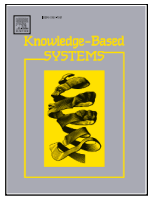
Knowledge-Based Systems 334 (2026) 115043

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



Federated LSTM autoencoders for time series anomaly detection in production-scale HPC systems

Emmen Farooq *, Michela Milano , Andrea Borghesi

DISI, University of Bologna, Bologna, Italy

ARTICLE INFO

Keywords:
Anomaly detection
Federated learning (FL)
High-performance computing (HPC)
LSTM autoencoders
Machine learning
Time series analysis

ABSTRACT

High-Performance Computing (HPC) systems are becoming increasingly vulnerable to anomalies as their scale and complexity grow. In this work, we propose a federated learning (FL) framework that integrates Long Short-Term Memory (LSTM) autoencoders for time series anomaly detection, allowing decentralized model training without sharing raw data. Using real telemetry from the Marconi100 Tier-0 supercomputer, our approach improves the average F1-score from 0.388 to 0.867 (+123 %) and the AUC from 0.334 to 0.808 (+142 %). It also cuts the training data requirement by a factor of 15, reducing the collection period from 4.5 months to just 1.25 weeks. These improvements are consistent across unsupervised, semi-supervised, and supervised settings, and significance testing with the Wilcoxon signed-rank test confirms they are statistically robust ($p < 0.01$). To our knowledge, this is the first comprehensive evaluation of FL-based LSTM autoencoders for anomaly detection in real HPC environments.

1. Introduction

High-Performance Computing (HPC) systems serve as the computational backbone for scientific and industrial breakthroughs, powering demanding applications such as physics simulations, bioinformatics, and climate modeling [5]. These infrastructures typically comprise thousands of heterogeneous compute nodes distributed across large facilities, making their management and reliability increasingly challenging as scale grows. Despite advances in instrumentation and monitoring, anomaly detection in HPC environments remains a critical and largely manual task, often reliant on domain experts to interpret vast volumes of telemetry data from system logs and hardware sensors [6].

In recent years, data-driven methods have gained traction for fault detection in HPC, leveraging machine learning (ML) and deep learning (DL) to automate anomaly identification [7]. Among these, unsupervised and semi-supervised models such as autoencoders have become particularly valuable given the scarcity of labeled fault data in production [8]. However, traditional dense autoencoders, while effective for static data, fail to capture the temporal dependencies intrinsic to time-series telemetry. Long Short-Term Memory (LSTM) autoencoders address this limitation by modeling long-range correlations, making them especially suited for detecting evolving faults in HPC workloads [9].

Federated Learning (FL) offers a promising paradigm for collaborative anomaly detection in distributed systems without requiring centralized data collection. By training models locally at each node and sharing only model updates, FL enables cross-node knowledge transfer while preserving privacy and reducing communication overhead [10]. Although FL has been widely explored in domains such as IoT, healthcare, and edge computing, its application to HPC environments has been limited [11].

Unlike other domains where privacy is the primary motivation, the benefits of FL in HPC datacenters extend beyond data confidentiality. First, FL supports failure isolation by confining training to individual nodes or groups, improving resilience against intermittent faults or resource contention [5]. Second, it provides modular scalability, allowing training to expand seamlessly across clusters or workloads [12]. Third, FL fosters node-level autonomy by tailoring models to local behavior while reducing strain on centralized resources. These advantages are particularly compelling in production-scale HPC systems, where scalability, resiliency, and responsiveness are as critical as accuracy [13].

Prior work [14] introduced the first application of FL for anomaly detection in HPC using dense autoencoders. While it demonstrated the feasibility of decentralized training, dense autoencoders lacked the ability to capture temporal dynamics in time-series telemetry. To address this, a follow-up study [15] employed LSTM autoencoders in an unsupervised FL setting, reporting improved accuracy and underscoring the importance of temporal modeling.

Building on these foundations, the present work generalizes the use of FL with LSTM autoencoders across all three major learning paradigms: unsupervised, semi-supervised, and supervised. Our

* Corresponding author.
E-mail addresses: emmen.farooq3@unibo.it (E. Farooq), michela.milano@unibo.it (M. Milano), andrea.borghesi3@unibo.it (A. Borghesi).

https://doi.org/10.1016/j.knosys.2025.115043
Received 24 May 2025; Received in revised form 29 October 2025; Accepted 1 December 2025
Available online 5 December 2025

Table 1
Comparison of FL strategies in related domains vs. this study.

Domain	Application Area	FL Strategies Used	Model Type	Evaluation Metrics	Dataset Type	Key Challenges Addressed	Reference
IoMT	Human activity recognition	FedAvg	Hybrid LSTM-GRU with CNN	F1-score, Accuracy	UCI-HAR, HARTH, HAR7 +	Privacy, Decentralization	[1]
Healthcare	Brain tumor classification	FedAvg, FedProx	CNN	Accuracy, Recall, F1-score	Brain MRI (Kaggle)	Privacy, Heterogeneity	[2]
Network security	Privacy-preserving analytics	FedAvg, FedProx, FedNova	Deep neural network	Classification accuracy	Heterogeneous data	Robustness to attacks	[3]
Cybersecurity	Intrusion detection	FedAvg, FedAdagrad, FedYogi	AE, DNN	Detection Rate, F1	KDD'99, UNSW-NB15	Imbalanced data, adversarial inputs	[4]
This Study	HPC anomaly detection	FedAvg, FedAvgM, FedProx, FedAdam, FedYogi, FedAdagrad	**LSTM Autoencoder**	**F1-score, AUC**	Real telemetry (Marconi100)	Data scarcity, temporal modeling	This paper

evaluation is conducted on telemetry from the Marconi100 Tier-0 supercomputer [16], a production-scale HPC system hosted at a leading European facility. In addition to focusing on LSTM architectures, this study complements earlier dense autoencoder investigations [14], offering a technically distinct perspective. To the best of our knowledge, this work delivers the most comprehensive empirical study of FL for HPC anomaly detection to date, benchmarking six aggregation strategies–FedAvg, FedAvgM, FedProx, FedAdagrad, FedYogi, and FedAdam.

1.1. Contributions of this paper

This work proposes the first federated learning architecture based on LSTM autoencoders for time-series anomaly detection in operational Tier-0 HPC systems. The contributions are grouped into **technical innovations** and **validated empirical improvements**, as detailed below:

Technical contributions

- **Federated temporal anomaly detection architecture:** A decentralized LSTM autoencoder pipeline that preserves node-local data, reduces monitoring overhead, and captures temporal dependencies in HPC telemetry.
- **Advanced FL optimization strategies for HPC telemetry:** Formulation and benchmarking of six federated aggregation methods (FedAvg, FedAvgM, FedProx, FedAdagrad, FedYogi, FedAdam) to address data heterogeneity, instability with small node-local datasets, and non-IID anomaly patterns.
- **Cross-node knowledge transfer under high anomaly scarcity:** A collaborative learning mechanism that improves generalization across nodes even when anomalies are extremely rare and node behaviors diverge.
- **Robust anomaly scoring with temporal sensitivity:** Integration of reconstruction-based decision thresholds with sensitivity analysis to ensure consistent detection quality across different datasets and node groups.

Empirical findings

- **Superior detection performance in real HPC operations:** Average F1-score improves from 0.388 → 0.867 (+123 %) and AUC from 0.334 → 0.808 (+142 %) on telemetry from the Marconi100 Tier-0 supercomputer.
- **Strong robustness to data scarcity:** Federated models maintain high performance even when training data is reduced by a factor of 15 (from 4.5 months to 1.25 weeks).
- **Comprehensive learning paradigm evaluation:** Consistent improvements observed across *unsupervised*, *semi-supervised*, and *supervised* training modes.
- **Statistical validation of improvements:** All performance gains are significant under the Wilcoxon signed-rank test ($p < .01$).

1.2. Structure of the paper

Section 2 reviews the literature relevant to this study. Section 3 outlines the semi-supervised, unsupervised, and supervised anomaly detection methodologies, along with the FL approaches employed in this work. Section 4.2 presents the implementation details for both anomaly detection and FL. The experimental results are discussed in Section 4.6, and the concluding remarks are provided in Section 5.

2. Related works

Anomaly detection is a critical task across diverse domains, including credit card fraud prevention [17], the energy sector [18], IT infrastructure monitoring [19], and the Internet of Things (IoT) [20]. In this study, our focus is on anomaly detection in High-Performance Computing (HPC) systems, with a particular emphasis on the role of Federated Learning (FL). The following subsections provide an overview of the most relevant literature.

2.1. Anomaly detection in HPC systems

Anomalies in HPC systems can result in job failures, incomplete computations, and accelerated hardware degradation [21]. As systems move toward the Exascale era, anomaly detection becomes increasingly challenging due to hardware heterogeneity, temporal variability, and severe class imbalance [22,23]. Traditional supervised learning approaches often struggle in these settings, primarily because labeled fault data is scarce and imbalanced [24–26]. Consequently, unsupervised and semi-supervised methods, particularly autoencoder-based approaches, have gained traction [18]. Among these, LSTM autoencoders have demonstrated superior performance compared to dense autoencoders, owing to their ability to capture temporal dependencies in sensor data [6,9,21,27]. These models have proven effective in identifying subtle and evolving anomalies in HPC telemetry.

2.2. Federated learning for anomaly detection

Federated Learning has emerged as a promising paradigm in distributed environments such as healthcare and IoT, where data privacy and localized training are essential [28]. Beyond privacy, FL has shown its potential to improve convergence and accuracy in anomaly detection systems by enabling collaborative training across decentralized devices without exposing raw data [29,30]. For example, FL has enhanced

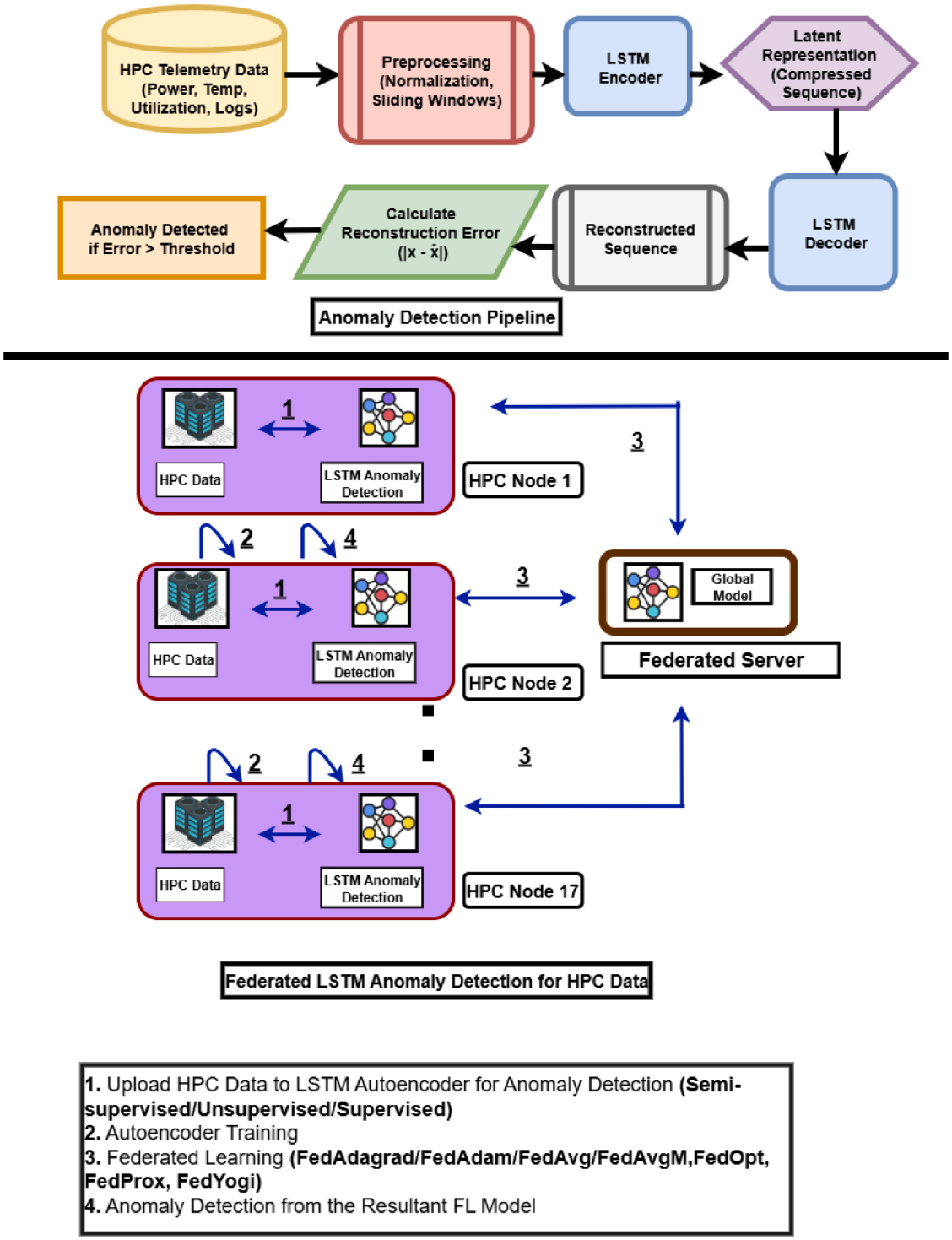


Fig. 1. Workflow of LSTM-based anomaly detection in HPC with federated learning. Top: local pipeline using LSTM autoencoder; anomalies flagged if reconstruction error exceeds threshold. Bottom: HPC nodes train local autoencoders (unsupervised/semi-supervised/supervised), send model updates to the server, which aggregates them (FedAvg, FedAvgM, FedAdam, FedAdagrad, FedOpt, FedProx, FedYogi) into a global model redistributed for refined detection.

anomaly detection in smart buildings, wireless sensor networks, and industrial IoT by leveraging models such as LSTM, GRU, and CNN-LSTM hybrids [17,31]. Moreover, recent advances in communication-efficient frameworks–such as attention-based methods and hierarchical aggregation–have further reduced the overhead of FL while preserving detection accuracy.

2.3. Advanced aggregation strategies in FL

To address the challenges of non-IID data and resource heterogeneity, several aggregation strategies have been proposed. These include FedAvgM, FedProx, FedAdagrad, FedYogi, and FedAdam [11,32]. In industrial anomaly detection tasks, algorithms such as FedAdam and FedYogi

have achieved notable improvements over the standard FedAvg baseline [14]. Beyond optimizer design, hybrid frameworks combining FL with ensemble learning and deep architectures have demonstrated resilience against distributional shifts, extending applicability to intrusion detection and other distributed security tasks [28].

