

CONTINUAL AND FEDERATED LEARNING FOR SPACE OPERATIONS

BRUNO SÁNCHEZ GÓMEZ

Thesis supervisor
JAMES MICHEL EGGLESTON (European Space Agency (ESA))

Tutor: JAVIER BÉJAR ALONSO (Department of Computer Science)

Degree
Master's Degree in Artificial Intelligence

Master's thesis

School of Engineering
Universitat Rovira i Virgili (URV)

Faculty of Mathematics
Universitat de Barcelona (UB)

Barcelona School of Informatics (FIB)
Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Abstract

Federated Continual Learning (FCL) has recently emerged as a promising solution for collaborative model training in dynamic, non-stationary environments. This paradigm is particularly relevant to the domain of Spacecraft Prognostics and Health Management (PHM), where fleets of similar satellites can benefit from shared knowledge while preserving data privacy and adapting to evolving operational conditions and data trends. This paper proposes an FCL framework enhanced with Concept Drift (CD) detection mechanisms, which trigger model updates more effectively by identifying significant changes in data distribution. We investigate different State-of-the-Art approaches and methods for CD detection and provide a comparative analysis to assess which of them better fit our case. The experiments are conducted on two datasets: the public ESA-ADB, used only for hyperparameter tuning of the CD mechanism, and a dataset extracted from a flying fleet of spacecraft, comprising several similar satellites that serve as natural clients in our federated scenario.

Keywords: artificial intelligence, machine learning, concept drift, federated learning, continual learning, prognostics and health management, anomaly detection, time series forecasting

Contents

1 Introduction

The domain of Spacecraft Prognostics and Health Management (PHM) is essential for ensuring the safety, reliability, and longevity of space missions [1, 2]. The proliferation of large satellite constellations has led to an unprecedented volume of telemetry data, creating significant opportunities for data-driven PHM [3]. The core task is to continuously monitor these data streams to detect anomalies and predict potential failures. However, this task is complicated by several fundamental challenges. Firstly, spacecraft telemetry data is sensitive and often subject to strict privacy and security constraints, making it impractical to centralize raw data from an entire fleet for model training [4]. Secondly, anomalies are typically rare events for any single spacecraft, and machine learning models often require a large number of examples to achieve high performance, making it difficult to train a robust model on data from a single satellite [5]. Finally, the operational environment of a spacecraft is inherently non-stationary; data distributions shift over time due to factors like sensor degradation, changing operational modes, or evolving environmental conditions [6]. A model trained on historical data may see its performance degrade significantly as the underlying data properties change, a phenomenon known as concept drift [7].

To address these challenges, several machine learning paradigms have been developed. **Federated Learning** (FL) was introduced to enable privacy-preserving, collaborative machine learning [8]. Its most common implementation, Federated Averaging (FedAvg), involves a central server that orchestrates training rounds with a set of clients [9]. The application of FL to time series data, particularly in industrial and IoT settings, has been an area of growing interest [10]. Representative application areas include predictive maintenance and anomaly detection in Industrial IoT (IIoT) where each machine or edge device holds local sensor time series and collaboratively trains a global model without sharing raw telemetry [11–13]. FL has also been applied to energy/load forecasting and smart-grid telemetry, healthcare wearable time series (patient monitoring), and distributed sensor networks for environmental monitoring; in all these domains the combination of temporal structure, label scarcity (for anomalies/failures), and privacy constraints motivates federated architectures or hybrid edge-server designs [11, 13, 14]. Practical works additionally address streaming-specific FL concerns such as asynchronous rounds, client drift, compression/pruning for resource-limited edge nodes, and federated protocols designed for continual/streaming updates rather than batch-only training [13, 15].

To handle non-stationary data, **Continual Learning** (CL) aims to develop models that can learn from a continuous data stream without forgetting previously acquired knowledge [16]. Catastrophic forgetting is the principal challenge: when models update on new data they can rapidly lose performance on earlier regimes. CL strategies fall into three broad families with many practical variants and applications. **Regularization-based** methods constrain updates to important parameters (e.g., Elastic Weight Consolidation, EWC), which has been effective in many sequential-task settings where task identities or importance estimates are available [17, 18]. **Memory / replay-based** methods store a (possibly compressed) subset of past examples (experience replay) or synthesize past data (generative replay) to rehearse earlier regimes while learning new ones; simple experience-replay buffers have proven surprisingly strong baselines in both reinforcement-learning and supervised continual tasks and are widely used for time-series and streaming problems [19, 20]. **Parameter-isolation and architectural** methods allocate or protect parameter subsets for different tasks (e.g., PackNet, winning subnetworks / sparsity-based approaches), which can avoid interference at the cost of increased memory or management complexity [21, 22]. In applied settings these families are often hybridized: for example, a small replay buffer plus a light regularizer gives a robust trade-off between plasticity and stability for streaming forecasting problems [16, 23]. Recent work also explores continual methods tailored to time

series forecasting (online forecasting, groundwater/energy forecasting and industrial sensor streams), showing that replay-based schemes and light consolidation rules are practical and performant for temporally correlated data [24]. For this work we selected Experience Replay, a memory-based approach, based on a preliminary study that showed its effectiveness for our time-series forecasting task [25].

Finally, **Concept Drift** (CD) detection methods have been developed to identify changes in the underlying data distribution over time, which invalidates models trained on outdated data [6, 7]. The ADaptive WINdowing (ADWIN) algorithm [26] and the Page-Hinkley test [27] are two widely used examples that monitor data or model performance to detect significant changes. ADWIN is an adaptive sliding window algorithm that detects changes by keeping updated statistics about a data stream and adjusting the window size when the data distribution changes [26]. The Page-Hinkley test is a sequential analysis technique that monitors the cumulative deviation of a variable from its running mean, signaling a change when this deviation exceeds a predefined threshold [27]. Expanding beyond those two, the literature includes: the Drift Detection Method (DDM) which monitors a learner's error-rate and raises warnings/drift alarms based on statistically significant increases [28]; the Early Drift Detection Method (EDDM) that tracks distances between classification errors and is tuned to detect gradual drift [29]; KSWIN, a Kolmogorov-Smirnov windowing test for distributional changes; and a family of Hoeffding/variance-based detectors (HDDM variants) and CUSUM-like tests adapted to streaming ML [30, 31]. Important distinctions in the drift literature are **error-based / model-performance detectors** (DDM, EDDM, and ADWIN or PH when used on errors) that require true values or labels, and **input-based / unsupervised detectors** (KSWIN, statistical tests, reconstruction-loss monitors, or fully unsupervised drift detectors) that can run without immediate feedback and are therefore attractive for model-unbiased drift detection [7, 32]. Concept drift techniques have been evaluated in many PHM-like domains (wind-turbine fault detection, industrial CPS, and predictive maintenance pipelines) showing that drift detection (coupled with targeted adaptation) improves the timeliness and robustness of anomaly/failure detection pipelines in real deployments [33–35].

While FL protects data privacy and CL handles evolving data streams, applying them in isolation is insufficient for the spacecraft PHM domain. Standard FL frameworks often assume that data distributions are static, and their performance suffers under concept drift [36]. On the other hand, traditional CL methods are typically designed for centralized data settings [37]. A naive solution to model degradation is periodic retraining, where the model is frequently updated on new data. However, this approach is inefficient, potentially retraining on stable data (wasting computational resources) or reacting too slowly to sudden changes (lagging behind a drift) [7, 38]. This creates a clear need for a paradigm that unifies the strengths of both approaches while addressing the inefficiency of periodic model updates. The integration of these fields into a unified Federated Continual Learning (FCL) framework is an emerging research direction, but a key open question is determining the optimal strategy for triggering model updates [39].

