2023. [Online]. Available: https://arxiv.org/abs/2210.10970v1

[144] A. Taik and S. Cherkaoui, “Electrical Load Forecasting Using Edge Computing and Federated Learning,” *IEEE International Conference on Communications*, vol. 2020-June, Jun. 2020, doi: 10.1109/ICC40277.2020.9148937.

[145] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, “Federated-Learning-Based Anomaly Detection for IoT Security Attacks,” *IEEE Internet Things J*, vol. 9, no. 4, pp. 2545–2554, Feb. 2022, doi: 10.1109/JIOT.2021.3077803.

[146] S. Khan, G. Singh Gaba, and A. Gurtov, “A Federated Learning Based Security for Controller Pilot Data Link Communication,” 2022, Accessed: Oct. 18, 2023. [Online]. Available: https://www.researchgate.net/publication/366683346

[147] M. Wang, Y. Lei, Y. Liang, X. Lv, and W. Mo, “AGI-FEDAVG: A UAV Power Line Inspection Algorithm Based on Federated Learning,” *Proceedings of 2021 IEEE 3rd*

International Conference on Civil Aviation Safety and Information Technology, ICCASIT 2021, pp. 1030–1034, 2021, doi: 10.1109/ICCASIT53235.2021.9633432.

[148] "Docker: Accelerated Container Application Development." Accessed: Nov. 07, 2023. [Online]. Available: https://www.docker.com/

[149] R. Fielding *et al.*, "Hypertext Transfer Protocol -- HTTP/1.1," Jun. 1999, doi: 10.17487/RFC2616.

[150] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," May 2015, doi: 10.17487/RFC7540.

[151] I. Olivos and M. Johansson, "Comparative Study of REST and gRPC for Microservices in Estab-lished Software Architectures", Accessed: Nov. 12, 2023. [Online]. Available: www.liu.se

[152] Y. Fu and C. Soman, "Real-time Data Infrastructure at Uber," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 2503–2516, 2021, doi: 10.1145/3448016.3457552.

[153] Z. Niu, L. Dong, and Y. Zhu, "The Runtime model checking Method for Zero Trust Security Policy," p. 2022, doi: 10.1145/3558819.3558821.

[154] Z. Adahman, W. Malik, and Z. Anwar, "An analysis of zero-trust architecture and its cost-effectiveness for organizational security," *Comput Secur*, vol. 122, p. 102911, 2022, doi: 10.1016/j.cose.2022.102911.

[155] D. C. Nguyen *et al.*, "Federated Learning for Smart Healthcare: A Survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, p. 60, Feb. 2022, doi: 10.1145/3501296.

[156] B. Ghimire and D. B. Rawat, "Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things," *IEEE Internet Things J*, vol. 9, no. 11, pp. 8229–8249, Jun. 2022, doi: 10.1109/JIOT.2022.3150363.

[157] M. Driss, I. Almomani, · Zil E Huma, and J. Ahmad, "A federated learning framework for cyberattack detection in vehicular sensor networks," *Complex & Intelligent Systems*, vol. 8, pp. 4221–4235, 2022, doi: 10.1007/s40747-022-00705-w.

[158] TaibiDavide and MezzaliraLuca, "Micro-Frontends," *ACM SIGSOFT Software Engineering Notes*, vol. 47, no. 4, pp. 25–29, Sep. 2022, doi: 10.1145/3561846.3561853.

[159] J. Männistö, A.-P. Tuovinen, and M. Raatikainen, "Experiences on a Frameworkless Micro-Frontend Architecture in a Small Organization," *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*, pp. 61–67, Mar. 2023, doi: 10.1109/ICSA-C57050.2023.00025.

[160] H. Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise AgüeraAg and A. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," 2017.

[161] D. Gao, C. Ju, X. Wei, Y. Liu, T. Chen, and Q. Yang, "HHHFL: Hierarchical Heterogeneous Horizontal Federated Learning for Electroencephalography," Sep. 2019, Accessed: Nov. 11, 2023. [Online]. Available: https://arxiv.org/abs/1909.05784v3

[162] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data." Sep. 23, 2019.

[163] T.-M. H. Hsu, H. Qi, G. Research, and M. Brown, "Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification," Sep. 2019, Accessed: Nov. 11, 2023. [Online]. Available: https://arxiv.org/abs/1909.06335v1

[164] "Flower: A Friendly Federated Learning Framework." Accessed: Nov. 11, 2023. [Online]. Available: https://flower.dev/

[165] “Barathwaja/federated-learning: This will consists of all the Federated Learning codes.” Accessed: Nov. 21, 2023. [Online]. Available: https://github.com/Barathwaja/federated-learning

[166] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/NECO.1997.9.8.1735.

[167] “Getting started.” Accessed: Nov. 11, 2023. [Online]. Available: https://keras.io/getting_started/

[168] A. Mirzal, “Statistical Analysis of Microarray Data Clustering using NMF, Spectral Clustering, Kmeans, and GMM,” *IEEE/ACM Trans Comput Biol Bioinform*, vol. 19, no. 2, pp. 1173–1192, 2022, doi: 10.1109/TCBB.2020.3025486.

[169] “tslearn’s documentation — tslearn 0.6.2 documentation.” Accessed: Nov. 11, 2023. [Online]. Available: https://tslearn.readthedocs.io/en/stable/#

[170] “Grafana: The open observability platform | Grafana Labs.” Accessed: Nov. 11, 2023. [Online]. Available: https://grafana.com/

[171] “Prometheus - Monitoring system & time series database.” Accessed: Nov. 11, 2023. [Online]. Available: https://prometheus.io/

[172] “google/cadvisor: Analyzes resource usage and performance characteristics of running containers.” Accessed: Nov. 11, 2023. [Online]. Available: https://github.com/google/cadvisor

[173] C. Zhang, F. Zhang, X. Guo, B. He, X. Zhang, and X. Du, “IMLBench: A Machine Learning Benchmark Suite for CPU-GPU Integrated Architectures,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1740–1752, Jul. 2021, doi: 10.1109/TPDS.2020.3046870.

[174] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, “Toward Personalized Federated Learning,” *IEEE Trans Neural Netw Learn Syst*, 2022, doi: 10.1109/TNNLS.2022.3160699.

[175] T. Zang, C. Zheng, S. Ma, C. Sun, and W. Chen, “A General Solution for Straggler Effect and Unreliable Communication in Federated Learning,” *ICC 2023 - IEEE International Conference on Communications*, pp. 1194–1199, May 2023, doi: 10.1109/ICC45041.2023.10279635.

[176] H. Wu, P. Wang, and C. V. A. Narayana, “Straggler-resilient Federated Learning: Tackling Computation Heterogeneity with Layer-wise Partial Model Training in Mobile Edge Network,” Nov. 2023, Accessed: Nov. 21, 2023. [Online]. Available: https://arxiv.org/abs/2311.10002v1