ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

COMPUTER SCIENCE AND ENGINEERING

Ciclo 37

Settore Concorsuale: 09/H1 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

Settore Scientifico Disciplinare: ING-INF/05 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI

FEDERATED LEARNING THEORETICAL TOOLS AND PLATFORM DEVELOPMENT

Presentata da: Emmen Farooq

Coordinatore Dottorato

Paola Salomoni

Supervisore

Michela Milano

Co-supervisore

Michele Lombardi

Esame finale anno 2026

Dedication

Dedicated to my wonderful father,to whom I owe everything.

Acknowledgments

I would like to begin by extending my heartfelt gratitude to my supervisor, Prof. Michela Milano, for her exceptional guidance, profound wisdom, and unwavering support throughout my entire PhD journey. Her expertise, along with her insightful feedback, has been absolutely instrumental in shaping and refining this thesis. Beyond her academic mentorship, Prof. Milano's encouragement and guidance have been crucial in helping me navigate the many challenges inherent to research. I am deeply thankful for the academic freedom she allowed me, creating an environment where I could grow both professionally and personally.

I also wish to express my sincere appreciation to my colleague, Dr. Andrea Borghesi, for his invaluable support and collaboration. Dr. Borghesi's readiness to discuss ideas, provide advice, and share his knowledge has been a constant source of enrichment for my work. His contributions have significantly advanced my research, and I am truly grateful for his expertise and generosity in sharing his time and insights.

My profound thanks also go to Huawei for their generous sponsorship of my PhD and for providing the cutting-edge hardware essential to the successful completion of my research. Their support has been a cornerstone in allowing me to work with the latest technology, and I am truly appreciative of the opportunities this has provided me.

I am deeply thankful to my colleagues at Huawei, especially Domenico Pannofino, Dr. Scordino Claudio, and Xing Yiren. Their friendship, technical support, and constructive feedback have been incredibly helpful throughout this journey. The collaborative spirit we shared has not only enriched my research but also contributed significantly to my professional development. I consider myself fortunate to have had the opportunity to work alongside such talented and supportive individuals.

A special thank you goes to my parents, whose unwavering love, encouragement, and sacrifices have always been the foundation of my determination and resilience. Their belief in me has provided a constant source of strength, and I dedicate this achievement to them.

Finally, I would like to express my deepest love and gratitude to my husband. His patience, understanding, and steadfast support throughout both the highs

and lows of this journey have been invaluable. His unwavering belief in me kept me motivated, and his love and emotional support were the grounding force that allowed me to persevere. This thesis would not have been possible without him by my side.

Abstract

High-Performance Computing (HPC) systems play a crucial role in modern scientific and industrial applications, yet their growing complexity introduces challenges in fault detection and system reliability. This thesis explores the application of Federated Learning (FL) and Federated Transfer Learning (FTL) to enhance anomaly detection in HPC nodes. Traditional machine learning (ML) approaches require large labeled datasets, which are often unavailable in HPC environments. To address this, we propose decentralized learning methodologies that leverage data across multiple nodes while preserving data privacy and reducing network overhead.

Our research investigates the effectiveness of unsupervised, semi-supervised, and supervised FL-based anomaly detection models, including LSTM autoencoders and other deep learning techniques. Empirical evaluations on real-world supercomputing systems, such as Marconi100 at CINECA, demonstrate substantial improvements in anomaly detection performance, with F1-score enhancements from 0.31 to 0.84 in semi-supervised learning and a 15-fold reduction in data collection time. Furthermore, FTL enables knowledge transfer from high-performing to low-performing nodes, reducing the need for extensive retraining.

By integrating FL and FTL, this thesis contributes to the advancement of intelligent, scalable, and efficient anomaly detection in HPC systems, facilitating improved system uptime, performance, and reliability.

Keywords: Federated Learning, Federated Transfer Learning, Anomaly Detection, High-Performance Computing, Machine Learning, LSTM, Transfer Learning.

Contents

Abstract v

1 Introduction 1
- 1.1 Context and Motivation . . . 1
- 1.2 Theoretical Background (Simplified Overview) . . . 2
- 1.3 Problem Statement and Research Gaps . . . 3
- 1.4 Research Objectives and Questions . . . 4
- 1.5 Thesis Contributions . . . 5
- 1.6 Thesis Outline . . . 7

2 Background 9
- 2.1 Anomaly Detection: A General Overview . . . 9
- 2.2 Anomaly Detection in High-Performance Computing (HPC) Systems 10
- 2.3 Federated Learning: Principles and Algorithms . . . 12
  - 2.3.1 Motivation for Federated Learning . . . 12
  - 2.3.2 Federated Learning Algorithms and Optimization Strategies 12
- 2.4 Federated Transfer Learning (FTL) . . . 13
  - 2.4.1 Concept and Rationale . . . 13
- 2.5 Summary . . . 14

3 Related Works 18
- 3.1 Anomaly Detection in High-Performance Computing . . . 18
- 3.2 Federated Learning for Anomaly Detection . . . 19
- 3.3 Advanced Aggregation Strategies in FL . . . 20
- 3.4 Transfer Learning and Federated Transfer Learning (FTL) . . . 21
- 3.5 Recent Advances in FL and FTL for Anomaly Detection . . . 21