To address this, we propose a Concept Drift-aware Federated Continual Learning framework. By integrating explicit concept drift detection mechanisms at the client level, our system can trigger model updates more intelligently and efficiently. When a significant change in the local data distribution is detected on a satellite, it initiates a training round. This asynchronous, event-driven approach ensures that the learning process is both resource-efficient and highly responsive to changes in the operational state of the spacecraft. In this paper, we present a comprehensive study on the application of this framework to spacecraft PHM, focusing on time series forecasting and subsequent anomaly detection. Our main contributions are:

- A novel framework that integrates Federated Learning, Continual Learning, and client-side concept drift detection for spacecraft PHM.
- An extensive experimental evaluation on a confidential dataset from a real-world satellite constellation, where each satellite acts as a natural client in the federated setting.
- A comparative analysis of an asynchronously updated model triggered by a concept drift detection mechanism against a baseline FL model and a periodically retrained FCL model.
- An investigation into the impact of using a locally fine-tuned model for inference versus a global aggregated model.

It is relevant to note that Concept Drift-aware Federated Continual Learning has also been explored previously, though with a different approach. Casado et al. propose a strategy that couples a custom drift detector with a drift-aware update rule for federated aggregation [40]. In contrast, our work systematically evaluates different configurations of established drift detectors (ADWIN and Page-Hinkley) with a Continual Learning algorithm that had been previously validated for our forecasting use case (Experience Replay), emphasizing empirical assessment of practical design choices rather than introducing a new detector or aggregation rule. Moreover, their evaluation is conducted on a Human Activity Recognition dataset from smartphones, which differs markedly from spacecraft PHM in operational conditions, data characteristics and task goals.

The remainder of this paper is organized as follows: Section 2 details our proposed methodology. Section 4 describes the experimental setup and presents the results of our study. Finally, Section 6 concludes the paper and outlines potential directions for future research.

2 Methodology

Our proposed methodology integrates a time series forecasting model, a federated continual learning framework, and a concept drift detection mechanism to create a comprehensive PHM system for spacecraft.

2.1 Forecasting and Anomaly Detection

At the core of our system is a forecasting model based on Telemanom, an LSTM-based deep learning model that has proven to be effective for anomaly detection in spacecraft telemetry [41]. We train the Telemanom model on nominal (i.e., non-anomalous) time series data to predict future telemetry values. The model is trained to minimize the reconstruction error, which is the difference between the predicted and actual values. Note that the aim of this work is not to identify the optimal ML model, but to validate the proposed framework; accordingly, we adopt a well-established and validated approach, allowing us to focus on federated continual learning and drift-triggered updates.

At inference time, we compute per-time-step forecasting errors as an exponentially-weighted moving average (EWMA) of the mean absolute error (MAE). The EWMA uses a smoothing window of size $w = B \cdot S_\mathrm{s} \cdot p_\mathrm{s}$, where B is the batch size, S_s is a smoothing span, and p_s is a smoothing percentile controlling effective window length.

Once trained, we detect anomalies via a channel-wise rolling-window thresholding of these smoothed forecasting errors. For each stride of time steps, we compute per-channel thresholds from the recent window by:

1. Sorting the errors in the window and keeping the lowest 99% to reduce outlier influence;
2. Computing the mean μ and standard deviation σ on this trimmed set;
3. Setting the threshold as $\alpha \cdot \mu + \beta \cdot \sigma$ (with fixed hyperparameters $\alpha, \beta > 0$)

The resulting threshold is applied to the corresponding stride, and any error exceeding it is flagged as an anomaly. Finally, we pad detections by ± 1 day to provide temporal context and tolerance around detected events.

Variable	**Value**
Telemanom input size	250
LSTM layer sizes	[80, 80]
Rolling window size	5000
Rolling window stride	70
Threshold mean scale (α)	1.3
Threshold std scale (β)	8.5
EWMA smoothing span (S_s)	30
EWMA smoothing percentile (p_s)	0.05

Table 1: Default hyperparameter values used in forecasting and anomaly detection.

2.2 Federated Continual Learning

To enable collaborative learning across a fleet of satellites while preserving data privacy, we employ a Federated Continual Learning (FCL) framework. In our setup, each satellite acts as a client, and a central server coordinates the learning process. The training proceeds in rounds, with each round consisting of the following steps:

1. The central server sends the current global model to the clients.
2. Each client trains the model on its local data.
3. The clients send their updated model parameters back to the server.
4. The server aggregates the client models (e.g., using Federated Averaging) to produce a new global model.

We investigate two different strategies for using the models for inference. In the first approach, which represents the standard practice, all clients use the aggregated global model for forecasting and anomaly detection. In the second approach, which we call "finetuning", each client uses its own locally trained model (before the aggregation step) for inference. This allows the client models to be more specialized to the specific characteristics of each satellite, while still benefiting from the knowledge shared through the federated learning process.

In the asynchronous setting, we wait for the *Retrain Wait* period (two weeks) *after* a drift is detected to accumulate representative data from the new concept before retraining.

Parameter	Value
CL Algorithm	Experience Replay [42]
Retrain Period (Sync FCL)	16 weeks
Retrain Wait (Async FCL)	2 weeks
# of Clients	1 or 5
# of Federated Rounds	2

Table 2: Federated Continual Learning (FCL) configuration.

2.3 Concept Drift-Triggered Updates

To make the continual learning process more efficient and responsive, we incorporate a concept drift detection mechanism. Instead of retraining the models at fixed intervals, we use a drift detector to monitor the data stream on each client. When a concept drift is detected on a client, it triggers a retraining of the model.

We explored two approaches for drift detection: input-based, which monitors the model's input data, and error-based, which monitors the model's prediction error. For each approach, we evaluated two drift detection methods: the Adaptive Windowing (ADWIN) algorithm and the Page-Hinkley (PH) test. We tuned two key hyperparameters for each method: *delta* and *minimum window length* for ADWIN, and the *threshold* and *minimum number of instances* for the PH-test.

We compare two different retraining strategies:

- **Synchronous Periodic Retraining:** The models are retrained at fixed intervals, with synchronous updates across all clients (see Algorithm 1).
- **Asynchronous Drift-Triggered Retraining:** Each client has its own drift detection mechanism. When a drift is detected on a client, it initiates a retraining process. This results in asynchronous updates, as different clients may experience concept drift at different times. The server's aggregation strategy is adapted to incorporate individual client updates as they occur, using FedAsync [43] (see Algorithm 2).

By comparing these strategies, we aim to evaluate the benefits of using a concept drift detection mechanism to guide the continual learning process.

Algorithm 1 Synchronous Periodic Retraining (with FedAvg)

Input: Global model w; clients $k = 1, \ldots, N$; retraining period P; replay buffer $\mathcal{R}$, number of federated rounds N_r

Output: Updated global model w and client inference models $\tilde{w}$

Client-side (runs continuously):

initialize $\tilde{w} \leftarrow w$; $\mathcal{B} \leftarrow \emptyset$; $t_0 \leftarrow 0$
for each new sample (x_t, y_t) **do**
  append (x_t, y_t) to $\mathcal{B}$
  if $(t - t_0) = P$ **then**
    $\mathcal{R}$.ExtendDataset($\mathcal{R}, \mathcal{B}$)
    set $n_k \leftarrow |\mathcal{R}|$
    for $r = 1, \ldots, N_r$ **do**
      receive w from server
      $w'_k \leftarrow$ ContinualLearning(w; $\mathcal{R}$)
      send (w'_k, n_k) to server
    end for
    if *Client-side finetuning* **then**
      set $\tilde{w} \leftarrow w'_k$
    else
      receive w from server
      set $\tilde{w} \leftarrow w$
    end if
    set $\mathcal{B} \leftarrow \emptyset$
  end if
end for

Server-side (every period P):

receive updates (w'_k, n_k) for all clients $k = 1, \ldots, N$
set $w \leftarrow \sum_{k=1}^{N} \frac{n_k}{\sum_j n_j} w'_k$ *(FedAvg)*
broadcast w to all clients

Algorithm 2 Asynchronous Drift-Triggered Retraining (with FedAsync)

Input: Global model w; clients $k = 1, \ldots, N$, per-client drift detectors D_k, EWMA error function $\mathcal{E}$, approach $A \in \{\text{input}, \text{error}\}$, wait period Δ, replay buffer $\mathcal{R}$, new update weight α
Output: Updated global model w and client inference models $\tilde{w}$