While FL has been widely applied in domains such as IoT, healthcare, and cybersecurity, most studies employ dense or convolutional architectures. These approaches often overlook the temporal dynamics essential for effective time-series anomaly detection in HPC environments. Table 1 summarizes key FL strategies, architectures, evaluation metrics, and datasets used in prior work, contrasting them with the contributions of this study. In particular, our work is the first to systematically benchmark multiple FL optimizers (e.g., FedYogi, FedAdagrad, FedAvgM) in combination with LSTM autoencoders on real telemetry data from a Tier-0 supercomputer.

Comparative landscape. Although time-series anomaly detection has been studied in other domains (e.g., energy, IoT, LANL system logs), a clean apples-to-apples catalogue of SOTA methods is not available under the constraints of federated HPC telemetry. Differences in anomaly labeling, sensor semantics, and centralization assumptions limit direct comparability. For this reason, our study emphasizes principled intra-domain baselines: dense vs. LSTM autoencoders to assess temporal modeling, and local vs. federated training to isolate the effect of FL[21] in HPC operations. We highlight cross-platform benchmarking on public datasets such as LANL, Grid5000, or Google traces as a valuable future direction.

2.4. Research gap

Although FL has been validated in several domains, its application to HPC anomaly detection remains at an early stage. Initial studies using dense autoencoders in FL settings lacked the capacity to capture temporal dependencies [14], while more recent works employing LSTM-based FL approaches did not systematically compare aggregation strategies [15]. This study addresses these gaps by evaluating a comprehensive set of FL optimizers in conjunction with LSTM autoencoders, leveraging real-world telemetry from a production-scale HPC system. To the best of our knowledge, it represents the first empirically validated application of FL for time-series anomaly detection in operational Tier-0 HPC environments.

3. Methodology

This section describes the proposed methodology for applying federated learning (FL) with LSTM autoencoders to anomaly detection in high-performance computing (HPC) systems.

3.1. Overview

Fig. 1 presents a high-level view of the framework. The objective is to enhance anomaly detection accuracy across HPC nodes while reducing training time and data requirements. Although privacy is not a primary concern in this setting–since all nodes belong to the same supercomputer–FL provides architectural and performance advantages. By enabling decentralized training, FL leverages behavioral similarities across nodes without transferring raw telemetry data, thereby reducing communication overhead and improving scalability.

The framework is evaluated under three training paradigms:

- **Unsupervised:** Training on unlabeled data containing both normal and anomalous samples.
- **Semi-supervised:** Training exclusively on data from normal system behavior.
- **Supervised:** Training on fully labeled datasets with explicit anomaly annotations.

Algorithm 1: FedAvg: Federated averaging algorithm.

Server executes:
  Initialize global model w_0
  for *each round* $r = 0, 1, 2, \ldots$ **do**
    $c \leftarrow \max(C \times K, 1)$
    $S_r \leftarrow$ (random subset of c clients)
    foreach *client* $k \in S_r$ ***in parallel*** **do**
      $\Delta_r^k \leftarrow \text{ClientUpdate}(k, w_r)$
    end
    $\Delta_r \leftarrow \sum_{k \in S_r} \frac{n_k}{n} \Delta_r^k$
    $w_{r+1} \leftarrow w_r + \Delta_r$
  end
  ClientUpdate(k, w_r)**:**
    $w \leftarrow w_r$
    $\mathcal{B} \leftarrow$ (split D_k into batches of size B)
    for *each local epoch* $e = 1$ *to* E **do**
      foreach *batch* $b \in \mathcal{B}$ **do**
        $w \leftarrow w - \eta \nabla \ell(w; b)$
      end
    end
    $\Delta \leftarrow w - w_r$
return Δ to server

Algorithm 2: FedAvgM: Federated Averaging with Server Momentum

Server executes:
  Initialize global model w_0, server momentum $v_0 \leftarrow 0$
  for each round $r = 0, 1, 2, \ldots$ **do**
    Select a random subset S_r of K clients
    for each client $k \in S_r$ **in parallel do**
      $\Delta_r^k \leftarrow \texttt{ClientUpdate}(k, w_r)$
    Aggregate updates: $\Delta_r \leftarrow \sum_{k \in S_r} \frac{n_k}{n} \Delta_r^k$
    Update server momentum: $v_{r+1} \leftarrow \mu v_r + \Delta_r$
    Update global model: $w_{r+1} \leftarrow w_r + \eta_s v_{r+1}$
ClientUpdate(k, w_r):
  $w \leftarrow w_r$
  Split local data D_k into batches $\mathcal{B}$ of size B
  for each local epoch $e = 1$ to E **do**
    for each batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  Return $\Delta^k = w - w_r$

Each node independently trains a local LSTM autoencoder, while a central server aggregates models using different FL strategies. This enables collaborative training that preserves node-specific learning.

3.2. LSTM autoencoder for anomaly detection

The core detection model is an LSTM autoencoder, chosen for three main reasons: (i) the telemetry is sampled at 15-m intervals, and the sequence length of 10 steps (150 m; Table 3) is modest, favoring lightweight gated recurrent models over heavier attention-based ones; (ii) ablation studies show that replacing LSTMs with dense autoencoders reduces F1/AUC by 15-16 % (Table 10), highlighting the importance of temporal modeling; and (iii) in federated training with small node-local datasets, the parameter efficiency and stability of LSTMs make them preferable to more complex architectures.

The LSTM autoencoder is trained by minimizing the Mean Squared Error (MSE) between inputs and reconstructions, as MSE provides smoother gradients and promotes stable convergence during optimization. At inference, however, anomalies are scored using the Mean Abso-

Algorithm 3: FedProx: Federated Learning with Proximal Gradient Descent

Server executes:
  Initialize global model w_0
  for *each round* $r = 0, 1, 2, \ldots$ **do**
    $S_r \leftarrow$ (random subset of K clients)
    foreach *client* $k \in S_r$ ***in parallel*** **do**
      $\Delta_r^k \leftarrow \text{ClientUpdate}(k, w_r)$
    end
    $\Delta_r \leftarrow \sum_{k \in S_r} \frac{n_k}{n} \Delta_r^k$
    $w_{r+1} \leftarrow w_r + \eta_s \cdot \Delta_r$
  end
  ClientUpdate(k, w_r):
    $w \leftarrow w_r$
    $\mathcal{B} \leftarrow$ (split $\mathcal{D}_k$ into batches of size B)
    for *each local epoch* $e = 1$ *to* E **do**
      foreach *batch* $b \in \mathcal{B}$ **do**
        $g \leftarrow \nabla \ell(w; b) + \mu(w - w_r)$
        $w \leftarrow w - \eta \cdot g$
      end
    end
    $\Delta \leftarrow w - w_r$
    return Δ to server

lute Error (MAE) of the reconstruction, which is more robust to heavy-tailed residuals and outliers. An input is flagged as anomalous if its MAE exceeds the threshold $\theta = 0.65$, as determined through sensitivity analysis (Section 4.4).

$$\text{Error}_t = \frac{1}{n} \sum_{i=1}^{n} \left| x_{t,i} - \hat{x}_{t,i} \right|. \tag{1}$$

3.3. Federated learning setup

In the federated learning (FL) setup for HPC anomaly detection, each HPC node acts as a client. Instead of transferring raw telemetry data, clients train local models on their node-specific data and transmit only the learned updates to the central server. The server then aggregates these updates to improve the global model. The optimization problem solved by FL is:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w). \tag{2}$$

Here:

- $f(w)$: the **global objective function**, representing the overall training loss across all clients.
- w: the vector of **global model parameters** maintained by the central server.
- K: the **total number of clients** (HPC nodes) participating in FL.
- $F_k(w)$: the **local loss function** of client k, computed using its local dataset.
- n_k: the **number of training samples** available at client k.
- $n = \sum_{k=1}^{K} n_k$: the **total number of samples** across all clients.
- $\frac{n_k}{n}$: the **weighting factor** ensuring that each client's contribution is proportional to its dataset size.

In each communication round r, clients minimize their local loss $F_k(w)$ on their telemetry data and send model updates to the server. The server then aggregates these updates to update the global parameters w:

$$w_{r+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_r^k, \tag{3}$$

Algorithm 4: FedAdagrad, **FedYogi**, and **FedAdam**: Federated optimizers with variant second-moment updates.

Server executes:
  Initialize global model w_0
  Initialize decay parameters $\beta_1, \beta_2 \in [0, 1)$, $v_{-1} \geq \tau^2$
  for *each round* $r = 0, 1, 2, \ldots$ **do**
    $c \leftarrow \max(C \times K, 1)$
    $S_r \leftarrow$ (random set of c clients)
    foreach *client* $k \in S_r$ ***in parallel*** **do**
      $\Delta_r^k \leftarrow \text{ClientUpdate}(k, w_r)$
    end
    $\Delta_r \leftarrow \sum_{k \in S_r} \frac{n_k}{n} \Delta_r^k$
    $m_r \leftarrow \beta_1 m_{r-1} + (1 - \beta_1)\Delta_r$
    // **FedAdagrad**
    $v_r^{\text{Adagrad}} \leftarrow v_{r-1} + (\Delta_r)^2$
    // **FedYogi**
    $v_r^{\text{Yogi}} \leftarrow v_{r-1} - (1 - \beta_2) \cdot (\Delta_r)^2 \cdot \text{sign}(v_{r-1} - (\Delta_r)^2)$
    // **FedAdam**
    $v_r^{\text{Adam}} \leftarrow \beta_2 v_{r-1} + (1 - \beta_2) \cdot (\Delta_r)^2$
    $w_{r+1} \leftarrow w_r + \eta_s \cdot \frac{m_r}{\sqrt{v_r} + \tau}$
  end
  ClientUpdate(k, w_r):
    $w \leftarrow w_r$
    $\mathcal{B} \leftarrow$ (split local data $\mathcal{D}_k$ into batches of size B)
    for *each local epoch* $e = 1$ *to* E **do**
      foreach *batch* $b \in \mathcal{B}$ **do**
        $w \leftarrow w - \eta \nabla \ell(w; b)$
      end
    end
    $\Delta \leftarrow w - w_r$
    return Δ to server

where w_r^k denotes the local model parameters obtained by client k after local training in round r. This weighted averaging balances the contribution of clients with large and small datasets, ensuring fairness and stability in the global optimization.

We implemented six FL optimization strategies:

3.3.1. FedAvg

Federated averaging (FedAvg) In each communication round of federated learning, the central server aggregates the local models received from participating clients (HPC nodes). The contribution of each client is weighted according to the size of its local dataset, so that nodes with more training samples have a proportionally stronger influence on the global update. This aggregation rule, which underpins Algorithm 1, is expressed as:

$$w_{r+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_r^k. \tag{4}$$

where:

- r denotes the communication round index, with $r = 0, 1, 2, \ldots$.
- K is the total number of clients (HPC nodes) participating in the federated training.
- w_r^k represents the local model parameters trained by client k at round r.
- n_k is the number of training samples available on client k.
- $n = \sum_{k=1}^{K} n_k$ is the total number of samples across all clients.
- w_{r+1} is the updated global model after aggregating the local updates from round r.

3.3.2. FedAvgM

FedAvgM extends the standard FedAvg algorithm by introducing server-side momentum, which helps to stabilize convergence and improve performance under non-IID data distributions. Instead of directly averaging client updates, the server maintains a momentum term v_r that accumulates past updates in an exponentially decayed manner. This allows the global model to move more smoothly in parameter space, reducing oscillations caused by heterogeneous client data. The aggregation rule is defined as:

$$v_r = \beta v_{r-1} + \sum_{k=1}^{K} \frac{n_k}{n} \Delta w_r^k, \quad w_{r+1} = w_r - v_r. \tag{5}$$

where:

- r is the communication round index, with $r = 0, 1, 2, \ldots$.
- K is the total number of participating clients (HPC nodes).
- Δw_r^k denotes the model update computed locally by client k during round r.
- n_k is the number of training samples available on client k.
- $n = \sum_{k=1}^{K} n_k$ is the total number of samples across all clients.
- $\beta \in [0, 1)$ is the server-side momentum coefficient that controls how strongly past updates influence the current step.
- v_r is the server's momentum term at round r, combining the weighted average of current client updates with a fraction of the previous momentum v_{r-1}.
- w_r represents the global model parameters at round r.
- w_{r+1} is the updated global model obtained after applying the momentum-adjusted update v_r.

3.3.3. FedProx

FedProx extends FedAvg by introducing a proximal term into each client's local optimization objective. This term penalizes large deviations from the current global model, thereby reducing the risk of client drift and improving stability under heterogeneous (non-IID) data distributions. The local objective function for client k in communication round r is given as:

$$\min_w \; F_k(w) + \frac{\mu}{2} \|w - w_r\|^2. \tag{6}$$

where:

- $F_k(w)$: the empirical loss function of client k, computed on its local dataset.
- w: the local model parameters being optimized on client k.
- w_r: the global model parameters distributed by the server at communication round r.
- μ: the proximal coefficient that controls the strength of the penalty term.
- $\|w - w_r\|^2$: the squared Euclidean distance between the local and global model parameters, enforcing closeness to the server's model.

This proximal adjustment ensures that clients update their models in a way that balances fitting local data while staying close to the global model, thus stabilizing training in non-IID settings.

3.3.4. FedAdagrad

FedAdagrad extends federated optimization by applying the Adagrad update rule at the server side (Algorithm 4). Unlike fixed learning-rate strategies, it adapts the step size per round by accumulating squared gradients, thereby dampening updates along frequently observed directions and allowing larger progress along rare ones. This makes FedAdagrad particularly robust under non-IID data distributions in federated settings.

$$v_r = v_{r-1} + \Delta_r^2, \quad w_{r+1} = w_r - \frac{\eta}{\sqrt{v_r + \tau}} \Delta_r. \tag{7}$$

where:

- r denotes the federated round index (server iteration).
- w_r are the global model parameters at round r.
- Δ_r is the aggregated model update received from clients in round r.
- v_r is the accumulated sum of squared updates up to round r, serving as a memory of past gradient magnitudes.
- η is the base server learning rate.
- τ is a small smoothing constant that prevents division by zero and stabilizes the denominator.

3.3.5. FedYogi

FedYogi builds on FedAdagrad by modifying the accumulator update to enhance stability in non-convex optimization landscapes (Algorithm 4). Instead of directly accumulating squared gradients, FedYogi adjusts the update direction by comparing the current accumulator to the new squared gradient, thus preventing uncontrolled growth and improving robustness when anomalies or irregular gradient patterns occur.

$$v_r = v_{r-1} - (1 - \beta_2) \cdot \text{sign}(v_{r-1} - \Delta_r^2) \cdot \Delta_r^2. \tag{8}$$

Here:

- v_r is the adaptive accumulator at round r (server-side state).
- v_{r-1} is the accumulator from the previous round.
- $\beta_2 \in (0, 1)$ is the exponential decay parameter controlling how strongly past information is retained.
- Δ_r is the aggregated gradient update from clients at round r.
- $\text{sign}(\cdot)$ ensures the update direction accounts for the difference between the old and new squared gradients.