Client-side (runs continuously):

initialize $\tilde{w} \leftarrow w$; *waiting*←false; $\mathcal{B} \leftarrow \emptyset$; $t_0 \leftarrow 0$
for each new sample (x_t, y_t) **do**
  $\hat{y}_t \leftarrow f(x_t; \tilde{w})$; $e_t \leftarrow \mathcal{E}(y_t, \hat{y}_t)$
  if $A =$ input **then**
    D_k.update(x_t)
  else
    D_k.update(e_t)
  end if
  if D_k signals drift and *waiting*=false **then**
    set *waiting*←true; $t_0 \leftarrow t$
  end if
  append (x_t, y_t) to $\mathcal{B}$
  if *waiting* and $(t - t_0) \geq \Delta$ **then**
    $\mathcal{R}$.ExtendDataset($\mathcal{R}, \mathcal{B}$)
    $w'_k \leftarrow$ ContinualLearning(w; $\mathcal{R}$)
    send (w'_k, k) to server
    if *Client-side finetuning* **then**
      set $\tilde{w} \leftarrow w'_k$
    else
      receive w from server
      set $\tilde{w} \leftarrow w$
    end if
    set *waiting*←false; $\mathcal{B} \leftarrow \emptyset$
  end if
end for

Server-side (on update arrival (w'_k, k)):

receive w'_k from client k
set $w \leftarrow (1 - \alpha)\, w + \alpha\, w'_k$ *(FedAsync)*
broadcast w to all clients

3 Implementation

This section describes the implementation details of the proposed methodology. It includes information about the software libraries used, the structure of the codebase, and any specific implementation choices made to optimize performance or ensure scalability.

4 Experiments

We conducted two sets of experiments to evaluate our proposed methodology. The first set of experiments consisted of a series of hyperparameter grid searches for the concept drift detection mechanism. The goal of these searches was to find the optimal hyperparameter

configuration for the drift detection mechanism on different datasets.

The second set of experiments focused on federated continual learning, where we compared the performance of different retraining strategies. The goal of these experiments was to evaluate the effectiveness of our proposed asynchronous drift-triggered retraining strategy against other common approaches.

4.1 Datasets

We used two datasets in our study:

- **ESA-ADB:** A public dataset from the European Space Agency, consisting of telemetry data from two different missions. While this dataset provides a large quantity of data, it does not have a natural split into clients for federated learning. We primarily used this dataset for the hyperparameter grid search experiments for the concept drift mechanism, specifically on the "Mission1" subset.
- **Constellation Dataset:** An anonymized, confidential dataset consisting of telemetry data from a number of satellites that share similar properties. This dataset is well-suited for our federated learning experiments, as each satellite can be treated as a separate client.

4.2 Experimental Setup

For all experiments, we used an initial training period of 16 weeks of data to simulate a scenario where the PHM system is deployed early in a mission's lifecycle, highlighting the benefits of continual learning in a data-limited setting.

To prevent data leakage, we performed an initial 50–50 train-test split of each spacecraft's data stream. For each experiment, all sensor values and errors are normalized to a N (0,1) distribution, using the mean and standard deviation of the corresponding training set. The specific usage of the train and test sets differed between the two sets of experiments, as detailed below.

4.2.1 Hyperparameter Grid Searches

For the grid search experiments, the goal was to find the best hyperparameter configuration for the drift detection mechanism. The data was used as follows:

- **Initial Training:** The first 16 weeks of the train set were used for initial model training.
- **Evaluation (Continual Learning):** The continual learning loop was run on the remainder of the train set. This part of the data was used to evaluate the performance of different hyperparameter configurations.
- **Validation:** Within the continual learning loop, a 20% validation set was used during each retraining phase to enable early stopping.
- **Testing:** The final, best-performing hyperparameter configuration was evaluated on the held-out test set. A model pre-trained on the final 16 weeks of the train set was used as a starting point for this evaluation.

4.2.2 Federated Continual Learning

For the FCL experiments, the goal was to compare different FCL strategies using the best hyperparameters found during the grid search. The data usage was structured differently:

- **Initial Training:** The last 16 weeks of the train set were used for the initial training of the federated models.
- **Validation:** In the same way as for the first experiment set, a 20% validation set was used during each retraining phase of the continual learning loop to enable early stopping.
- **Evaluation:** The different FCL strategies were then directly evaluated on the test set to analyze their performance in a realistic scenario.

This approach was chosen because the hyperparameters were already selected, so a separate evaluation set for hyperparameter tuning was not necessary.

4.3 Hyperparameter Grid Search Experiments

Concept drift detection mechanisms are known to be highly sensitive to the scale and specific distribution of the data. Therefore, we performed a series of hyperparameter grid searches for the drift detection mechanism. These searches were conducted for the Sat03 and Sat18 satellites from the Constellation dataset, and for Mission1 from the ESA-ADB dataset.

Due to the high computational cost of running a grid search within a federated learning framework, these experiments were performed outside of the federated setting (i.e., they were conducted in a "centralized" manner, with only one client per experiment). The results of the grid search for Mission1 were used to validate and compare the results with those obtained for the Constellation satellites, but were not used in the subsequent FCL experiments.

The grid searches were performed on several datasets to gain different insights:

- **Sat03:** To study the effect of the different approaches, methods, and hyperparameter values.
- **Sat18:** To assess whether the patterns observed in Sat03 generalize to other satellites.
- **Mission1 (ESA-ADB):** To study the generalizability of the behaviors observed in the Constellation dataset to a different mission.
- **Sat03-dt:** To evaluate the impact of injecting external engineering knowledge, in this case by detrending the data.

For the Constellation datasets, we used a 6×5 grid for the hyperparameter search, while a 4×4 grid was used for the ESA-ADB dataset. The specific hyperparameter values tested are detailed in Appendix A. These values were selected based on preliminary experiments using pre-computed reconstruction errors from a periodic retraining strategy (for the error-based approach, as the input-based approach requires only the data stream itself), where the focus was on identifying a collection of values that obtained a well spread-out amount of drift detections.

The performance of each configuration was evaluated based on two metrics: the trimmed Mean Absolute Error (MAE) for the forecasting task, and the F0.5-Score for the anomaly

detection task. The MAE was calculated after removing the top 5% of errors to mitigate the impact of anomalous points on the evaluation. For the F0.5-Score, precision was computed using the correction from [44], which ensures that the trivial algorithm of always predicting anomalies receives a precision of 0. The F0.5-Score was chosen to prioritize precision over recall, as false positives are more detrimental in a PHM context.

4.3.1 Hyperparameter Search (Sat03)

We begin by presenting the results of the grid search experiments on the Sat03 dataset. This satellite was selected due to its high number of anomalies, as compared to the other satellites, which makes it particularly suitable for evaluating the performance of the anomaly detection task.

First, we provide an overview of the distribution of performance across different approaches and methods in Figure 1. This figure illustrates the variability in performance for both the forecasting (Mean Error) and anomaly detection (F0.5 Score) tasks.

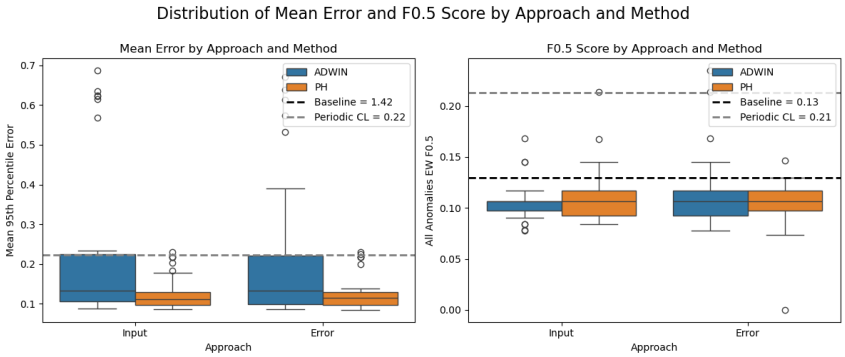

Figure 1: Distributions of performance per approach and method for Sat03.