This formulation stabilizes the update process by slowing down rapid changes in the accumulator, which is particularly beneficial in heterogeneous or non-convex settings often found in federated learning.

3.3.6. FedAdam

FedAdam adapts the Adam optimizer for federated learning (Algorithm 4), leveraging both first- and second-moment estimates of the aggregated client updates. This allows the server to perform adaptive learning rate updates while smoothing out gradient noise, which is especially useful in heterogeneous and non-IID federated settings.

$$m_r = \beta_1 m_{r-1} + (1 - \beta_1)\Delta_r, \tag{9}$$

$$v_r = \beta_2 v_{r-1} + (1 - \beta_2)\Delta_r^2, \tag{10}$$

$$w_{r+1} = w_r - \eta \cdot \frac{m_r}{\sqrt{v_r} + \tau}. \tag{11}$$

Here:

- m_r – first moment estimate (exponentially decayed moving average of gradients) at round r.
- v_r – second moment estimate (exponentially decayed moving average of squared gradients) at round r.
- Δ_r – aggregated gradient update from clients at round r.
- $\beta_1, \beta_2 \in (0, 1)$ – exponential decay rates controlling the momentum of first and second moments.
- η – global learning rate at the server.
- τ – small constant added for numerical stability in the denominator.
- w_r – global model parameters at round r.

While powerful, FedAdam can be sensitive to imbalanced client data and requires careful hyperparameter tuning to avoid instability.

3.4. Training modes

To assess the framework under varying supervision, we define three training modes:

3.4.1. Unsupervised

Models are trained on unlabeled telemetry data, reconstructing inputs and detecting anomalies via reconstruction error.

Table 2
Average number of normal, anomalous, total data points and average number of anomalies in Data Set 1 (D1) and Data Set 2 (D2).

Data Set	#Normal	#Anomalous	Total	% Anom
D1	12650.25	391.25	13041.49	3.00
D2	14200.75	241.41	14442.16	1.7

3.4.2. Semi-supervised

Models are trained only on normal behavior data, with anomalies flagged as deviations during inference. This is suitable when labeled anomalies are scarce.

3.4.3. Supervised

In the supervised regime we maintain the same LSTM-AE architecture and reconstruction objective as in the semi-supervised setting, where training is performed exclusively on sequences of normal behavior. Labels are used only to calibrate the decision threshold on a held-out validation set and to enable early stopping, ensuring that the model does not overfit to the training data. Importantly, no classification head is introduced–the autoencoder continues to operate as a reconstruction model, with anomalies flagged when the reconstruction error exceeds the calibrated threshold.

3.5. FL vs. non-FL comparison

To highlight the benefits of FL, we compare:

- **Local AE baseline:** Independent training on each node without collaboration.
- **FL AEs:** Federated training using one of the strategies described above.

Rationale for baselines. We emphasize that the "Local AE Baseline" was selected as the most operationally relevant comparator for FL in production-scale HPC systems. In this setting, each node independently trains its LSTM autoencoder, representing the common practice of node-level monitoring without cross-node collaboration. By contrast, a hypothetical centralized model would require sustained large-scale data movement and storage, which is atypical and undesirable in HPC telemetry pipelines. Furthermore, a centralized approach would not assess robustness under non-IID node behavior–the very challenge that FL explicitly targets.

Finally, our results already show that advanced FL optimizers such as FedAdagrad and FedAvgM approach near-saturation performance across all regimes (Tables 8–9), which makes a centralized proxy less informative in this context.

For clarity, the baseline results reported in Tables 8–9 under the label "ML" correspond to the Local AE baseline: each node independently trains its own LSTM autoencoder without federation, and group-level values are obtained by averaging across the 17 nodes in each cohort.

All experiments use identical hyperparameters, and we further analyze performance under reduced training data fractions (1/2, 1/4, 1/8, 1/16).

4. Experimental analysis

This section presents the experimental framework adopted in this study. We begin with a detailed description of the two datasets used, followed by the technical architecture of the anomaly detection model, based on an LSTM autoencoder. Finally, we outline the implementation of the Federated Learning (FL) framework.

The experiments were conducted on a system equipped with 42 Intel(R) Xeon(R) Gold 6240R CPUs, providing a total of 96 cores and 790 GB of RAM. The system runs on Ubuntu 22.0.

The primary focus of this work is anomaly detection at the compute node level, rather than system-wide failures or issues tied to user workloads. Accordingly, the analysis is restricted to data collected from hardware-level sensors within the monitoring infrastructure. These include measurements such as power consumption, core temperature, and clock speed of computing units, while workload-related information is excluded. Both datasets used in this study consist of such hardware telemetry, combined with node status information provided by system administrators.

4.1. Datasets and data justification

To conduct a comprehensive evaluation of the proposed federated anomaly detection framework, we selected two real-world HPC system datasets. These datasets capture a broad spectrum of operational patterns, fault scenarios, and node-level behaviors, making them particularly suitable for benchmarking unsupervised, semi-supervised, and supervised anomaly detection methods. The choice was driven by three key requirements: (i) sufficient temporal granularity, (ii) availability of anomaly annotations, and (iii) relevance to production-scale HPC environments. The characteristics of each dataset are described in detail below, together with the justification for their use in this study.

4.1.1. Dataset 1 (D1)

The first dataset used in this study was collected from the Marconi100,[1] a Tier-0 supercomputer hosted at the CINECA facility in Bologna. Marconi100 consists of 980 compute nodes, each equipped with two IBM POWER9 AC922 CPUs (16 cores per CPU, 3.1 GHz), 256 GB of RAM, and four NVIDIA Volta V100 GPUs. The system delivers a peak performance of 32 PFlop/s. An advanced monitoring infrastructure, Examon, is integrated into the machine, enabling real-time collection of a wide range of operational and performance metrics. Examon has been widely adopted across CINECA's HPC systems [8,33]. Marconi100 was selected for this study due to both its significance and the availability of detailed telemetry data.

The dataset comprises 462 metrics per node, including low-level sensor data such as fan speed, clock frequency, and power consumption of CPUs and GPUs. It also incorporates higher-level information such as system availability indicators, job scheduler logs, and node status reports.

To assign anomaly labels, the Nagios[2] monitoring system was used. Nagios provides health status information based on periodic checks and user-reported issues, allowing administrators to classify nodes as either healthy or anomalous. These status updates occur every 15 m, which we adopted as the temporal resolution of this study. This sampling frequency is consistent with prior work in HPC anomaly detection [5,9,34]. To capture temporal dynamics, statistical features–the mean and standard deviation–were computed for each 15-m window. Labels were then assigned in binary format: 1 for healthy nodes and 0 for anomalous conditions.

4.1.2. Dataset 2 (D2)

The second dataset (D2) was also collected from the Marconi100[3] supercomputer at CINECA, using the same set of 980 compute nodes as in Dataset 1 (D1).

During the D2 collection period, the monitoring infrastructure underwent a substantial upgrade. Specifically, the Examon system was enhanced to provide a more comprehensive and fine-grained representation of system behavior. This enriched dataset has since been released to the research community and industry practitioners [35], serving as a valuable resource for experimental evaluation in this work.

[1] https://www.hpc.cineca.it/hardware/marconi100
[2] https://www.nagios.org/
[3] https://www.hpc.cineca.it/hardware/marconi100

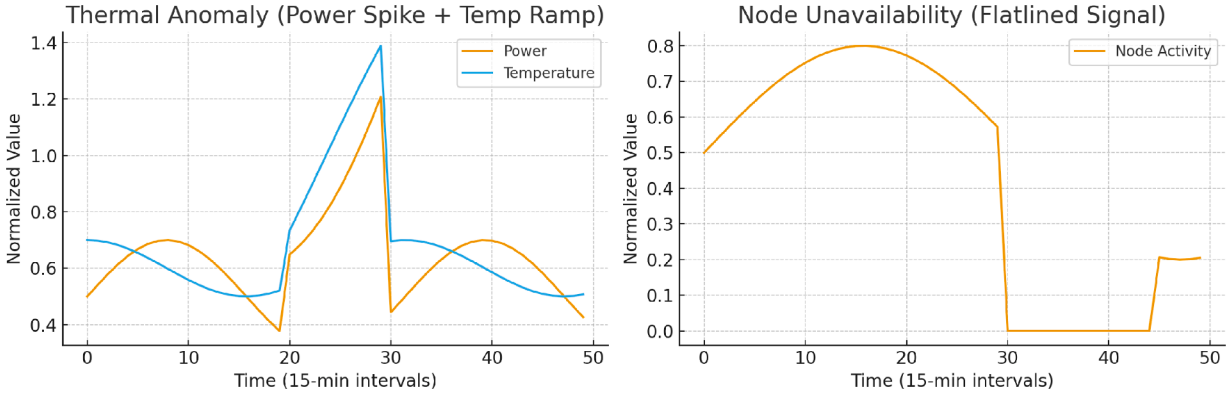


Fig. 2. Representative anonymized anomaly patterns. **Left:** Thermal anomaly characterized by power spikes co-occurring with temperature ramps. **Right:** Node unavailability illustrated by prolonged flatlined activity. Both examples follow the 15-m sampling cadence used in our datasets.

To improve system observability, additional software probes and physical sensors were deployed, increasing the number of recorded features per node from 462 in D1 to 573 in D2. The anomaly labeling methodology was also refined: system administrators curated the labels to emphasize operationally significant anomalies, reducing the anomaly rate from 3 % in D1 to 1.7 % in D2 (see Table 2). Notably, conditions previously treated as anomalies in D1–such as transient behavior due to configuration updates or operating system patches–were reclassified as normal in D2, reflecting a more accurate characterization of system dynamics.

This revised labeling strategy defines a distinct experimental scenario in D2, characterized by a broader feature set and a different anomaly profile. Although D1 and D2 share partial feature overlap, they were collected during separate operational periods of Marconi100, with differing monitoring configurations and methodologies. These differences make D2 a more challenging and realistic test case for assessing the effectiveness of Federated Learning (FL) in time-series-based anomaly detection.

For experimental evaluation, nodes were partitioned into five independent cohorts (Groups A-E), each comprising 17 nodes. This grouping strategy ensured stable client counts for federated training, while enabling repeated measurements across heterogeneous subsets of nodes for statistical significance testing. The same node groups were consistently used across all experiments, including both the local AE baselines and the FL-based models.

Operational anomalies flagged by Nagios include: (i) node-level unavailability events (e.g., heartbeat loss), (ii) thermal and power excursions leading to CPU/GPU throttling, and (iii) persistent sensor out-of-range conditions.

Because raw traces can contain sensitive operational details, we illustrate representative anonymized schematic patterns rather than node-identifying plots. For example, thermal anomalies often appear as short bursts of power spikes co-occurring with temperature ramps, while unavailability events manifest as prolonged flatlined activity in the monitored signals. These examples reflect the same 15-m sampling cadence used in our modeling pipeline Fig. 2.

Anomaly prevalence in each dataset remains low (3 % in D1 and 1.7 % in D2, see Table 2), highlighting the severe class imbalance characteristic of real HPC production systems.

4.1.3. Rationale for using Marconi100 data exclusively

Although other publicly available HPC datasets exist (e.g., LANL [36], Grid5000 [37], and Google Cluster traces [38]), we deliberately restricted our experiments to data from Marconi100. This decision was made to ensure consistency in both infrastructure and data semantics. Marconi100, a Tier-0 production supercomputer at CINECA, provides a unique opportunity to validate federated anomaly detection under real-world conditions, where organically occurring faults are observed instead of synthetic anomalies. In addition, the architectural homogeneity of its compute nodes aligns with our goal of isolating the effects of federated learning without the confounding influence of heterogeneous hardware or software configurations. While cross-platform generalization remains an important direction for future work, we argue that this focused evaluation offers a rigorous and replicable foundation for demonstrating the benefits of FL in operational HPC environments.

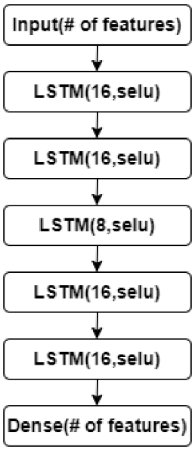


Fig. 3. Architecture of LSTM autoencoder.

It is important to highlight that the anomalies studied here are not artificially generated or manually injected. Instead, they correspond to real faults and irregularities observed in the production system and verified by system administrators.

4.2. Implementation

To evaluate the effectiveness of the proposed Federated Learning (FL) framework, we employ a Long Short-Term Memory (LSTM) autoencoder as the core anomaly detection model. Each selected HPC node is assigned a dedicated autoencoder, enabling node-specific behavior modeling. The framework is implemented across three learning paradigms–supervised, unsupervised, and semi-supervised–to ensure a comprehensive evaluation.

In the *semi-supervised* setting, models are trained exclusively on data from normal system operation, allowing the autoencoder to learn the structure of non-anomalous behavior in line with [25]. The *unsupervised* configuration uses a mixture of normal and anomalous samples without label guidance [12]. By contrast, the *supervised* approach leverages fully labeled datasets to explicitly discriminate between anomalous and non-anomalous samples [14].

Table 3
Hyperparameters of the LSTM Autoencoder. The model is trained by minimizing reconstruction error with **Mean Squared Error (MSE)** for stable convergence. At inference, anomalies are scored using **Mean Absolute Error (MAE)**, which is more robust to heavy-tailed residuals.

Hyperparameter	Value	Description
Time Steps	10	Number of time intervals used in each input sequence.
Optimizer	RMSprop	Gradient-based optimizer adapting learning rates.
Learning Rate	0.000001	Controls weight updates; smaller value ensures stability.
Loss	MSE (training)	Objective function for reconstruction learning.
Batch Size	128	Samples processed before updating weights.
Epochs	5	Complete passes through the training dataset.
Threshold	0.65	MAE-based anomaly detection threshold at inference.

4.2.1. Data preprocessing

Prior to training, data is preprocessed to exclude workload-related and non-numeric features. Node status information collected via the Nagios monitoring system is removed from training but retained for evaluation. Instances with missing values are discarded, and all remaining numerical features are normalized to the $[0, 1]$ range using Scikit-learn.

Each node's dataset was divided into 80 % training and 20 % testing, applied independently per node to maintain temporal consistency. This procedure preserves the natural anomaly prevalence within the test sets, which closely follows the dataset-level base rates: approximately 3 % in D1 and 1.7 % in D2 (see Table 2). Thus, the evaluation reflects realistic anomaly occurrence frequencies rather than artificially balanced splits. Given the sequential nature of LSTMs, the input data is reshaped into `(samples, timesteps, features)` format. Following established practices [12], we set the sequence length to 10 time steps, corresponding to a 150-m temporal window at the 15-m logging interval.

Handling high-dimensional features Both datasets contain a large number of sensor-level metrics (462 features in D1 and 573 in D2 following a monitoring system upgrade), which increases the potential for noise and redundancy. To address this, we drop all non-numeric attributes and normalize the numerical features to $[0, 1]$. Furthermore, the bottleneck of the LSTM autoencoder naturally performs dimensionality reduction by compressing input sequences into compact latent representations that preserve temporal dynamics while filtering out irrelevant variance. The robustness of this approach is demonstrated by the strong detection performance observed in D2, despite its additional 111 features, across all learning paradigms (Tables 5–7). Nonetheless, explicit feature selection techniques–such as sparsity-inducing regularization or statistical feature ranking–may further enhance efficiency, and we highlight this as an avenue for future research.

4.2.2. Model architecture and training

The LSTM autoencoder architecture is illustrated in Fig. 3, and the associated hyperparameters are summarized in Table 3. During training, the model minimizes reconstruction error using **Mean squared error (MSE)**, which provides smoother gradients and promotes stable convergence. At inference, anomalies are detected using the **Mean absolute error (MAE)** of the reconstruction, since MAE is more robust to heavy-tailed residuals. An input is flagged as anomalous if its MAE exceeds the threshold $\theta = 0.65$, as determined through sensitivity analysis (Section 4.4).

Model training is implemented using TensorFlow (Keras API). Node-level training pipelines are automated with batch scripts and Python orchestration. Detection performance is measured using the F1-score and the Area Under the ROC Curve (AUC) [11].

Table 4
Sensitivity of FL-LSTM to key hyperparameters.

Hyperparameter	Range	Best Value / Observation
Learning Rate	10^{-3}-10^{-7}	Best at 10^{-6}; higher unstable, lower too slow.
Batch Size	32, 64, 128, 256	**128** balanced variance and speed.
Epochs	1-10	**5** epochs sufficient; more gave no gain.
Time Steps	5-15	**10** steps optimal for temporal capture.

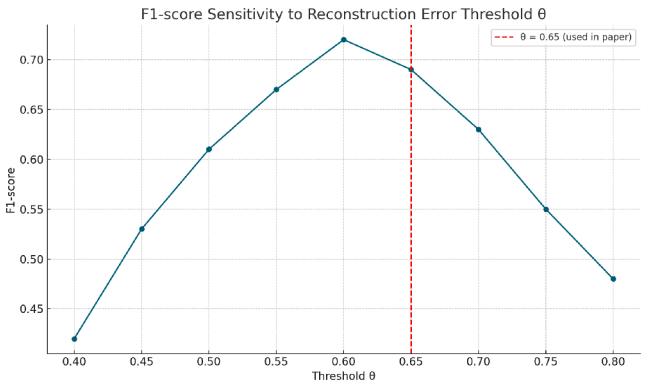


Fig. 4. F1-score variation with respect to reconstruction error threshold (θ).

4.2.3. Federated learning setup

Federated Learning is implemented with the Flower framework [39]. In this setup:

- Each HPC node acts as an FL client, running a local LSTM autoencoder.
- A central server (management node) coordinates aggregation and broadcasts updates.

Initially, each node trains its local autoencoder independently, providing the non-FL baseline. FL experiments are then conducted using six aggregation strategies: FedAvg, FedAvgM, FedAdagrad, FedYogi, FedProx, and FedAdam, all supported by Flower. The `FlowerClient` class enables node participation in each training round.

We evaluate FL with 1000, 1500, and 2000 communication rounds. Since performance converged after approximately 1000 rounds, further round-variation analysis is omitted for brevity.

4.3. Hyperparameter sensitivity analysis of anomaly detection model

To assess the robustness of our anomaly detection framework, we performed a sensitivity analysis by varying key hyperparameters.

Table 4 summarizes the impact of each parameter:

- **Learning rate:** Values above 10^{-5} caused unstable training, while values below 10^{-7} slowed convergence without delivering meaningful accuracy gains.
- **Batch size:** Smaller batches (e.g., 32) improved sensitivity to anomalies but introduced higher variance. In contrast, larger batches (e.g., 256) reduced variance but slightly decreased F1 performance.
- **Epochs:** Increasing the number of epochs beyond 5 yielded negligible improvements, indicating that the model converges early in the federated setting.
- **Time steps:** We experimented with ranges between 5 and 15. A sequence length of 10 offered the best balance between capturing temporal dependencies and maintaining computational efficiency.

These insights guided the selection of final hyperparameters. The Federated Learning (FL) implementation was tuned to support scalability, minimize communication costs, and remain resilient against client-side heterogeneity and potential faults in HPC environments.

4.4. Reconstruction error threshold sensitivity analysis

Anomaly detection models based on autoencoders–including the LSTM-based models used in this study–classify input samples as normal or anomalous according to a reconstruction error threshold (θ). In our baseline setup, this threshold was fixed at 0.65, guided by empirical evidence and prior literature [9]. However, the choice of θ is critical, as it directly affects the balance between precision and recall, particularly in scenarios where anomalies are rare or subtle. Fig. 4 presents the results of our threshold sensitivity analysis.

4.4.1. Experimental setup

To investigate the influence of θ on model performance, we conducted a sensitivity analysis using Group A nodes from Dataset D1 in the unsupervised setting. The threshold θ was systematically varied from 0.40 to 0.80 in increments of 0.05, and the corresponding F1-scores were recorded. All other model components and training procedures were kept constant.

4.4.2. Results and discussion

The analysis indicates that performance peaks around $\theta = 0.65$, which validates our initial choice. Nevertheless, performance degrades noticeably outside this range. At thresholds below 0.55, the model becomes overly sensitive, producing a large number of false positives. In contrast, thresholds above 0.70 reduce sensitivity to anomalies, increasing false negatives.

Although Fig. 4 shows that F1 performance is locally high around $\theta = 0.60$, this setting leads to increased false positives and less stable behavior across datasets and training modes. By contrast, $\theta = 0.65$ consistently provides the best balance between precision and recall, which is why it was adopted for all subsequent analyses. We also verified that using $\theta = 0.60$ does not alter the overall conclusions: federated models continue to outperform non-FL baselines in all regimes (Tables 5–7). This threshold choice is therefore both empirically justified and aligned with prior HPC studies [14,25].

4.5. Metrics

The performance of the proposed approach was evaluated using two widely adopted metrics in line with state-of-the-art methodologies [9,24]: (i) the F1-score and (ii) the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). The F1-score, defined in (12), represents the harmonic mean of precision and recall, and provides a balanced measure that considers both false positives and false negatives.

$$F1\text{-}Score = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

The AUC-ROC metric [40] quantifies the overall ability of the model to discriminate between positive and negative classes by computing the area under the ROC curve. This curve plots the true positive rate against the false positive rate across various classification thresholds, and the AUC value corresponds to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative one.

4.6. Results

This section presents the experimental results that demonstrate the effectiveness of our proposed approach. To evaluate the performance, we address the following research questions:

1. **RQ1:** Does the integration of FL with anomaly detection techniques in HPC environments enhance the performance of anomaly detection models?
2. **RQ2:** Can the adoption of FL reduce the amount of training data needed to achieve comparable performance to baseline, non-FL anomaly detection models trained on the full dataset?

Table 5
F1 and AUC scores for FL vs. Non-FL in unsupervised learning under data reduction for D1 and D2. Wilcoxon signed-rank test confirms significance ($p < .01$).

Dataset	Group	Metric	Full	1/2	1/4	1/8	1/16
D1	A	Non-FL F1	0.31	0.24	0.20	0.18	0.12
		FL F1	**0.86**	**0.65**	**0.45**	**0.40**	**0.35**
		Non-FL AUC	0.38	0.25	0.17	0.14	0.08
		FL AUC	**0.74**	**0.55**	**0.43**	**0.42**	**0.40**
	B	Non-FL F1	0.25	0.19	0.15	0.13	0.09
		FL F1	**0.78**	**0.59**	**0.51**	**0.39**	**0.33**
		Non-FL AUC	0.44	0.31	0.21	0.16	0.12
		FL AUC	**0.77**	**0.58**	**0.49**	**0.48**	**0.46**
	C	Non-FL F1	0.37	0.28	0.22	0.19	0.13
		FL F1	**0.87**	**0.64**	**0.57**	**0.49**	**0.42**
		Non-FL AUC	0.47	0.35	0.26	0.22	0.18
		FL AUC	**0.85**	**0.62**	**0.53**	**0.50**	**0.49**
	D	Non-FL F1	0.24	0.19	0.16	0.14	0.11
		FL F1	**0.84**	**0.65**	**0.53**	**0.34**	**0.29**
		Non-FL AUC	0.33	0.27	0.14	0.11	0.07
		FL AUC	**0.69**	**0.44**	**0.39**	**0.36**	**0.34**
	E	Non-FL F1	0.22	0.17	0.13	0.08	0.05
		FL F1	**0.79**	**0.57**	**0.48**	**0.33**	**0.28**
		Non-FL AUC	0.25	0.18	0.09	0.05	0.01
		FL AUC	**0.63**	**0.40**	**0.31**	**0.29**	**0.26**
D2	A	Non-FL F1	0.54	0.45	0.36	0.25	0.18
		FL F1	**0.96**	**0.85**	**0.74**	**0.63**	**0.59**
		Non-FL AUC	0.51	0.43	0.35	0.24	0.15
		FL AUC	**0.94**	**0.85**	**0.73**	**0.59**	**0.52**
	B	Non-FL F1	0.42	0.38	0.29	0.18	0.09
		FL F1	**0.85**	**0.76**	**0.68**	**0.57**	**0.49**
		Non-FL AUC	0.42	0.33	0.21	0.14	0.09
		FL AUC	**0.82**	**0.73**	**0.60**	**0.51**	**0.43**
	C	Non-FL F1	0.22	0.14	0.09	0.06	0.03
		FL F1	**0.70**	**0.61**	**0.52**	**0.44**	**0.38**
		Non-FL AUC	0.17	0.12	0.09	0.05	0.01
		FL AUC	**0.67**	**0.59**	**0.48**	**0.39**	**0.29**
	D	Non-FL F1	0.33	0.25	0.16	0.09	0.05
		FL F1	**0.77**	**0.68**	**0.57**	**0.48**	**0.37**
		Non-FL AUC	0.30	0.28	0.20	0.11	0.05
		FL AUC	**0.80**	**0.69**	**0.57**	**0.42**	**0.35**
	E	Non-FL F1	0.31	0.24	0.15	0.08	0.04
		FL F1	**0.81**	**0.73**	**0.64**	**0.53**	**0.46**
		Non-FL AUC	0.28	0.17	0.11	0.08	0.04
		FL AUC	**0.78**	**0.67**	**0.56**	**0.45**	**0.36**

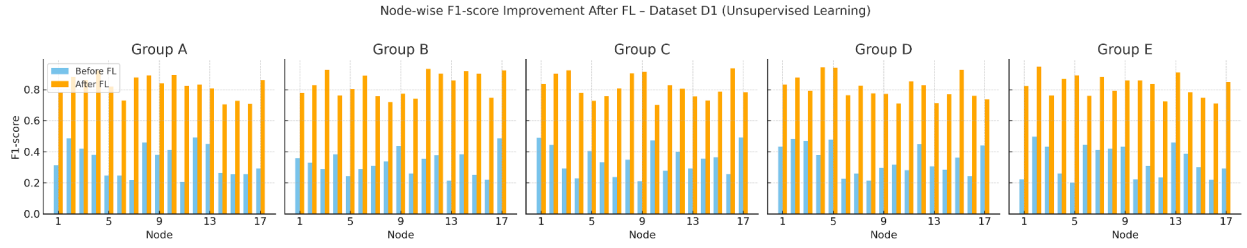
3. **RQ3:** Does the use of advanced FL aggregation algorithms further improve the performance of anomaly detection models compared to the conventional FedAvg algorithm?

To evaluate the performance of FL-powered LSTM autoencoders, Tables 5–7 present detailed F1 and AUC scores across five groups (A–E) and two datasets (D1 and D2) under varying data availability levels. These tables highlight the consistent and substantial improvements that FL provides across all learning paradigms.

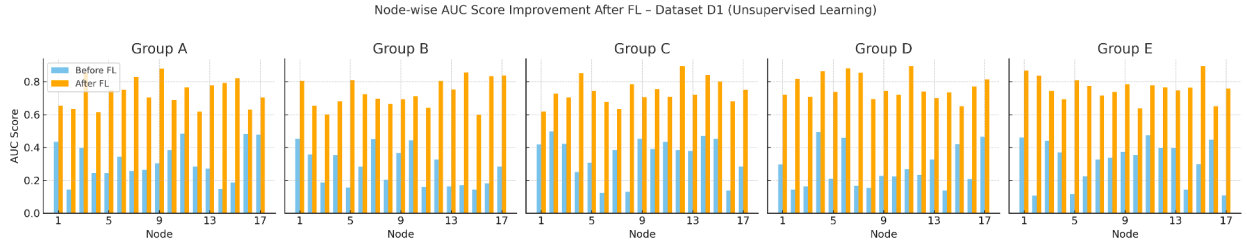
Unsupervised learning (Table 5). In the absence of labeled data, FL significantly enhances anomaly detection capability. On Dataset D1, Group C's F1 score improves from 0.37 (non-FL) to 0.87 (FL) and AUC from 0.47 to 0.85 –a +135 % and +81 % increase, respectively. Similar trends are observed across other groups, with FL consistently outperforming the standalone LSTM baseline.

For Dataset D2, FL achieves even stronger gains. In Group A, the F1 score improves from 0.54 to 0.96, while the AUC jumps from 0.51 to 0.94. Even under *extreme data reduction (1/16)*, FL maintains robust performance, e.g., Group E improves F1 from 0.04 (non-FL) to 0.46 (FL). These results confirm that even without access to labeled samples, FL leverages distributed patterns across nodes to generalize anomaly detection with minimal data–supporting RQ1 and RQ2.