In the error boxplots, we observe that all of the Concept Drift-Aware approaches (i.e., *CDA*) outperform the *Baseline* (no continual learning), and most also improve over the periodic retraining approach (i.e., *Periodic CL*). There seems to be no significant difference between the input-based and error-based approaches; however, the PH method tends to yield considerably more consistent (and slightly better) results than ADWIN.

Meanwhile, in the F0.5 Score boxplots, we see that most of the drift-triggered approaches in fact underperform compared to the Baseline and the Periodic CL approaches, with only a few configurations achieving similar or better results as the latter. In this case, there does not seem to be any consistent difference between the methods or approaches.

A key observation that we can make from these results is that there seems to be a trade-off between optimizing for the forecasting task and the anomaly detection task, as the CDA configurations clearly and consistenly improve the performance on the former, while most of them degrade the performance on the latter. This suggests that the two tasks are perhaps somewhat misaligned and may require different strategies for optimal performance.

Next, we analyze the effect of the number of retrainings on performance, as shown in Figure 2.

Here, we observe a clear trend where an increasing number of retrainings leads to better performance on the forecasting task (lower Mean Error). Notably, all of the Concept Drift-Aware (CDA) approaches with 5 and 6 retraining phases achieve, or even surpass,

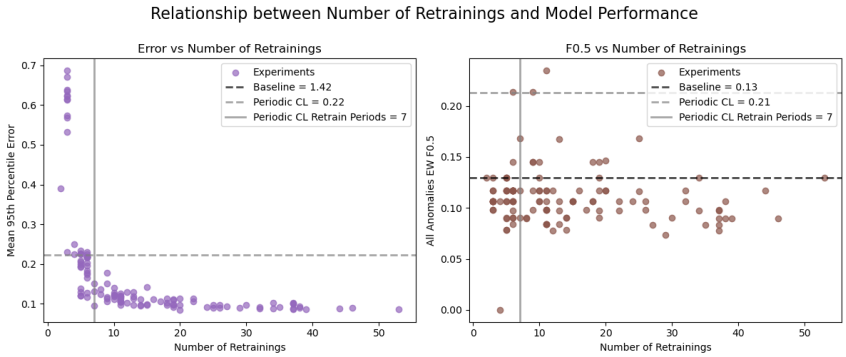

Figure 2: Effect of the number of retrainings on performance for Sat03.

the forecasting performance of the Periodic CL strategy, which uses 7 retraining periods. Furthermore, all CDA configurations with an equal or greater number of retraining phases consistently improve the mean error compared to Periodic CL. This underscores how CDA mechanisms can identify key moments for retraining, making the updates more effective than simply retraining at fixed intervals.

However, for the anomaly detection task (F0.5 Score), there is no clear trend; in fact, the performance seems to be quite variable regardless of the number of retrainings. This further supports the idea that optimizing for anomaly detection may require a different approach than simply increasing the frequency of model updates.

Finally, we study the performance trends across different hyperparameter values for both ADWIN and Page-Hinkley. To assess whether the choice of hyperparameter values leads to significant differences in performance, we performed ANOVA tests for each of the two tasks (forecasting and anomaly detection) across the different values of each hyperparameter. The results showed no significant differences in performance for either task with respect to the *delta* (ADWIN) or *threshold* (PH) hyperparameters. In contrast, there were significant differences observed for the *minimum window length* (ADWIN) and *minimum number of instances* (PH) hyperparameters. These trends can be analyzed in detail in the heatmaps provided in Appendix B, which provide insights into how sensitive each method is to its hyperparameters. For now, we focus our analysis on the more insightful *minimum window length* (ADWIN) and *minimum number of instances* (PH) hyperparameters, as shown in Figure 3.

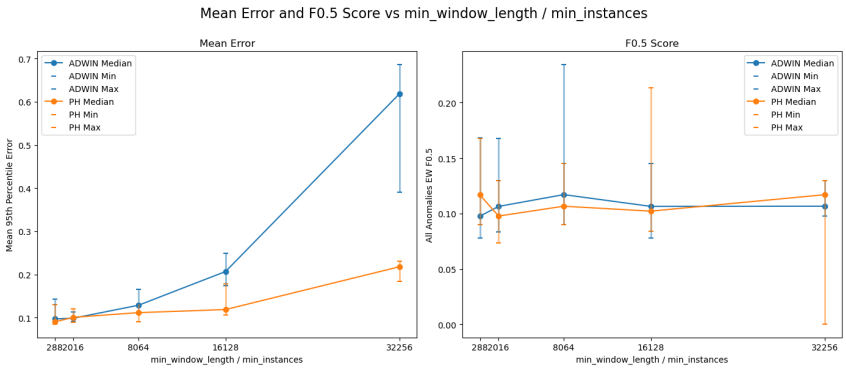

Figure 3: Performance trends for *min_window_length* (ADWIN) and *min_instances* (PH) on Sat03.

In the forecasting task, we observe a clear trend where higher values of the *minimum window length* (ADWIN) and *minimum number of instances* (PH) lead to worse performance (higher Mean Error). This suggests that setting these parameters too high may cause the drift detection mechanism to be less responsive to changes in the data, resulting in fewer retrainings and thus poorer adaptation to new patterns. Conversely, lower values for these hyperparameters seem to facilitate more frequent retrainings, which in turn improves forecasting accuracy.

For the anomaly detection task (F0.5 Score), while no general trend is observed for any of the hyperparameters, there are some specific combinations that appear to strike notably better results than the rest. These promising configurations warrant further investigation and validation to draw robust conclusions about their effectiveness and generalizability.

4.3.2 Generalization (Sat18 & Mission1)

Next, we present the results of the grid search experiments on the Sat18 and Mission1 datasets. The Sat18 satellite was chosen due to its similar proportion of anomalies as Sat03, which allows us to assess whether the patterns observed in Sat03 generalize to other satellites. The Mission1 dataset from ESA-ADB was selected to evaluate the generalizability of the behaviors observed in the Constellation dataset to a different mission. The results of these experiments are summarized in Figure 4.

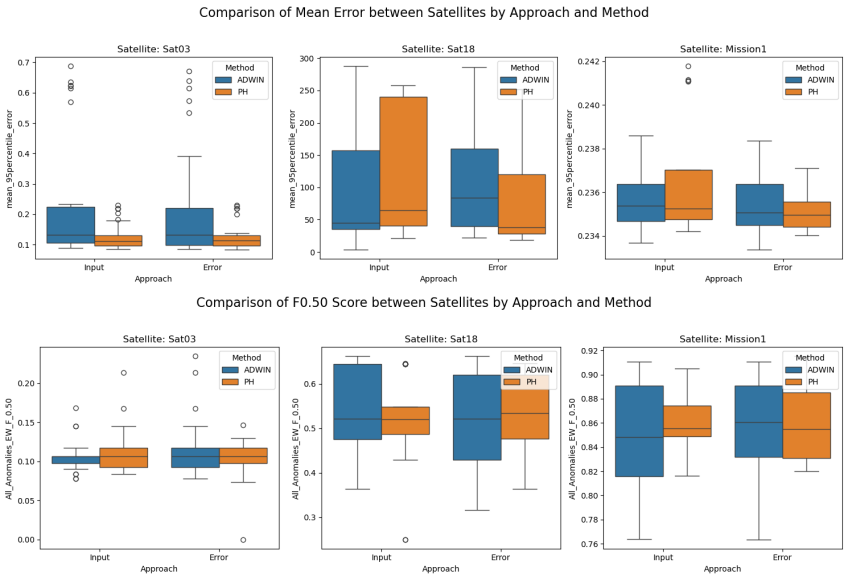

Figure 4: Performance comparison between Sat03, Sat18 and Mission1 per approach and method.