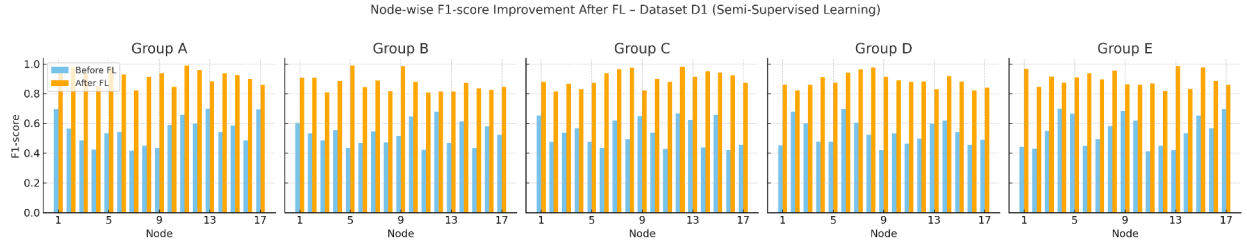
A potential concern in unsupervised training is that the autoencoder may inadvertently learn anomalous temporal patterns, since both normal and abnormal samples appear in the training set. In our case, this



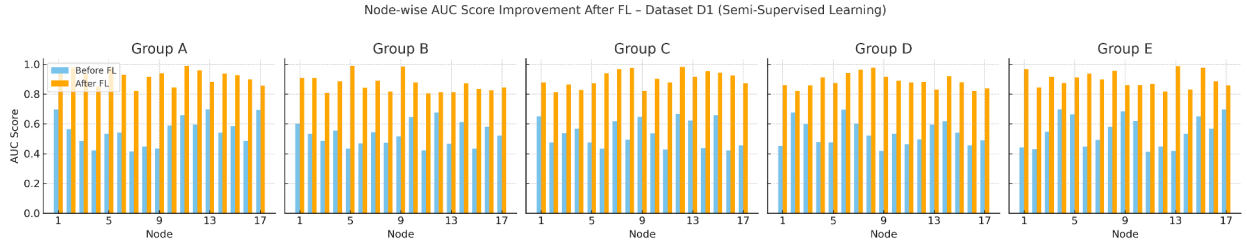
(a) F1-score – D1 (Unsupervised)



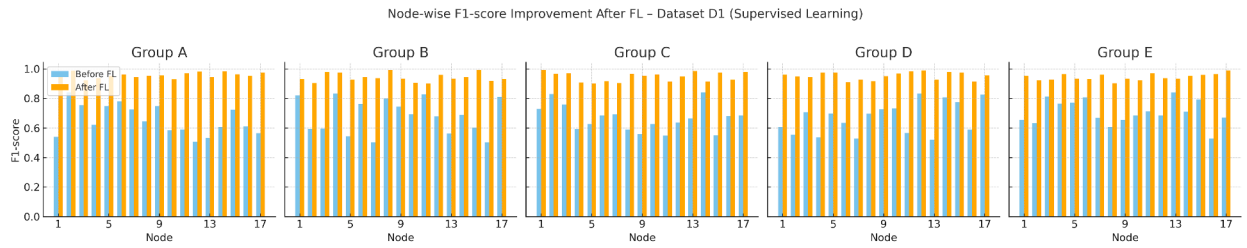
(b) AUC – D1 (Unsupervised)



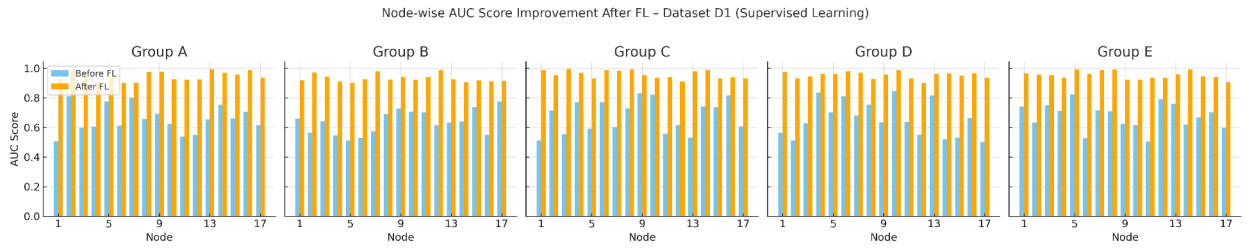
(c) F1-score – D1 (Semi-Supervised)



(d) AUC – D1 (Semi-Supervised)

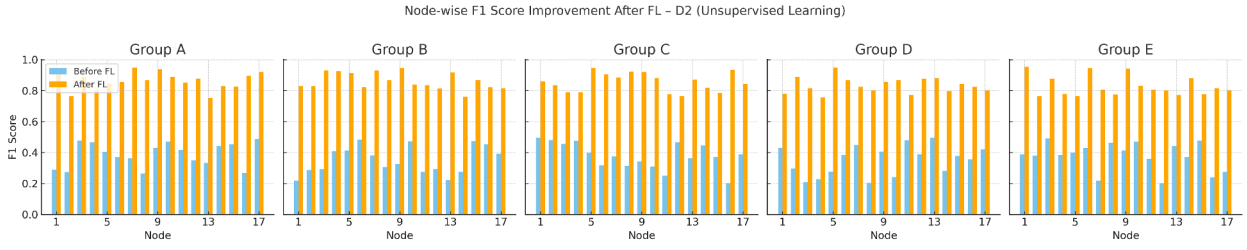


(e) F1-score – D1 (Supervised)

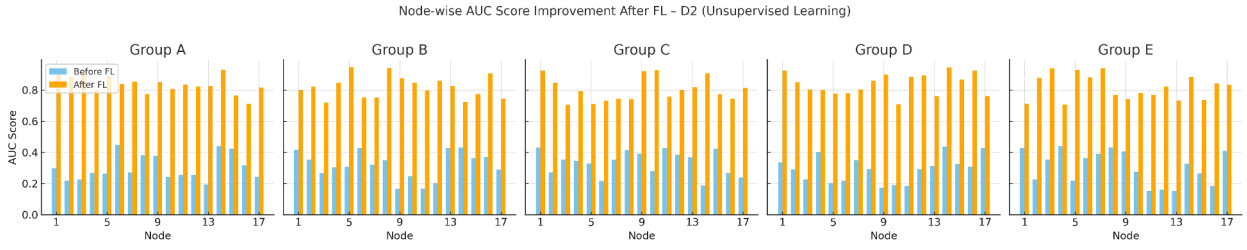


(f) AUC – D1 (Supervised)

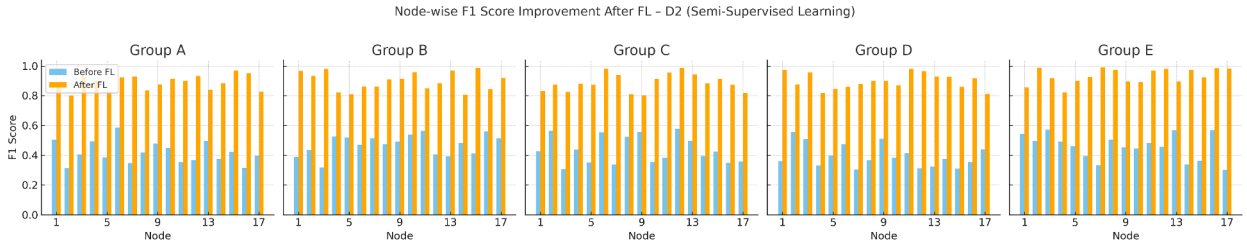
Fig. 5. Node-wise improvement in F1 and AUC scores across all learning modes (Unsupervised, Semi-Supervised, Supervised) for **Dataset D1**. Each subfigure compares baseline and post-FL performance across Groups A-E.



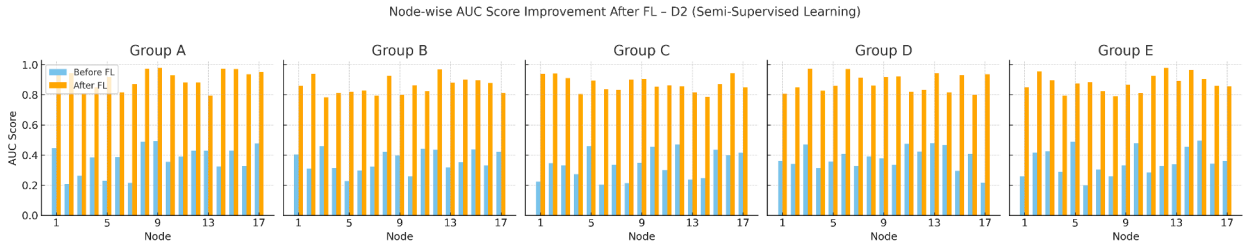
(a) F1-score Improvement – D2 (Unsupervised)



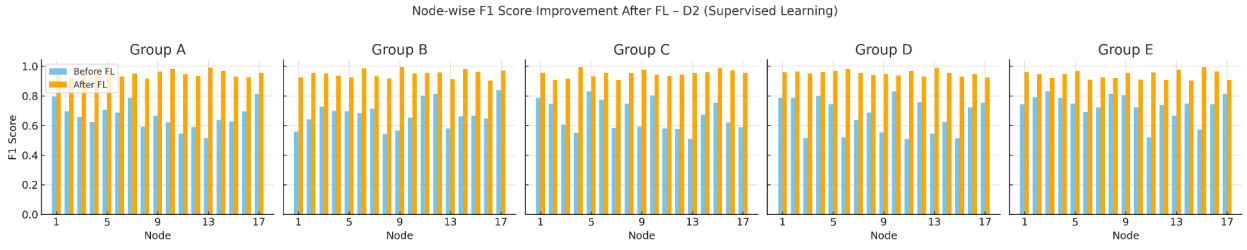
(b) AUC Improvement – D2 (Unsupervised)



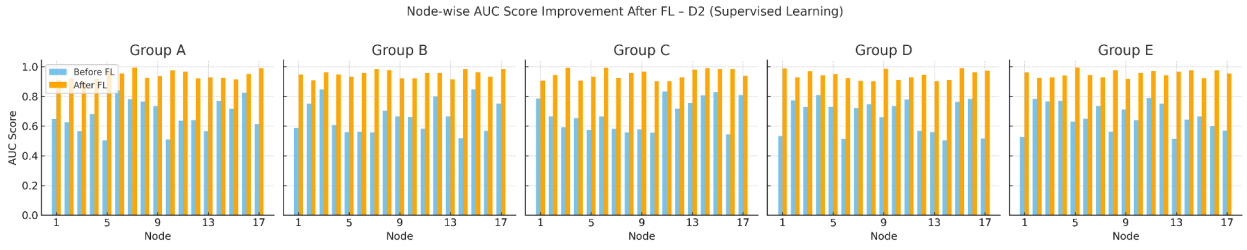
(c) F1-score Improvement – D2 (Semi-Supervised)



(d) AUC Improvement – D2 (Semi-Supervised)



(e) F1-score Improvement – D2 (Supervised)



(f) AUC Improvement – D2 (Supervised)

Fig. 6. Node-wise improvement in F1 and AUC scores across all learning paradigms (Unsupervised, Semi-Supervised, Supervised) for **Dataset D2**. Each subfigure shows scores before and after applying Federated Learning (FL) for Groups A-E.

Table 6
F1 and AUC scores for FL vs. Non-FL in semi-supervised learning under data reduction (D1 & D2). Wilcoxon signed-rank test confirms significance ($p < .01$).

Dataset	Group	Metric	Full	1/2	1/4	1/8	1/16
D1	A	Non-FL F1	0.68	0.57	0.48	0.43	0.33
		FL F1	**0.95**	**0.85**	**0.64**	**0.62**	**0.60**
		Non-FL AUC	0.58	0.44	0.38	0.35	0.26
		FL AUC	**0.92**	**0.76**	**0.65**	**0.63**	**0.58**
	B	Non-FL F1	0.62	0.52	0.43	0.38	0.30
		FL F1	**0.87**	**0.79**	**0.70**	**0.61**	**0.58**
		Non-FL AUC	0.64	0.50	0.42	0.37	0.30
		FL AUC	**0.95**	**0.79**	**0.71**	**0.69**	**0.64**
	C	Non-FL F1	0.74	0.61	0.50	0.44	0.34
		FL F1	**0.96**	**0.84**	**0.76**	**0.71**	**0.67**
		Non-FL AUC	0.67	0.54	0.47	0.43	0.36
		FL AUC	**0.99**	**0.83**	**0.75**	**0.71**	**0.67**
	D	Non-FL F1	0.61	0.52	0.44	0.39	0.32
		FL F1	**0.93**	**0.85**	**0.72**	**0.56**	**0.54**
		Non-FL AUC	0.53	0.46	0.35	0.32	0.25
		FL AUC	**0.87**	**0.65**	**0.61**	**0.57**	**0.52**
	E	Non-FL F1	0.59	0.50	0.41	0.33	0.26
		FL F1	**0.88**	**0.77**	**0.67**	**0.55**	**0.53**
		Non-FL AUC	0.45	0.37	0.30	0.26	0.19
		FL AUC	**0.81**	**0.61**	**0.53**	**0.50**	**0.44**
D2	A	Non-FL F1	0.91	0.78	0.64	0.50	0.39
		FL F1	**0.99**	**0.99**	**0.93**	**0.85**	**0.84**
		Non-FL AUC	0.71	0.62	0.56	0.45	0.33
		FL AUC	**0.99**	**0.99**	**0.95**	**0.80**	**0.70**
	B	Non-FL F1	0.79	0.71	0.57	0.43	0.30
		FL F1	**0.94**	**0.96**	**0.87**	**0.79**	**0.74**
		Non-FL AUC	0.62	0.52	0.42	0.35	0.27
		FL AUC	**0.99**	**0.94**	**0.82**	**0.72**	**0.61**
	C	Non-FL F1	0.59	0.47	0.37	0.31	0.24
		FL F1	**0.79**	**0.81**	**0.71**	**0.66**	**0.63**
		Non-FL AUC	0.37	0.31	0.30	0.26	0.19
		FL AUC	**0.85**	**0.80**	**0.70**	**0.60**	**0.47**
	D	Non-FL F1	0.70	0.58	0.44	0.34	0.26
		FL F1	**0.86**	**0.88**	**0.76**	**0.70**	**0.62**
		Non-FL AUC	0.50	0.47	0.41	0.32	0.23
		FL AUC	**0.98**	**0.90**	**0.79**	**0.63**	**0.53**
	E	Non-FL F1	0.68	0.57	0.43	0.33	0.25
		FL F1	**0.90**	**0.93**	**0.83**	**0.75**	**0.71**
		Non-FL AUC	0.48	0.36	0.32	0.29	0.22
		FL AUC	**0.96**	**0.88**	**0.78**	**0.66**	**0.54**

Table 7
F1 and AUC scores for FL vs. Non-FL in supervised learning under data reduction (D1 & D2). Wilcoxon signed-rank test confirms significance ($p < .01$).

Dataset	Group	Metric	Full	1/2	1/4	1/8	1/16
D1	A	Non-FL F1	0.93	0.78	0.70	0.62	0.55
		FL F1	**0.99**	**0.99**	**0.90**	**0.87**	**0.88**
		Non-FL AUC	0.86	0.64	0.63	0.59	0.47
		FL AUC	**0.99**	**0.89**	**0.74**	**0.74**	**0.68**
	B	Non-FL F1	0.87	0.73	0.65	0.57	0.52
		FL F1	**0.99**	**0.99**	**0.96**	**0.86**	**0.86**
		Non-FL AUC	0.92	0.70	0.67	0.61	0.51
		FL AUC	**0.99**	**0.92**	**0.80**	**0.80**	**0.74**
	C	Non-FL F1	0.99	0.82	0.72	0.63	0.56
		FL F1	**0.99**	**0.99**	**0.99**	**0.96**	**0.95**
		Non-FL AUC	0.95	0.74	0.72	0.67	0.57
		FL AUC	**0.99**	**0.96**	**0.84**	**0.82**	**0.77**
	D	Non-FL F1	0.86	0.73	0.66	0.58	0.54
		FL F1	**0.99**	**0.99**	**0.98**	**0.81**	**0.82**
		Non-FL AUC	0.81	0.66	0.60	0.56	0.46
		FL AUC	**0.99**	**0.78**	**0.70**	**0.68**	**0.62**
	E	Non-FL F1	0.84	0.71	0.63	0.52	0.48
		FL F1	**0.99**	**0.99**	**0.93**	**0.80**	**0.81**
		Non-FL AUC	0.73	0.57	0.55	0.50	0.40
		FL AUC	**0.93**	**0.74**	**0.62**	**0.61**	**0.54**
D2	A	Non-FL F1	0.99	0.99	0.86	0.69	0.61
		FL F1	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
		Non-FL AUC	0.99	0.82	0.81	0.69	0.54
		FL AUC	**0.99**	**0.99**	**0.99**	**0.91**	**0.80**
	B	Non-FL F1	0.99	0.92	0.79	0.62	0.52
		FL F1	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
		Non-FL AUC	0.90	0.72	0.67	0.59	0.48
		FL AUC	**0.99**	**0.99**	**0.91**	**0.83**	**0.71**
	C	Non-FL F1	0.84	0.68	0.59	0.50	0.46
		FL F1	**0.99**	**0.99**	**0.97**	**0.91**	**0.85**
		Non-FL AUC	0.65	0.51	0.55	0.50	0.40
		FL AUC	**0.97**	**0.93**	**0.79**	**0.71**	**0.57**
	D	Non-FL F1	0.95	0.79	0.66	0.53	0.48
		FL F1	**0.99**	**0.99**	**0.99**	**0.95**	**0.90**
		Non-FL AUC	0.78	0.67	0.66	0.56	0.44
		FL AUC	**0.99**	**0.99**	**0.88**	**0.74**	**0.63**
	E	Non-FL F1	0.93	0.78	0.65	0.52	0.47
		FL F1	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**
		Non-FL AUC	0.76	0.56	0.57	0.53	0.43
		FL AUC	**0.99**	**0.99**	**0.87**	**0.77**	**0.64**

risk is mitigated by several factors. First, the anomaly base rate is very low (3 % in D1 and 1.7 % in D2, see Table 2), so the MSE loss is dominated by normal samples.

Second, federated aggregation promotes the learning of common normal behaviors across nodes, which reduces the influence of rare, node-specific anomalies. Third, our empirical results demonstrate that the model does not overfit to anomalies: FL models trained with only 1/8 or 1/16 of the training data consistently outperform non-FL models trained with the full dataset (Table 5). This robustness under data scarcity provides indirect evidence that the unsupervised FL approach captures generalizable normal patterns rather than anomalous ones.

Semi-supervised learning (Table 6). When trained only on normal data, FL demonstrates exceptional generalization. On D1, Group A's F1 score improves from 0.68 (non-FL) to 0.95 (FL) using full data. Similar improvements hold even under data scarcity; with just 1/4 of training data, FL achieves an F1 of 0.64, compared to 0.48 for non-FL.

Across D2, Group B reaches near-perfect detection: FL increases F1 from 0.79 to 0.94, and AUC from 0.62 to 0.99 using the full dataset. Importantly, with just 1/8 of data, FL retains high accuracy (F1 = 0.79 vs. 0.43 non-FL), confirming FL's ability to reduce training requirements (RQ2) without sacrificing performance. This generalization is attributed to FL's collaborative training, which allows each node to benefit from shared knowledge of normal behavior–even in the absence of anomaly labels.

Supervised learning (Table 7). With fully labeled data, FL consistently achieves near-optimal performance. In Dataset D1, Groups A through E reach F1 scores of 0.99 across the board when using FL, whereas non-FL models range from 0.84 to 0.93. The AUC values show similar trends, increasing from 0.73–0.95 (non-FL) to 0.99 (FL).

On Dataset D2, where data is more complex and anomalies are more sparse, FL maintains strong performance: for example, Group C improves from F1 = 0.84 (non-FL) to 0.99 (FL). Even with 1/16 of training data, FL achieves F1 = 0.99 in all groups, compared to as low as 0.47 for the non-FL baseline.

These findings reveal the scalability and robustness of FL in high-supervision settings, while also highlighting its ability to drastically reduce the need for data collection and labeling –answering RQ1 and RQ2 affirmatively.

4.6.1. Individual node analysis

Figs. 5 and 6 present node-level analyses of F1 and AUC improvements across all learning paradigms for Datasets D1 and D2, respectively. Each subplot contrasts baseline (non-FL) LSTM autoencoders with their federated counterparts, thereby illustrating the effect of FL on individual HPC nodes grouped into cohorts A through E.

For Dataset D1, FL consistently enhanced performance across all node groups. In the unsupervised setting, Group C demonstrated the most substantial improvement: several nodes achieved F1-score gains exceeding +100 %, with post-FL values approaching 0.87 compared to

baseline scores as low as 0.37. Comparable trends emerged in Group D under semi-supervised training, where federated learning boosted multiple nodes from F1 scores near 0.50 to above 0.85. In the supervised configuration, nearly all nodes attained F1 scores greater than 0.95 after FL, indicating near-perfect anomaly detection across the cohort.

For Dataset D2, the gains were even more pronounced, particularly under data-scarce conditions. In Group A, for instance, the lowest-performing node's F1 score rose from 0.54 to 0.96 in the unsupervised setting. Group B exhibited strong heterogeneity before FL, with F1 scores ranging from 0.09 to 0.42; federated training not only lifted these nodes into the 0.85-0.95 range but also reduced performance variance. Even in Group E–where semi-supervised baselines were extremely poor (F1 ≤ 0.25)–post-FL training elevated node-level scores to 0.70 or higher.

Overall, these results highlight the ability of FL to generalize anomaly detection capabilities across diverse nodes. The visual evidence in Figs. 5 and 6 provides strong support for **RQ1**, demonstrating that FL yields consistent and significant node-level improvements in anomaly detection.

4.6.2. Performance results using a limited training dataset

To evaluate the robustness of FL under practical constraints, we examined how reducing the amount of available training data affects anomaly detection performance. This directly addresses **RQ2**: *Can the adoption of FL reduce the amount of training data needed to achieve comparable or superior performance relative to non-FL anomaly detection models trained on the full dataset?*

We simulated data scarcity by training models on fractions of the full dataset: 1/2, 1/4, 1/8, and 1/16, corresponding to training durations of approximately 2.25 months, 1.25 months, 2.5 weeks, and 1.25 weeks, respectively. The full dataset spans 4.5 months of telemetry.

Motivation. In production-scale HPC environments, extended data collection cycles delay model deployment, undermining timely anomaly detection. Our objective is to assess whether FL can sustain high performance while substantially reducing this data collection burden.

Experimental setup. FL models were evaluated across three learning paradigms–unsupervised, semi-supervised, and supervised–on both D1 and D2 datasets. Results were averaged across five groups (A-E). For each data fraction, we compared:

- **Non-FL baselines**, trained independently on both full and reduced datasets.
- **FL models**, trained collaboratively on the same reduced subsets.

Key results (Tables 5–7).

- **Unsupervised learning:** FL trained with only 1/8 of the data often surpassed non-FL models trained on the full dataset.
  Example: D1 Group C – F1: 0.49 (FL, 1/8) vs. 0.37 (Non-FL, full).
- **Semi-supervised learning:** FL remained effective even with as little as 1/16 of the training data.
  Example: D2 Group A – F1: 0.84 (FL, 1/16) vs. 0.39 (Non-FL, 1/16).
- **Supervised learning:** FL consistently achieved near-optimal performance across all fractions.
  Example: D2 Group E – F1: 0.99 (FL, 1/16) vs. 0.47 (Non-FL, 1/16).

These findings clearly demonstrate that FL enables high anomaly detection accuracy with significantly less training data, strongly supporting **RQ2**.

Relation to RQ1. Although the focus here is data efficiency, the results also reinforce:

- **RQ1:** *Does FL integration improve anomaly detection in HPC environments?*
  Across all paradigms, FL consistently outperformed local LSTM baselines.

Conclusion. FL reduces training data requirements by up to **15**×, while sustaining or even improving detection accuracy. This makes FL particularly well-suited for real-world HPC environments, where rapid deployment and scalability are essential.

4.6.3. Performance evaluation of federated learning aggregation techniques

To evaluate the effectiveness of advanced FL aggregation strategies for anomaly detection, we conducted extensive experiments using six algorithms–FedAvg, FedProx, FedAdagrad, FedAvgM, FedYogi, and FedAdam. The evaluation spanned five groups (A-E) across both datasets (D1 and D2) under unsupervised, semi-supervised, and supervised learning paradigms. The detailed F1 and AUC results are reported in Tables 8 and 9, highlighting key trends in performance and generalization.

In these tables, the baseline marked as "ML" is not an additional algorithm but rather the Local AE baseline, i.e., non-federated LSTM autoencoders trained independently at each node, with results averaged within each group.

Across both datasets, **FedAdagrad** and **FedAvgM** consistently delivered the strongest results.

On Dataset D1, FedAdagrad achieved peak F1 scores of up to 0.94 (Group C, unsupervised) and 0.99 (Groups A-E, supervised), alongside AUC scores reaching 0.92 and 0.99, respectively. FedAvgM closely matched these results, performing particularly well in semi-supervised settings where both methods reached near-perfect scores across multiple groups. FedYogi and FedProx also yielded competitive outcomes, though with slightly less consistency. By contrast, FedAdam performed poorly, with F1 and AUC values rarely exceeding 0.20–suggesting convergence issues or instability in the federated HPC anomaly detection setting.

Summary of Table 8 (Dataset D1):

- **FedAdagrad:** Highest overall performance, up to 0.94 F1 and 0.99 AUC.
- **FedAvgM:** Nearly identical to FedAdagrad, excelling in semi-supervised training.
- **FedYogi & FedProx:** Strong but less consistent compared to the top two.
- **FedAdam:** Fails to generalize, with scores capped at 0.20.

On Dataset D2, these trends were reinforced. FedAdagrad and FedAvgM again achieved top-tier results, with both methods reaching 0.99 F1 and AUC in Group A under semi-supervised and supervised configurations. Their ability to generalize across groups and modes highlights resilience to variations in monitoring infrastructure and data distribution. Among the remaining methods, FedYogi was the most robust third performer–particularly in supervised settings–while FedProx remained competitive but less stable. FedAdam again showed the weakest performance, with F1 and AUC values consistently below 0.41.

Summary of Table 9 (Dataset D2):

- **FedAdagrad & FedAvgM:** Achieve near-perfect F1 and AUC across all groups and learning modes.
- **FedYogi:** A solid third option, especially in supervised and semi-supervised training.
- **FedProx:** Competitive but slightly less reliable across groups.
- **FedAdam:** Underperforms across all settings, never exceeding 0.41.

These findings provide strong evidence for RQ3: advanced FL aggregation techniques, particularly FedAdagrad and FedAvgM, substantially outperform the standard FedAvg baseline. Their consistent superiority across datasets and training modes underscores their suitability for production-scale anomaly detection in federated HPC environments, where communication constraints and heterogeneous data are critical factors.

Taken together, these results explain why FL remains robust under severe data reduction. The ablation study (Table 10) confirms that both

Table 8
F1 and AUC scores for different FL aggregation strategies on Dataset D1 across unsupervised, semi-supervised, and supervised learning modes. "ML" denotes the Local AE baseline (non-federated LSTM autoencoders trained independently at each node, with group averages reported).

Group	Strategy	F1 Score			AUC Score		
		Unsupervised	Semi-Supervised	Supervised	Unsupervised	Semi-Supervised	Supervised
A	ML	0.31	0.68	0.93	0.38	0.58	0.86
	FedAvg	0.86	0.95	0.99	0.74	0.92	**0.99**
	FedAdagrad	**0.93**	**0.99**	0.99	0.80	0.96	0.99
	FedAvgM	0.90	**0.99**	0.99	0.77	**0.99**	0.99
	FedProx	0.82	0.91	0.95	0.71	0.88	0.95
	FedYogi	0.89	0.95	0.95	0.76	0.95	0.95
	FedAdam	0.18	0.20	0.20	0.16	0.20	0.20
B	ML	0.25	0.62	0.87	0.44	0.64	0.92
	FedAvg	0.78	0.87	0.99	0.77	0.95	0.99
	FedAdagrad	0.84	0.91	0.99	0.83	**0.99**	0.99
	FedAvgM	0.81	**0.99**	0.99	0.80	**0.99**	0.99
	FedProx	0.74	0.83	0.95	0.73	0.91	0.95
	FedYogi	0.80	0.95	0.95	0.79	0.95	0.95
	FedAdam	0.17	0.19	0.20	0.16	0.20	0.20
C	ML	0.37	0.74	0.99	0.47	0.67	0.95
	FedAvg	0.87	0.96	0.99	0.85	0.99	0.99
	FedAdagrad	**0.94**	**0.99**	0.99	**0.92**	0.99	0.99
	FedAvgM	0.91	**0.99**	0.99	0.89	**0.99**	0.99
	FedProx	0.83	0.92	0.95	0.81	0.95	0.95
	FedYogi	0.90	0.95	0.95	0.88	0.95	0.95
	FedAdam	0.19	0.20	0.20	0.18	0.20	0.20
D	ML	0.24	0.61	0.86	0.33	0.53	0.81
	FedAvg	0.84	0.93	0.99	0.69	0.87	0.99
	FedAdagrad	**0.91**	0.97	0.99	0.74	0.91	0.99
	FedAvgM	0.87	**0.99**	0.99	0.72	0.94	0.99
	FedProx	0.80	0.89	0.95	0.66	0.83	0.95
	FedYogi	0.87	0.95	0.95	0.71	0.90	0.95
	FedAdam	0.18	0.20	0.20	0.15	0.19	0.20
E	ML	0.22	0.59	0.84	0.25	0.45	0.73
	FedAvg	0.79	0.88	0.99	0.63	0.81	0.93
	FedAdagrad	**0.85**	0.92	0.99	0.68	0.84	0.99
	FedAvgM	0.82	**0.95**	0.99	0.65	0.87	**0.97**
	FedProx	0.75	0.84	0.95	0.60	0.77	0.89
	FedYogi	0.81	0.91	0.95	0.65	0.83	0.95
	FedAdam	0.17	0.19	0.20	0.13	0.17	0.20

temporal modeling through LSTMs and federation itself provide complementary gains, which jointly sustain accuracy even with limited samples. The comparison of aggregation strategies (Tables 8–9) shows that adaptive server-side optimizers such as FedAdagrad and FedAvgM mitigate non-IID effects and stabilize training under sparse anomalies, outperforming the FedAvg baseline.