From these results several generalization patterns emerge. For forecasting (error metric), the consistency advantage of PH over ADWIN that we observed on Sat03 now only holds for the error-based approach: PH yields slightly lower median error and tighter dispersion than ADWIN when monitoring reconstruction errors. However, for the input-based approach this trend is reversed in both Sat18 and Mission1, where ADWIN slightly outperforms (or matches) PH and remains comparably stable. Notably, ADWIN appears largely insensitive to whether the trigger is input- or error-based across all three datasets. PH, in contrast, shows a clearer dependence on the monitored signal, becoming less stable (broader spread) under the input-based variant on Sat18 and Mission1.

For anomaly detection (F0.5-Score), the two methods achieve similar central tendency overall, yet their variability profiles differ: ADWIN systematically exhibits higher variance (particularly on Sat18 and Mission1), producing both the strongest and weakest configurations. This indicates a wider effective hyperparameter operating range (beneficial for potentially discovering high-performing settings, but riskier without careful selection). PH is comparatively conservative: especially under the input-based approach it produces tighter distributions, implying more predictable behavior at the cost of fewer top outliers. Again, ADWIN's performance remains largely unaffected by the monitored stream, whereas PH attains its most consistent anomaly detection results when operating on inputs rather than errors.

An important observation is that most patterns seen on Sat03 do not replicate on Sat18 and Mission1, while the latter two behave similarly to each other for both tasks. This divergence likely reflects spacecraft-specific drift and anomaly regimes: Sat03 may experience a richer mix of drift types (abrupt, seasonal, intermittent), whereas Sat18 and Mission1 may show more homogeneous, lower-amplitude or more gradually evolving shifts. Practically, this indicates that:

1. Hyperparameters selected on a single client (e.g., Sat03) do not necessarily transfer directly to others.
2. The behavior of the concept drift detection algorithms can vary significantly across different datasets, highlighting the importance of dataset-specific tuning.
3. Clustering clients by drift/anomaly dynamics (as Sat18 and Mission1 implicitly form) could enable shared tuning or meta-initialization.

This also underscores the risk of over-generalizing from a single "representative" device in Federated Continual Learning.

Despite the dataset-specific divergences, the overarching trade-off identified earlier persists: configurations that are more robust for forecasting (lower reconstruction error) often degrade anomaly detection performance, and vice versa. For example, aggressive, frequent retraining (favored by lower minimum window / instance settings) improves short-term adaptation and reduces forecast error but can shrink the residual distribution, making subtle anomalies harder to distinguish. Conversely, more conservative retraining schedules retain a sharper separation between nominal and anomalous error profiles but allow drift accumulation, increasing forecasting error. These observations might imply the need to either decouple the tasks (train a separate model for each task) or explicitly optimize both objectives (include a term related to anomaly detection into the loss function).

4.3.3 Engineering knowledge (Sat03-dt)

To evaluate the impact of incorporating external engineering knowledge, we conducted another grid search on a detrended version of the Sat03 dataset (Sat03-dt), with the same hyperparameter values. The detrending process involved removing long-term trends from the data, which essentially eliminates any concept drift, as most of the variability in the resulting data comes from noise in the sensor readings. This process allows the model to focus on the short-term data distribution and detect more subtle changes. In this report, we do not focus on the specific results of the grid search, as the main goal for this experiment was to study the impact of the detrending process itself. Therefore, we only show the comparison of final testing results using the best configurations from both the original and detrended experiments, as presented in Figure 5.

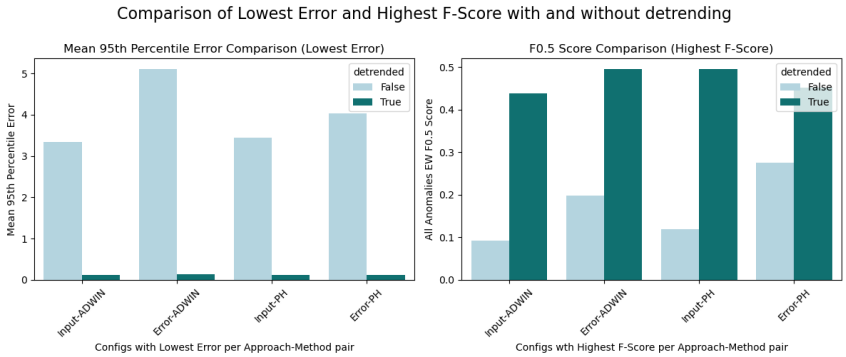

Figure 5: Comparison of best runs between Sat03 and Sat03-dt.

From these plots, it is evident that the detrending process has a significant impact on both the forecasting and anomaly detection performance, leading to substantial improvements in both metrics. This suggests that incorporating domain knowledge, such as detrending, can be highly beneficial for enhancing the model's ability to adapt to changes in the data and improve overall performance. However, it is important to note that the detrending process effectively removes the concept drift from the data, which may not always be feasible or desirable in real-world scenarios where the underlying data distribution can change over time.

With this experiment, we demonstrate the potential benefits of injecting external engineering knowledge into the data preprocessing pipeline. However, one of the main goals of this work is to develop a methodology that can adapt to changes in the data without relying on such preprocessing steps, as they may not always be applicable or effective. Therefore, our primary focus remains on developing robust federated and continual learning strategies that can handle concept drift directly from the raw data, which is the reason why the rest of the experiments will be conducted on the original, non-detrended datasets.

4.4 Federated Continual Learning Experiments

In our main set of experiments, we compared the performance of three different FCL strategies:

1. **Baseline:** A classic Federated Learning model without any continual learning. This was used to test the initial pre-trained model.
2. **Synchronous Periodic Retraining:** An FCL model with synchronous, periodic updates across all clients, with a retraining interval of 16 weeks.
3. **Asynchronous Drift-Triggered Retraining:** An FCL model with asynchronous updates triggered by a concept drift detection mechanism on each client. For this strategy, we tested both input-based and error-based approaches with ADWIN and the PH-test.

The experimental setup consisted of five satellites from the Constellation dataset as clients: Sat02, Sat03, Sat05, Sat15, and Sat21. The initial models were pre-trained using 16 weeks of data in a federated setting. For the asynchronous configurations, we selected the best-performing hyperparameter configurations from the baseline (Sat03) grid search experiments. Specifically, for each approach-method pair (e.g., input-based ADWIN), we

chose two configurations: the one that yielded the lowest reconstruction error and the one that resulted in the highest F0.5-Score.

For each of these strategies, we also evaluated the impact of using the global model for testing versus allowing for client-side finetuning. The performance of the models was evaluated on both the forecasting and anomaly detection tasks. For the forecasting task, we report the average Mean Error across clients, while for the anomaly detection task, we calculate the Global F0.5-Score by aggregating the true positives, false positives, and false negatives across all clients before computing the score.

We first summarize the comparative performance of the three strategies in Figure 6, which aggregates results across clients for forecasting (Mean Error; lower is better) and anomaly detection (Global F0.5-Score; higher is better).

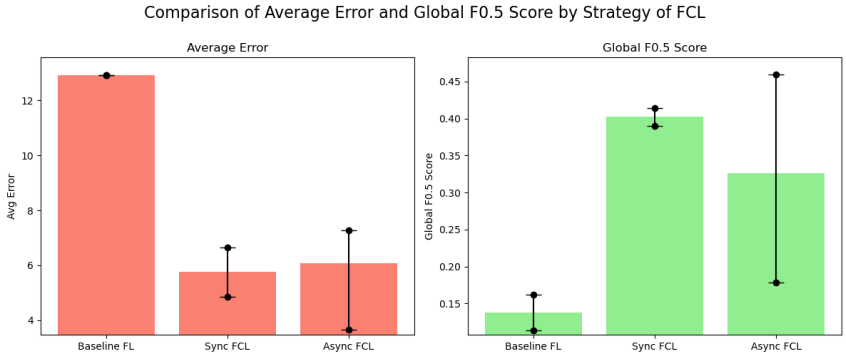

Figure 6: Comparison of FCL strategies.

The two Federated Continual Learning (FCL) strategies (synchronous periodic and asynchronous drift-triggered) both yield substantial gains over the Baseline FL model for forecasting (lower Mean Error) and anomaly detection (higher F0.5), confirming that continual adaptation is critical in this federated PHM setting. Between the two FCL variants, Synchronous FCL attains slightly better central (median) performance and tighter dispersion, whereas Asynchronous FCL exhibits noticeably higher variance yet also contains the best individual runs in both tasks. This pattern highlights the importance of per-client tuning, due to the high sensitivity of the drift detection mechanism to the chosen hyperparameters, making it a possible advantage or hindrance depending on the configuration.

Another possible explanation is that the asynchronous paradigm introduces additional sources of stochasticity, which could be due to:

- Staleness of local models at aggregation time (the FedAsync weighting can under- or over-compensate for delayed updates) [43].
- Heterogeneous and unaligned drift-trigger times producing unbalanced contribution across clients.
- Potential amplification of client-specific noise when some clients retrain more often than others.

The fact that only a single asynchronous aggregation rule (FedAsync) was evaluated further suggests room for stabilization via alternative schemes in future work; e.g. staleness-aware dampening with adaptive learning rates [45], buffer-based batching of asynchronous updates (FedBuff) [46], variance reduction corrections (SCAFFOLD-style control variates) [47], normalization of update magnitudes (FedNova) [48], or drift-aware weighting that discounts

updates lacking sufficient evidence of distributional change [49]. Exploring such strategies could retain peak performance while narrowing variability.

We next assess the impact of client-side finetuning of the received global model prior to evaluation. Figure 7 compares evaluation with and without finetuning for forecasting and anomaly detection.

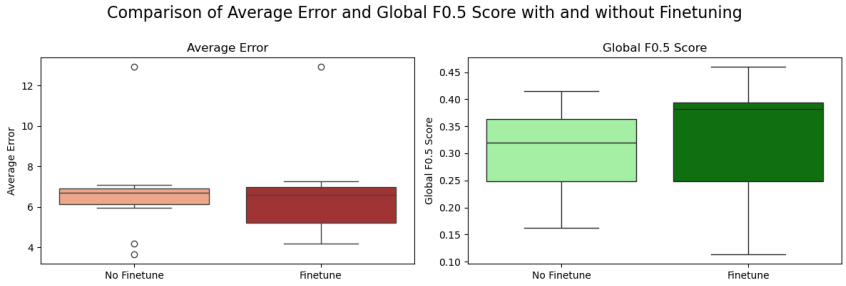

Figure 7: Impact of finetuning for FCL.

Finetuning (personalized adaptation of the received global model before evaluation) shifts the distributions favorably for both tasks: error distributions move toward lower values and the F0.5-Score distributions toward higher values, with a markedly higher median F0.5. The lower bounds (top quartile of errors / bottom quartile of F0.5) remain comparable to the non-finetuned case, indicating no systematic degradation. The broader spread observed with finetuning reflects increased variance introduced by client-specific adaptation: personalization can exploit local specificities for improved reconstruction and anomaly separation, but also risks overfitting to recent local windows (especially when anomalies are sparse or drift events cluster). That variance, however, is asymmetric (yielding higher ceilings without pulling down the floor) suggesting that moderate personalization is net beneficial in this federated continual learning regime. Conceptually, finetuning improves local calibration (reducing bias introduced by global averaging across heterogeneous clients) while the retained global pre-training acts as a regularizer that prevents catastrophic divergence; thus we observe upside potential with limited downside. Future refinements could include elastic regularization toward the global weights [50], or selective layer-wise adaptation [51] to further control variance while preserving the demonstrated gains.

5 Sustainability Analysis

This section provides an analysis of the sustainability aspects of the proposed methodology following the guidelines outlined in the Guía Informe Sostenibilitat TFG-TFM from UPC.

6 Conclusion

This work presented a Concept Drift-aware Federated Continual Learning (FCL) framework for spacecraft Prognostics and Health Management (PHM). Our approach combines an LSTM-based Telemanom forecaster with federated training and client-side drift detection to trigger model updates asynchronously. We evaluated the framework on a confidential constellation dataset—where each spacecraft acts as a natural client—and used the public ESA-ADB dataset solely to tune drift-detection hyperparameters.

Empirically, we found that drift-triggered updates improve forecasting performance compared to both a static baseline and a periodically retrained model when a comparable number of retrainings is performed. In particular, as the number of retrainings increases, the trimmed mean absolute error consistently decreases; several Concept Drift-Aware (CDA) configurations match or surpass the forecasting performance of periodic continual learning with fewer or similar update counts. Across detectors, Page–Hinkley tended to yield more stable outcomes than ADWIN, while input- and error-based monitoring produced similar trends.

For anomaly detection, results were mixed: many CDA configurations underperformed the baseline and periodic retraining in terms of F0.5, highlighting a misalignment between optimizing forecasting loss and maximizing anomaly-detection precision under our rolling-window thresholding scheme. This suggests that improving anomaly detection may require objectives and update triggers that better reflect detection-specific goals (e.g., adaptive thresholds or detection-aware losses) rather than solely pursuing lower forecasting error.

Operationally, the event-driven, client-side triggering provides responsiveness to local distribution shifts and can avoid unnecessary global rounds, supporting resource efficiency in federated spacecraft fleets. We also examined using locally fine-tuned models for inference versus the aggregated global model; the relative benefits were satellite-dependent, indicating a trade-off between specialization and cross-satellite generalization.

This study has limitations. Results are centered on one forecasting backbone (Telemanom) and a threshold-based detection strategy; other architectures, objectives, and detectors may shift the balance between forecasting and detection performance. Moreover, while we tuned drift-detection hyperparameters on ESA-ADB and transferred them to the constellation setting, broader validations on public multi-client datasets would strengthen generality. Finally, we did not fully characterize compute/communication cost under strict on-board constraints.

In future work, it could be beneficial to pursue:

1. Detection-aware training objectives to better align forecasting and anomaly detection.
2. Broader drift detectors (e.g., ensemble or multivariate tests) and principled, on-line hyperparameter adaptation.
3. Drift- and staleness-aware asynchronous aggregation and client selection policies to balance freshness, fairness, and efficiency.
4. Robustness to client heterogeneity via personalization layers or meta-learning.
5. Evaluations on additional public datasets and hardware-in-the-loop studies reflecting on-board compute and communication limits.

Overall, our results indicate that concept drift-triggered, asynchronous updates are a practical and effective mechanism for maintaining forecasting performance in federated PHM for spacecraft constellations, while also revealing the need for detection-aware objectives to translate those gains into consistent anomaly-detection improvements.

References

[1] C. S. Kulkarni, F. Figueroa *et al.*, "Prognostics for systems health management – model and hybrid based approaches," NASA, Tech. Rep. NASA/TM–2020–5007187, 2020. [Online]. Available: https://ntrs.nasa.gov/citations/20205007187

[2] Z. Cao, J. Zhang, Z. Li, X. Zhao, D. Guo, and H. Fu, "Study on prognostics and health management of fluid loop system for space application," *Journal of Physics: Conference Series*, vol. 2184, no. 1, p. 012059, 2022.

[3] T. Le, E. Chaumette, N. Larrieu, and G. Celeux, "Prognostics and health management for satellite systems: A literature review and a case study," *IEEE Systems Journal*, vol. 16, no. 3, pp. 4879–4890, 2022.

[4] S. K. Jagatheesaperumal *et al.*, "Enabling trustworthy federated learning in industrial iot: Bridging the gap between interpretability and robustness," in *arXiv preprint*, 2024. [Online]. Available: https://arxiv.org/abs/2409.02127

[5] N. Günnemann-Gholizadeh, "Machine learning methods for detecting rare events in temporal data," Ph.D. dissertation, Technische Universität München, 2018, doctoral dissertation. [Online]. Available: https://mediatum.ub.tum.de/doc/1444158/document.pdf

[6] J. Gama and Žliobaitė, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[7] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[8] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization:distributed optimization beyond the datacenter," 2015. [Online]. Available: https://arxiv.org/abs/1511.03575

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[10] L. Li, Y. Fan, M. Tse, and K. Lin, "Federated learning for industrial internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2031–263, 2020.