Furthermore, latent space visualizations (Fig. 7) reveal improved separation between normal and anomalous patterns after FL, consistent with the enhanced F1 and AUC scores reported in Tables 5–7. Node-level improvements across groups (Figs. 5–6) and the robustness of threshold sensitivity (Fig. 4) further support that FL aggregates generalizable normal patterns across nodes, enabling resilient anomaly detection even under data scarcity.

Qualitative interpretation of FL gains While quantitative metrics such as F1-score and AUC confirm the effectiveness of Federated Learning (FL), understanding the underlying reasons for these improvements is essential for both practical deployment and scientific validation.

One key observation is that HPC nodes, though monitored individually, often share similar operational patterns under normal conditions. FL leverages this commonality by enabling models to collaboratively learn generalized representations of normal behavior. This collective learning enhances anomaly detection on individual nodes, particularly when local anomaly data is limited or unevenly distributed.

Another important factor is the choice of aggregation strategy. Advanced optimizers such as FedAdagrad and FedYogi consistently outperform FedAvg in several settings. By adapting learning rates based on gradient history, they demonstrate greater robustness to non-IID data distributions and sparse anomalies. In contrast, FedAdam underperforms due to its sensitivity to noisy gradient updates–a frequent challenge in HPC environments, where anomalies are rare and irregularly distributed.

FL also shows clear benefits in reduced-data scenarios. By aggregating knowledge across nodes, it mitigates performance degradation for nodes with scarce historical data or those recently introduced into the system. This ability to generalize from shared operational patterns makes the global model more resilient to data scarcity and better suited for dynamic HPC environments.

Overall, the observed gains in F1-score and AUC stem from FL's capacity to integrate common behavioral dynamics across nodes. While anomalies remain node-specific and infrequent, many normal states are shared among nodes. FL captures these shared dynamics, thereby strengthening model generalization and improving anomaly detection across the entire system.

It is important to note that our evaluation does not attempt to catalogue all external SOTA time-series anomaly detectors, as such methods are designed for centralized settings with different data semantics. Instead, the baselines chosen here are those most operationally relevant for HPC: (i) local vs. federated autoencoders, and (ii) optimizer-level variations. These allow us to directly measure the contribution of federation under realistic non-IID conditions. Given that FedAdagrad and FedAvgM already approach near-saturation performance across learning regimes, additional centralized baselines would provide limited extra value in this context.

Table 9
F1 and AUC scores for different FL aggregation strategies on Dataset D2 across unsupervised, semi-supervised, and supervised learning modes. "ML" denotes the Local AE baseline (non-federated LSTM autoencoders trained independently at each node, with group averages reported).

Group	Strategy	F1 Score			AUC Score		
		Unsupervised	Semi-Supervised	Supervised	Unsupervised	Semi-Supervised	Supervised
A	ML	0.51	0.74	0.94	0.51	0.66	0.94
	FedAvg	0.94	0.97	0.99	0.94	0.98	0.99
	FedAdagrad	**0.97**	**0.99**	0.99	**0.97**	0.99	0.99
	FedAvgM	0.96	**0.99**	0.99	0.96	**0.99**	0.99
	FedProx	0.91	0.93	0.96	0.91	0.95	0.96
	FedYogi	0.95	0.97	0.97	0.95	0.97	0.97
	FedAdam	0.35	0.40	0.41	0.34	0.40	0.40
B	ML	0.42	0.67	0.89	0.42	0.63	0.82
	FedAvg	0.82	0.93	0.99	0.82	0.95	0.99
	FedAdagrad	0.91	0.96	0.99	0.91	**0.97**	0.99
	FedAvgM	0.88	**0.99**	0.99	0.88	**0.99**	0.99
	FedProx	0.80	0.89	0.95	0.80	0.91	0.95
	FedYogi	0.87	0.94	0.95	0.87	0.94	0.95
	FedAdam	0.33	0.37	0.38	0.32	0.36	0.37
C	ML	0.17	0.59	0.81	0.17	0.52	0.67
	FedAvg	0.67	0.87	0.98	0.67	0.90	0.98
	FedAdagrad	0.73	0.91	0.99	0.73	0.92	0.99
	FedAvgM	0.71	**0.99**	0.99	0.71	**0.99**	0.99
	FedProx	0.60	0.83	0.93	0.60	0.85	0.93
	FedYogi	0.69	0.90	0.94	0.69	0.90	0.94
	FedAdam	0.28	0.31	0.31	0.26	0.30	0.30
D	ML	0.30	0.65	0.85	0.30	0.56	0.79
	FedAvg	0.80	0.92	0.99	0.80	0.94	0.99
	FedAdagrad	**0.88**	0.95	0.99	0.88	0.96	0.99
	FedAvgM	0.85	**0.99**	0.99	0.85	**0.99**	0.99
	FedProx	0.77	0.89	0.95	0.77	0.91	0.95
	FedYogi	0.84	0.93	0.95	0.84	0.93	0.95
	FedAdam	0.31	0.35	0.36	0.30	0.34	0.35
E	ML	0.28	0.62	0.84	0.28	0.53	0.73
	FedAvg	0.78	0.91	0.99	0.78	0.93	0.99
	FedAdagrad	**0.84**	0.94	0.99	0.84	0.95	0.99
	FedAvgM	0.81	**0.99**	0.99	0.81	**0.99**	0.99
	FedProx	0.75	0.87	0.95	0.75	0.89	0.95
	FedYogi	0.80	0.92	0.95	0.80	0.92	0.95
	FedAdam	0.30	0.34	0.35	0.28	0.33	0.34

4.7. Ablation study: temporal modeling and aggregation effects

To gain deeper insight into the contribution of individual components of our framework, we conducted an ablation study. This analysis systematically modifies or removes specific architectural and training elements to assess their effect on overall performance. We focus on two key aspects:

1. Temporal modeling through LSTM autoencoders
2. The influence of federated aggregation algorithms

The objective is to isolate and quantify the role of each component in achieving robust anomaly detection.

Effect of temporal modeling

We replaced the LSTM autoencoders with dense (fully connected) autoencoders while keeping all other factors unchanged, including the training data, hyperparameters, and federated setup. This allows us to directly measure the importance of modeling temporal dependencies in HPC telemetry.

Observation:

As shown in Table 10, LSTM autoencoders consistently outperform dense autoencoders, improving F1 and AUC scores by 15-16 % across both datasets. This highlights that capturing temporal patterns is crucial for detecting subtle, evolving anomalies in HPC telemetry data.

Effect of federated aggregation algorithms

To evaluate the contribution of federated learning strategies, we removed all aggregation mechanisms and relied solely on local training at each node (i.e., no FL). We then compared this setup against models trained with FedAvg, FedYogi, and FedAdagrad.

Observation:

As reported in Table 11, federated learning significantly improves performance compared to standalone training. Moreover, advanced optimizers such as FedAdagrad yield the highest gains. These findings demonstrate that both temporal modeling and federated aggregation play complementary roles, jointly enhancing anomaly detection in distributed HPC environments.

Table 10
Ablation study: temporal modeling effect (Dense vs. LSTM autoencoders with FL).

Model architecture	F1 (D1)	AUC (D1)	F1 (D2)	AUC (D2)
Dense Autoencoder (FL)	0.72	0.71	0.74	0.72
LSTM Autoencoder (FL)	**0.87**	**0.81**	**0.90**	**0.84**

Table 11
Ablation study: effect of FL aggregation vs. local training on D1 and D2.

Model Variant	FL Strategy	F1 (D1)	AUC (D1)	F1 (D2)	AUC (D2)
Local AE Only	None	0.45	0.39	0.49	0.42
LSTM-AE + FL	FedAvg	0.86	0.74	0.90	0.84
LSTM-AE + FL	FedYogi	0.89	0.76	0.92	0.87
LSTM-AE + FL	FedAdagrad	**0.93**	**0.80**	**0.94**	**0.89**

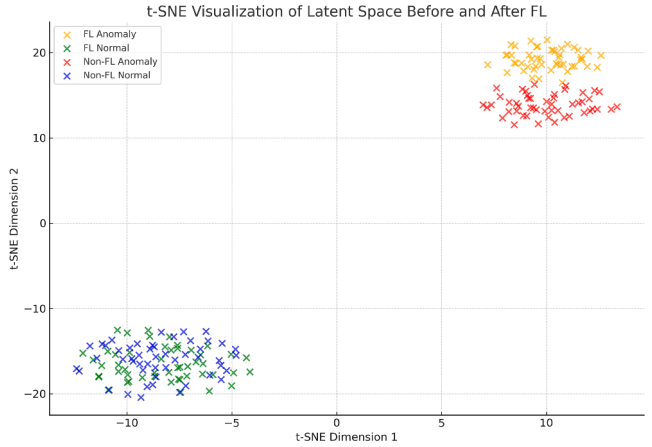


Fig. 7. t-SNE visualization of latent space representations before and after applying Federated Learning (FL). The blue and green clusters (non-FL normal and FL normal, respectively) show significant overlap, while anomalies (red for non-FL, orange for FL) remain more distinctly separated after FL, indicating improved anomaly discrimination.

4.8. Statistical significance

To evaluate the statistical significance of the performance improvements achieved by FL over local LSTM-based anomaly detectors in HPC environments, we apply the Wilcoxon signed-rank test [41].

This non-parametric test is particularly appropriate in our context because it does not assume normality of the underlying distributions–a critical factor given the non-Gaussian nature of node-specific telemetry data [42]. By comparing paired F1-scores for each client before and after FL integration, the test provides a robust mechanism to determine whether the observed gains are consistent and statistically reliable across clients [43].

In doing so, we ensure that the improvements attributed to FL are not the result of random variation or skewed by outlier nodes [44]. This strengthens the reliability of our claims regarding FL's generalization benefits in distributed HPC systems.

Concretely, we applied the Wilcoxon signed-rank test to compare baseline and FL-enhanced F1-scores across the five node groups. Across all learning paradigms and both datasets (D1 and D2), the results confirmed statistical significance ($p < 0.01$), validating the robustness of FL's contribution to anomaly detection performance.

4.9. Latent space visualization using t-SNE

To gain deeper insight into how Federated Learning (FL) enhances internal feature representations, we conducted a qualitative analysis of the LSTM encoder's latent space using t-distributed Stochastic Neighbor Embedding (t-SNE). Fig. 7 shows the 2D projections of the high-dimensional latent vectors produced by LSTM autoencoders under both non-FL and FL settings.

In the non-FL case, the latent embeddings of normal and anomalous instances form overlapping clusters, reflecting limited discriminative capability of the local models. In contrast, FL leads to more compact and distinctly separated clusters, especially for anomalous instances. This separation indicates that FL promotes the learning of generalized and discriminative representations by aggregating information across multiple nodes.

These observations align with the quantitative improvements in F1-score and AUC, highlighting the representational advantages gained through federated collaboration.

Interpretation of latent space visualization. Fig. 7 illustrates the distribution of normal (blue) and anomalous (red) samples in the latent space of the LSTM autoencoder, together with their associated reconstruction errors. We emphasize that the purpose of this visualization is not to show strict point-level overlap, but rather to highlight that anomalies–while scattered–tend to occupy peripheral regions where reconstruction error values are elevated. In contrast, normal points cluster tightly in central regions of the latent space, where reconstruction errors remain consistently low.

This behavior is desirable for anomaly detection: anomalies naturally diverge from normal clusters due to their irregular temporal dynamics, and their separation is reinforced by the elevated reconstruction error. Thus, latent space separability and reconstruction error provide two complementary signals that jointly enhance the model's ability to distinguish normal and anomalous behavior.

4.10. Deployment feasibility analysis

To assess the practicality of deploying the proposed Federated Learning (FL)-based anomaly detection framework in real-world high-performance computing (HPC) environments, we carried out a detailed analysis focusing on computational efficiency, scalability, and resource utilization. Although our implementation was tested on a non-HPC platform, the observed performance strongly suggests production-level readiness.

Each client model–an LSTM autoencoder representing an individual HPC node–completed local training in roughly 26 s. A full FL training round, including server-side aggregation and communication overhead, required approximately 53 s. Inference latency was consistently low, averaging 2.4 s for local models and 2.9 s for the global model. These runtimes demonstrate the framework's ability to support near real-time anomaly detection, which is critical for proactive fault mitigation in mission-critical infrastructures.

Memory and computational demands remained modest, ensuring compatibility with diverse system configurations. No performance bottlenecks were detected even as the system was scaled to multiple clients. This lightweight, non-intrusive design aligns well with the operational constraints of HPC environments, where minimizing overhead is a priority.

When extended to production-grade clusters–equipped with high-throughput interconnects and specialized accelerators–the framework is expected to benefit from lower communication latency and faster convergence. These gains could be further enhanced by techniques such as hierarchical FL or federated dropout. Moreover, the modular architecture of the framework enables seamless integration with established monitoring stacks (e.g., Examon, Nagios), allowing plug-and-play deployment without disrupting existing workflows.

In addition to offline and batch-based inference, the framework can be extended for streaming anomaly detection, supporting integration with real-time alerting systems and dynamic resource management. Its ability to generalize across nodes while requiring only 1.25 weeks of training data–a 15-fold reduction compared to conventional methods–significantly reduces the time-to-deployment, making it suitable even during the early stages of system commissioning.

Overall, the combination of empirical results, architectural scalability, and operational efficiency demonstrates the high feasibility of deploying the proposed framework in both current and future HPC environments. By offering a strong balance of accuracy, responsiveness, and deployability, the system presents a practical solution for proactive, distributed anomaly detection in production-scale computing infrastructures.