[11] B. Shubyn *et al.*, "Federated learning for anomaly detection in industrial iot-enabled production environment supported by autonomous guided vehicles," in *Proceedings of the International Conference on Computational Science (ICCS)*, 2022. [Online]. Available: https://www.iccs-meeting.org/archive/iccs2022/papers/133530391.pdf

[12] J. Ahn *et al.*, "Federated learning for predictive maintenance and anomaly detection using time series data distribution shifts in manufacturing processes," *Sensors*, vol. 23, no. 17, p. 7331, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/17/7331

[13] A. Gkillas and A. Lalos, "Towards resource-efficient federated learning in industrial iot for multivariate time series analysis," *arXiv preprint*, 2024. [Online]. Available: https://arxiv.org/abs/2411.03996

[14] B. Farahani *et al.*, "Smart and collaborative industrial iot: A federated learning perspective," *Computer Networks / Journal (reviewed work)*, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864823000354

[15] C. B. Mawuli, L. Che, J. Kumar, S. U. Din, Z. Qin, Q. Yang, and J. Shao, "Fedstream: Prototype-based federated learning on distributed concept-drifting data streams," pp. 7112–7124, 2023.

[16] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[17] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: https://www.pnas.org/content/114/13/3521

[18] F. Husz'ar, "Note on the quadratic penalties in elastic weight consolidation," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 115, no. 11, pp. E2496–E2497, 2018.

[19] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," *arXiv preprint (NeurIPS 2019 workshop / NeurIPS accepted work)*, 2019. [Online]. Available: https://arxiv.org/abs/1811.11682

[20] X. Li, B. Tang, and H. Li, "Adaer: An adaptive experience replay approach for continual lifelong learning," *Neurocomputing*, vol. 572, p. 127204, 2024.

[21] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018_workshops/w25/html/Mallya_PackNet_Adding_Multiple_CVPR_2018_paper.html

[22] T. Kang, "Wireless sensor network in environment monitoring," in *Proceedings of the 2021 International Conference on Social Development and Media Communication (SDMC 2021)*. Atlantis Press, 2022, pp. 1435–1439.

[23] S.-W. Lee, J.-H. Kim, J. Jun, J.-W. Ha, and B.-T. Zhang, "Overcoming catastrophic forgetting by incremental moment matching," 2018. [Online]. Available: https://arxiv.org/abs/1703.08475

[24] Q. Besnard and N. Ragot, "Continual learning for time series forecasting: A first survey," *Engineering Proceedings*, vol. 68, no. 1, 2024. [Online]. Available: https://www.mdpi.com/2673-4591/68/1/49

[25] N. Kuhn, B. Sánchez Gómez, N. Blasco Moreno, and F. Antonello, "Validation-gated continual learning for time-series anomaly detection," 2025.

[26] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[27] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[28] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection (ddm)," in *Advances in Artificial Intelligence — SBIA 2004 (Lecture Notes in Computer Science)*, vol. 3171, 2004, pp. 286–295. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-28645-5_29

[29] M. Baena-García, J. Campo-Ávila, R. Fidalgo-Merino, A. Bifet, R. Gavald, and R. Morales-Bueno, "Early drift detection method," 01 2006.

[30] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, vol. 416, p. 340–351, Nov. 2020. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2019.11.111

[31] I. Frías-Blanco, J. del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2015.

[32] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift — a survey on monitoring in evolving environments (part a: unsupervised drift detection)," *Frontiers in Artificial Intelligence*, 2024.

[33] M. T. Martinez and P. L. D. Leon, "Unsupervised segmentation and labeling for smartphone acquired gait data," 2021.

[34] D. Jayaratne, "Continuous detection of concept drift in industrial cyber-physical systems using closed-loop incremental machine learning," *Discover Artificial Intelligence / Springer (journal / collection)*, 2021.

[35] J. Zenisek, G. Kronberger, J. Wolfartsberger *et al.*, "Concept drift detection with variable interaction networks," *arXiv preprint / EuroCAST proceedings (related)*, 2021.

[36] R. Casado, A. Garcia, N. Vallez, and P. Castells, "Federated learning for non-iid data: A survey," *ACM Computing Surveys*, 2022.

[37] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," *arXiv preprint arXiv:1904.07734*, 2019. [Online]. Available: https://arxiv.org/abs/1904.07734

[38] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "Adaptive learning from stream under concept drift," *Intelligent Data Analysis*, vol. 20, no. 6, pp. 1249–1253, 2016.

[39] J. Yoon, E.-S. Lee, J. Jeong, and S.-J. Hwang, "Federated continual learning with weighted averaging and knowledge distillation," *arXiv preprint arXiv:2104.07353*, 2021.

[40] F. E. Casado, D. Lema, M. F. Criado, R. Iglesias, C. V. Regueiro, and S. Barro, "Concept drift detection and adaptation for federated and continual learning," *Multimedia Tools and Applications*, vol. 81, no. 3, pp. 3397–3419, 2022. [Online]. Available: https://doi.org/10.1007/s11042-021-11219-x

[41] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Telemanom: A deep learning-based anomaly detector for spacecraft telemetry," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1459–1467.

[42] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "Continual learning with experience replay: A scaled-up study on cl benchmarks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[43] C. Xie, O. Koyejo, and I. Gupta, "Asynchronous federated optimization," in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019, pp. 9606–9617.

[44] M. E. A. Sehili and Z. Zhang, "Multivariate time series anomaly detection: Fancy algorithms and flawed evaluation methodology," 2023. [Online]. Available: https://arxiv.org/abs/2308.13068

[45] D. Qiao, S. Guo, J. Zhao, J. Le, P. Zhou, M. Li, and X. Chen, "Asmafl: Adaptive staleness-aware momentum asynchronous federated learning in edge computing," *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 3390–3406, 2025.

[46] T. Nguyen, R. Pathak, W.-M. Hwu, and H.-T. Lin, "Fedbuff: Buffering for accelerated asynchronous federated learning," 2022, arXiv preprint arXiv:2206.09374.

[47] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi, "Scaffold: Stochastic controlled averaging for on-device federated learning," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[48] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "Fednova: Normalizing client updates for efficient federated optimization," 2020, arXiv preprint arXiv:2007.07481.

[49] H. He, Q. Zhang, K. Yi, K. Shi, Z. Niu, and L. Cao, "Distributional drift adaptation with temporal conditional variational autoencoder for multivariate time series forecasting," 2024. [Online]. Available: https://arxiv.org/abs/2209.00654

[50] D. Chen, J. Hu, V. J. Tan, X. Wei, and E. Wu, "Elastic aggregation for federated optimization," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 12 187–12 197.

[51] J. Park, J. Kim, H. Kwon, I. Yoon, and K. Sohn, "Layer-wise auto-weighting for non-stationary test-time adaptation," 2023. [Online]. Available: https://arxiv.org/abs/2311.05858

A Hyperparameter Grids

This appendix details the hyperparameter grids explored for the drift detection methods, ADWIN and Page-Hinkley (PH), across the Constellation and Mission1 datasets. For each method, we performed a grid search on two key hyperparameters for both the input and error-based detection approaches, while other parameters were held at fixed values.

A.1 ADWIN (Adaptive Windowing)

ADWIN is a change detection algorithm that maintains a variable-length window of recent data to identify changes in the data's distribution. The following hyperparameters were part of the grid search:

- **delta**: This parameter represents a confidence value. It is used to determine if two sub-windows of data are statistically different enough to declare a change point.
- **min_window_length**: This sets the minimum required size for a sub-window. Windows smaller than this are not considered for drift evaluation, which can help prevent false positives at the expense of detection delay.

The hyperparameters with fixed values were `clock` (the frequency of checking for drift), set to 12 hours; `max_buckets` (controlling the granularity of the internal data structure), set to 100; and `grace_period` (the initial number of samples to observe before starting detection), set to 1 week. The values for `delta` and `min_window_length` explored in the grid search are summarized in Table 3.

Dataset	Hyperparameter	Input Approach	Error Approach
Constellation	**delta**	10^{-5}, 10^{-10}, 10^{-25}, 10^{-50}, 10^{-100}, 10^{-200}	10, 0.1, 10^{-5}, 10^{-10}, 10^{-25}, 10^{-50}
	min_window_length	1 d, 1 w, 1 mo, 2 mo, 4 mo	1 d, 1 w, 1 mo, 2 mo, 4 mo
Mission1	**delta**	10^{-5}, 10^{-10}, 10^{-25}, 10^{-50}	10^{-5}, 10^{-10}, 10^{-25}, 10^{-50}
	min_window_length	1 w, 1 mo, 2 mo, 4 mo	1 w, 1 mo, 2 mo, 4 mo

Table 3: ADWIN Hyperparameter Search Grid

A.2 PH (Page-Hinkley)

The Page-Hinkley test is a sequential statistical method designed to detect a change in the average of a Gaussian signal. It works by monitoring the cumulative deviation of observed values from their running mean. The hyperparameters tuned were:

- **threshold**: Also known as lambda (λ), this value sets the detection threshold. A concept drift is signaled if the internally calculated Page-Hinkley statistic surpasses this threshold.

- **min_instances**: This defines the minimum number of data points that must be observed before the detector becomes active and can signal a change.

Fixed values were used for `delta` (the minimum magnitude of change to be detected), set to 0.001, and `alpha` (a forgetting factor for weighting the mean), set to 0.9. The values for `threshold` and `min_instances` explored in the grid search are summarized in Table 4.

Dataset	Hyperparameter	Input Approach	Error Approach
Constellation	**threshold**	40, 35, 30, 25, 20, 10	10, 5, 2.5, 1, 0.75, 0.5
	min_instances	1 d, 1 w, 1 mo, 2 mo, 4 mo	1 d, 1 w, 1 mo, 2 mo, 4 mo
Mission1	**threshold**	34, 35, 40, 100	3, 3.5, 5, 10
	min_instances	1 w, 1 mo, 2 mo, 4 mo	1 w, 1 mo, 2 mo, 4 mo

Table 4: Page-Hinkley Hyperparameter Search Grid

B Hyperparameter Heatmaps

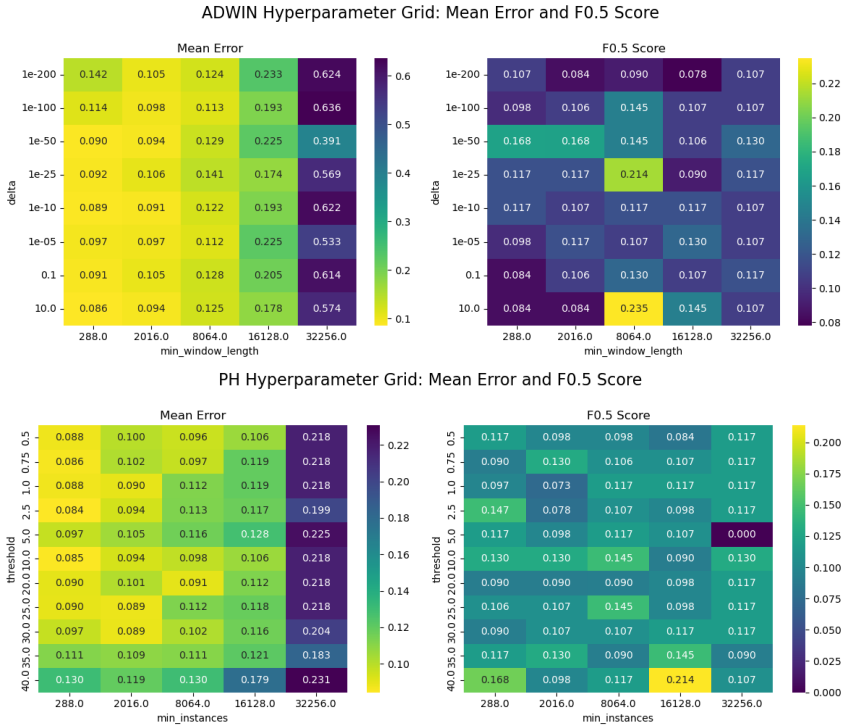

Figure 8: Hyperparameter heatmaps for ADWIN and PH on Sat03.

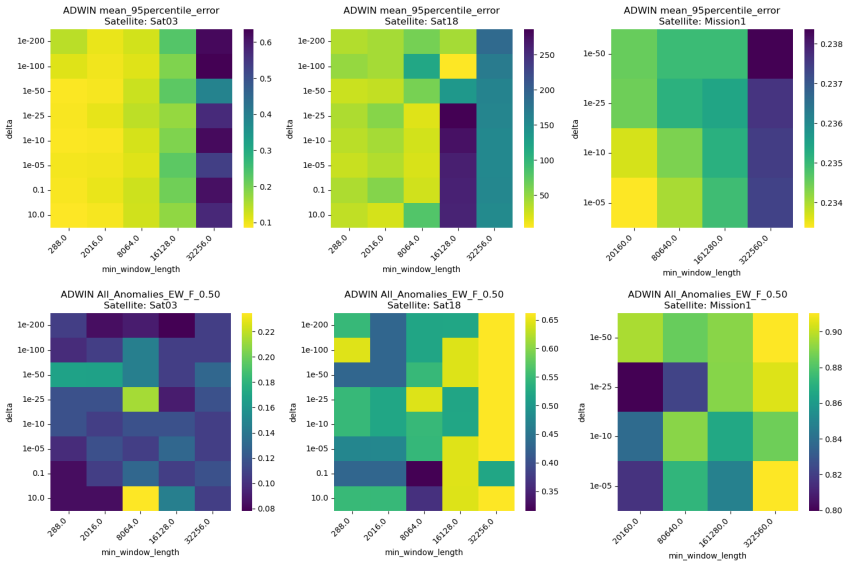

Figure 9: Comparison of ADWIN hyperparameter heatmaps between Sat03, Sat18 and Mission1.

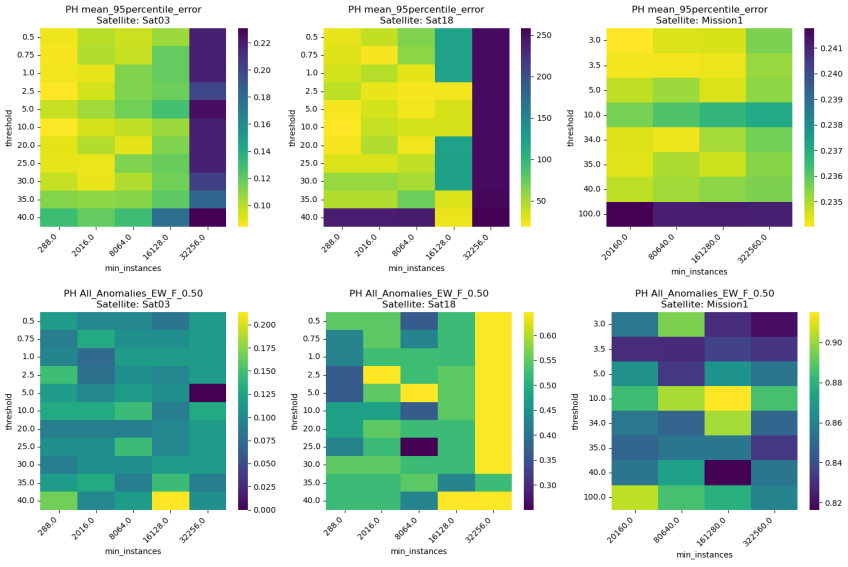

Figure 10: Comparison of Page-Hinkley hyperparameter heatmaps between Sat03, Sat18 and Mission1.