4.11. Computational and communication overhead of federated learning

While Federated Learning (FL) provides important architectural benefits for distributed anomaly detection–most notably decentralized training and the avoidance of raw data sharing–its deployment in HPC

environments requires a careful evaluation of both computational and communication overheads.

Computational cost. Each client trains a local LSTM autoencoder. As summarized in Table 3, the architecture is lightweight, consisting of a small number of LSTM layers and optimized for fast convergence. On our HPC infrastructure, training a client model for 5 epochs with a batch size of 128 required, on average, less than 30 s. Even across 1000 communication rounds, the computational load remained well within acceptable limits, with no evidence of bottlenecks. The choice of compact LSTM autoencoders ensures scalability without straining compute resources.

Communication cost. In each round, participating nodes transmit their model updates (weights) to the central server. With approximately 300,000 parameters per model stored in `float32`, each upload and download is about 0.6 MB, resulting in 1.2 MB per client per round. With 17 clients, this corresponds to roughly 20 MB of communication per round. Over 1000 rounds, the total volume reaches approximately 20 GB. Given the high-bandwidth interconnects of Tier-0 HPC systems, this overhead is modest.

Although communication is generally not the limiting factor in HPC settings, further reductions are possible through techniques such as update sparsification, model quantization, and client subsampling. While these were not employed in this study, they represent promising directions for scaling FL to even larger deployments.

In summary, the proposed FL configuration demonstrates both computational efficiency and manageable communication costs. These results confirm the feasibility of applying federated anomaly detection in real HPC environments, supporting its integration into production monitoring pipelines without requiring major architectural modifications or additional infrastructure.

5. Conclusion

This work presents the first systematic study of Federated Learning (FL) with LSTM autoencoders for anomaly detection in real-world HPC systems using time-series telemetry. Using data from the Marconi100 Tier-0 supercomputer, we show that FL substantially improves anomaly detection performance across unsupervised, semi-supervised, and supervised settings–without requiring centralized data collection or access to labeled anomalies. Our approach achieves an average F1-score increase from 0.388 to 0.867 and an AUC improvement from 0.334 to 0.808, while reducing the training data requirement by a factor of 15. These gains are supported by statistical significance testing and reinforced through ablation studies.

Beyond raw performance, our analysis highlights the importance of temporal modeling with LSTM autoencoders and the role of advanced aggregation strategies in FL. In particular, FedAdagrad and FedAvgM consistently delivered the best results across different data fractions and learning modes, ensuring robustness even under conditions of extreme data scarcity. These findings emphasize the scalability and efficiency of FL for practical deployment in operational HPC environments.

By enabling decentralized, collaborative training directly on HPC nodes, this research addresses the dual challenges of data privacy and distributed system monitoring. It also paves the way for early fault prediction and adaptive monitoring in next-generation supercomputing facilities.

6. Future work

In future work, we plan to enhance our FL framework by incorporating explainable AI techniques (e.g., attention mechanisms or SHAP) to improve the interpretability of anomaly predictions and assist in root-cause analysis. We also intend to explore personalized federated models that adapt global knowledge to local node behaviors, potentially improving detection in heterogeneous environments.

To address rare or low-frequency faults, we plan to investigate federated transfer learning methods that enable knowledge sharing from data-rich to data-scarce nodes. Furthermore, we aim to optimize communication efficiency through model compression, sparse updates, and asynchronous aggregation to support large-scale deployments.

Expanding the data scope to include multimodal sources such as workload logs and environmental metrics is another avenue we plan to pursue. This will support broader generalization across diverse HPC platforms. We also plan to extend our framework with continual learning capabilities for adapting to system drift without full retraining.

Lastly, we are interested in integrating our models into real-time monitoring pipelines and evaluating their performance under adversarial settings, where we plan to implement robust and fault-tolerant FL strategies to ensure reliability in production-grade HPC environments.

While this study demonstrates the effectiveness of LSTMs for temporal modeling in HPC telemetry, future research would also explore alternative architectures such as GRUs, TCNs, and Transformers. These models capture different temporal dynamics or offer improved efficiency under varying sampling regimes. Incorporating such architectures into the federated setting represents a promising direction for further work.

CRediT authorship contribution statement

Emmen Farooq: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization; **Michela Milano:** Validation, Supervision, Software, Resources, Project administration, Funding acquisition, Formal analysis, Conceptualization; **Andrea Borghesi:** Writing – review & editing, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no competing interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work has been partially supported by European Project HORIZON-CL4-2021-HUMAN-01-AI4Europe (g.a. 101070000) and by European Project HORIZON-EUROHPC-JU-SEANERGYS (g.a. 101177590). The authors would like to thank Huawei for providing the hardware used in this research and for their support.

References

[1] F.R. Albogamy, Federated learning for IOMT-enhanced human activity recognition with hybrid LSTM-GRU networks, Sensors 25 (3) (2025) 907.
[2] N. Sivakumar, A.R. Khan, S. Umar, R.N. Ravikumar, I. Bremnavas, M. Lunagaria, K. Vaghela, G.G. Tejani, S.K. Sharma, A hybrid brain tumor classification using FL with FedAvg and FedProx for privacy and robustness across heterogeneous data sources, IEEE Access 13 (2025), 57705–57719.
[3] P. Radoglou-Grammatikis, P.S. Bouzinis, I. Makris, T. Lagkas, V. Argyriou, G.T. Papadopoulos, P. Fouliras, G. Seritan, P. Sarigiannidis, AI4FIDS: multimodal federated intrusion detection, IEEE Trans. Emerg. Top. Comput. (2025) 1–15.
[4] A. Tedjini, M. Chenine, An Intrusion Detection System Based on Federated Deep Learning, Ph.D. thesis, KASDI MERBAH UNIVERSITY OUARGLA.
[5] A. Borghesi, M. Molan, M. Milano, A. Bartolini, Anomaly detection and anticipation in high performance computing systems, IEEE Trans. Parallel Distrib. Syst. 33 (4) (2021) 739–750.
[6] M. Molan, M.S. Ardebili, J.A. Khan, F. Beneventi, D. Cesarini, A. Borghesi, A. Bartolini, GRAAFE: graph anomaly anticipation framework for exascale HPC systems, Future Gener. Comput. Syst. 160 (2024) 644–653.
[7] G.A. Baumgart, J. Shin, A. Payani, M. Lee, R.R. Kompella, Not all federated learning algorithms are created equal: a performance evaluation study, (2024). arXiv preprint arXiv:2403.17287

[8] A. Bartolini, F. Beneventi, A. Borghesi, D. Cesarini, A. Libri, L. Benini, C. Cavazzoni, Paving the way toward energy-aware and automated datacentre, in: Workshop Proceedings of the 48th International Conference on Parallel Processing, 2019, pp. 1–8.
[9] M. Molan, A. Borghesi, D. Cesarini, L. Benini, A. Bartolini, RUAD: unsupervised anomaly detection in HPC systems, Future Gener. Comput. Syst. 141 (2023) 542–554.
[10] A. Mora, D. Fantini, P. Bellavista, Federated learning algorithms with heterogeneous data distributions: an empirical evaluation, in: 2022 IEEE/ACM 7th Symposium on Edge Computing (SEC), IEEE, 2022, pp. 336–341.
[11] N. Hamdi, Federated learning-based intrusion detection system for internet of things, Int. J. Inf. Secur. 22 (6) (2023) 1937–1948.
[12] E. Farooq, A. Borghesi, A federated learning approach for anomaly detection in high performance computing, in: 2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2023, pp. 496–500.
[13] W. Guo, Y. Wang, X. Chen, P. Jiang, Federated transfer learning for auxiliary classifier generative adversarial networks: framework and industrial application, J. Intell. Manuf. 35 (4) (2024) 1439–1454.
[14] E. Farooq, M. Milano, A. Borghesi, Harnessing federated learning for anomaly detection in supercomputer nodes, Future Gener. Comput. Syst. 161 (2024) 673–685.
[15] E. Farooq, A. Borghesi, LSTM-based unsupervised anomaly detection in high-performance computing: a federated learning approach, in: 2024 IEEE International Conference on Big Data (BigData), IEEE, 2024, pp. 7735–7744.
[16] CINECA, Marconi100 — hpc.cineca.it, 2025, (https://www.hpc.cineca.it/systems/hardware/marconi100/), [Accessed 15-05-2025].
[17] S.M.S. Bukhari, M.H. Zafar, M. Abou Houran, S.K.R. Moosavi, M. Mansoor, M. Muaaz, F. Sanfilippo, Secure and privacy-preserving intrusion detection in wireless sensor networks: federated learning with SCNN-Bi-LSTM for enhanced reliability, Ad Hoc Netw. 155 (2024) 103407.
[18] X. Wang, H. Wang, B. Bhandari, L. Cheng, AI-empowered methods for smart energy consumption: a review of load forecasting, anomaly detection and demand response, Int. J. Precis. Eng. Manuf. Green Technol. 11 (3) (2024) 963–993.
[19] N. Jeffrey, Q. Tan, J.R. Villar, A hybrid methodology for anomaly detection in cyber–physical systems, Neurocomputing 568 (2024) 127068.
[20] S.H. Rafique, A. Abdallah, N.S. Musa, T. Murugan, Machine learning and deep learning techniques for internet of things network anomaly detection–current research trends, Sensors 24 (6) (2024) 1968.
[21] X. Zhu, J. Cherukuri, T. Yuan, Failure and maintenance analysis of supercomputers, in: 2016 Annual Reliability and Maintainability Symposium (RAMS), IEEE, 2016, pp. 1–6.
[22] H. Neau, R. Ansart, C. Baudry, Y. Fournier, N. Mérigoux, C. Koren, J. Laviéville, N. Renon, O. Simonin, HPC challenges and opportunities of industrial-scale reactive fluidized bed simulation using meshes of several billion cells on the route of exascale, Powder Technol. 444 (2024) 120018.
[23] L. Tunini, A. Magrin, G. Rossi, D. Zuliani, Global navigation satellite system (GNSS) time series and velocities about a slowly convergent margin processed on high-performance computing (HPC) clusters: products and robustness evaluation, Earth Syst. Sci. Data 16 (2) (2024) 1083–1106.
[24] B. Aksar, Y. Zhang, E. Ates, B. Schwaller, O. Aaziz, V.J. Leung, J. Brandt, M. Egele, A.K. Coskun, Proctor: a semi-supervised performance anomaly diagnosis framework for production hpc systems, in: High Performance Computing: 36th International Conference, ISC High Performance 2021, Virtual Event, June 24–July 2, 2021, Proceedings 36, Springer, 2021, pp. 195–214.
[25] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, L. Benini, A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems, Eng. Appl. Artif. Intell. 85 (2019) 634–644.
[26] M. Molan, A. Borghesi, L. Benini, A. Bartolini, Analysing supercomputer nodes behaviour with the latent representation of deep learning models, in: European Conference on Parallel Processing, Springer, 2022, pp. 171–185.
[27] P. Zou, A. Li, K. Barker, R. Ge, Detecting anomalous computation with RNNS on GPU-accelerated HPC machines, in: Proceedings of the 49th International Conference on Parallel Processing, 2020, pp. 1–11.
[28] V. Mothukuri, P. Khare, R.M. Parizi, S. Pouriyeh, A. Dehghantanha, G. Srivastava, Federated-learning-based anomaly detection for IoT security attacks, IEEE Internet Things J. 9 (4) (2021) 2545–2554.
[29] J. Pei, K. Zhong, M.A. Jan, J. Li, Personalized federated learning framework for network traffic anomaly detection, Comput. Netw. 209 (2022) 108906.
[30] R.A. Sater, A.B. Hamza, A federated learning approach to anomaly detection in smart buildings, ACM Trans. Internet Things 2 (4) (2021) 1–23.
[31] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, M.S. Hossain, Deep anomaly detection for time-series data in industrial IoT: a communication-efficient on-device federated learning approach, IEEE Internet Things J. 8 (8) (2020) 6348–6358.
[32] M.J. Idrissi, H. Alami, A. El Mahdaouy, A. El Mekki, S. Oualil, Z. Yartaoui, I. Berrada, Fed-ANIDS: federated learning for anomaly-based network intrusion detection systems, Expert Syst. Appl. 234 (2023) 121000.
[33] A. Bartolini, A. Borghesi, A. Libri, F. Beneventi, D. Gregori, S. Tinti, C. Gianfreda, P. Altoè, The DAVIDE big-data-powered fine-grain power and performance monitoring support, in: Proceedings of the 15th ACM International Conference on Computing Frontiers, 2018, pp. 303–308.
[34] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, L. Benini, Anomaly detection using autoencoders in high performance computing systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, 33, 2019, pp. 9428–9433.
[35] A. Borghesi, C. Di Santi, M. Molan, M.S. Ardebili, A. Mauri, M. Guarrasi, D. Galetti, M. Cestari, F. Barchi, L. Benini, et al., M100 Exadata: a data collection campaign on the CINECA's Marconi100 tier-0 supercomputer, Sci. Data 10 (1) (2023) 288.
[36] L.A.N. Laboratory, USRC data sources | los alamos national laboratory — lanl.gov. (https://www.lanl.gov/engage/organizations/aldsct/hpc/usrc/data), [Accessed 11-05-2025].
[37] Grid5000 Consortium, Grid5000 — grid5000.fr, 2025, (https://www.grid5000.fr), [Accessed 11-05-2025].
[38] Google, GitHub - google/cluster-data: borg cluster traces from Google — github.com, 2025, (https://github.com/google/cluster-data), [Accessed 11-05-2025].
[39] The Flower Authors, Flower: a friendly federated AI framework — flower.ai, 2025, (https://flower.ai/), [Accessed 12-05-2025].
[40] S. Kim, K. Choi, H.-S. Choi, B. Lee, S. Yoon, Towards a rigorous evaluation of time-series anomaly detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, 36, 2022, pp. 7194–7201.
[41] R.F. Woolson, Wilcoxon signed-rank test, Encycl. Biostat. 8 (2005).
[42] J. Manokaran, G. Vairavel, Dl-ADS: improved grey wolf optimization enabled AE-LSTM technique for efficient network anomaly detection in internet of thing edge computing, IEEE Access 12 (2024), 75983–76002.
[43] Q. Ji, F. Pan, W. Ding, W. Zhang, The usage of hypothesis testing in identifying anomalies in time series data for cigarette cutting production, IEEE Access 12 (2024), 147516–147526.
[44] S. Haggenmüller, M. Schmitt, E. Krieghoff-Henning, A. Hekler, R.C. Maron, C. Wies, J.S. Utikal, F. Meier, S. Hobelsberger, F.F. Gellrich, et al., Federated learning for decentralized artificial intelligence in melanoma diagnostics, JAMA Dermatol. 160 (3) (2024) 303–311.