4 Datasets 23
- 4.1 Introduction . . . 23
- 4.2 Dataset 1 (D1) . . . 24
- 4.3 Dataset 2 (D2) . . . 24

4.4 Comparison of D1 and D2 . . . 25

5 Federated Learning with Autoencoders for Anomaly Detection in HPC Systems 27
5.1 Introduction . . . 28
5.2 Motivation . . . 29
5.3 Methodology . . . 32
5.3.1 Overview . . . 33
5.3.2 Dense Autoencoder for Anomaly Detection . . . 33
5.3.3 LSTM Autoencoder for Anomaly Detection . . . 34
5.3.4 Federated Learning Setup . . . 34
5.3.5 Training Modes . . . 35
5.3.6 FL vs. Non-FL Comparison . . . 35
5.4 Experimental Evaluation . . . 35
5.4.1 Experimental Setup . . . 36
5.4.2 Datasets . . . 36
5.4.3 Implementation . . . 37
5.4.4 Metrics . . . 41
5.5 Experimental Results . . . 42
5.5.1 Results with Dense Autoencoders . . . 42
5.5.2 Results with LSTM Autoencoders . . . 50
5.5.3 Deployment Feasibility Analysis . . . 64

6 Federated Transfer Learning for Anomaly Detection in High-Performance Computing 67
6.1 Introduction . . . 67
6.2 Methodology . . . 71
6.2.1 Local Model Training and Node Ranking . . . 72
6.2.2 Federated Learning with FedAvg . . . 73
6.2.3 Federated Transfer Learning (FTL) . . . 74
6.2.4 Comparison with Random Node Selection . . . 74
6.2.5 Fine-Tuning for Local Personalization . . . 74
6.3 Experimental Evaluation . . . 75
6.3.1 Experimental Setup . . . 75
6.3.2 Datasets . . . 75
6.3.3 Implementation . . . 76
6.3.4 Evaluation Metrics . . . 79
6.3.5 Results . . . 84
6.3.6 Resource Cost Analysis . . . 93
6.4 Chapter Summary . . . 94

7 Discussions **95**

8 Conclusions **101**

106

Bibliography **106**

List of Figures

4.1 Anomaly Distribution across D1 and D2 Datasets 26

5.1 Anomaly Detection for HPC using FL 30
5.2 Comparison of Dense AE and LSTM AE architectures. 38
5.3 Increase in F-score after FL in anomaly detection using D1 across three learning paradigms. 46
5.4 Increase in F1 after Federated Learning on Dataset D2 across semi-supervised, supervised, and unsupervised settings. 47
5.5 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. 55
5.6 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. 56

6.1 Overview of the proposed Federated Transfer Learning pipeline. Left: local autoencoder training on each HPC node. Center: federated learning via FedAvg using top-N nodes. Right: knowledge transfer and fine-tuning on the remaining $100 - N$ nodes. 68
6.2 Knowledge Transfer 69
6.3 Comparison of F1-scores across 20 HPC nodes using decoder-only vs full fine-tuning strategies. While full fine-tuning slightly outperforms decoder-only tuning in most cases, the latter remains preferable in resource-constrained settings. 79
6.4 Grouped bar chart showing F1-score improvement (ΔF) across paradigms and node selection strategies for datasets D1 and D2. . . 88

6.5 Inter-node similarity analysis. (a) PCA projection shows Top-N nodes clustering near the center, suggesting typical behavior. (b) Cosine similarity heatmap indicates high encoder similarity across nodes, validating their transferability. 91
6.6 Resource cost comparison across key stages of the FTL pipeline. Decoder-only fine-tuning is significantly lighter than full fine-tuning in terms of time, memory, and energy. 94

List of Tables

4.1 Summary statistics of Dataset 1 (D1). Values represent average counts per node across the monitoring period. 24
4.2 Summary statistics of Dataset 2 (D2). Values represent average counts per node across the monitoring period. 25
4.3 Comparison of D1 and D2 datasets. 26

5.1 Hyperparameters for Dense AE and LSTM AE models. 39
5.2 Experimental Results Obtained Over Reduced Number of Nodes in Anomaly Detection Using D1 45
5.3 Experimental Results Obtained Over Reduced Number of Nodes in Anomaly Detection Using D2 45
5.4 Experimental Results Obtained Over 70 Nodes in Semi-supervised Anomaly Detection Using D1 45
5.5 Experimental Results Obtained Over 70 Nodes in Semi-supervised Anomaly Detection Using D2 46
5.6 Experimental Results Obtained Over 70 Nodes in Unsupervised Anomaly Detection Using D1 48
5.7 Experimental Results Obtained Over 70 Nodes in Unsupervised Anomaly Detection Using D2 48
5.8 Experimental Results Obtained Over 70 Nodes in Supervised Anomaly Detection Using D1 48
5.9 Experimental Results Obtained Over 70 Nodes in Supervised Anomaly Detection Using D2 49
5.10 Experimental Results Obtained Over 70 Nodes Using Advanced Aggregation Strategies of FL with D1 49
5.11 Experimental Results Obtained Over 70 Nodes Using Advanced Aggregation Strategies of FL with D2. 49
5.12 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 ($p < 0.01$). 52

5.13 F1 and AUC Scores for FL vs. Non-FL in Semi-Supervised Learning under Data Reduction (D1 & D2). Wilcoxon signed-rank test confirms significance ($p < 0.01$). 53
5.14 F1 and AUC Scores for FL vs. Non-FL in Supervised Learning under Data Reduction (D1 & D2). Wilcoxon signed-rank test confirms significance ($p < 0.01$). 54
5.15 Comparison of F1 and AUC scores across different Federated Learning aggregation strategies on Dataset D1 under unsupervised, semi-supervised, and supervised learning paradigms. 60
5.16 Comparison of F1 and AUC scores across different Federated Learning aggregation strategies on Dataset D2 under unsupervised, semi-supervised, and supervised learning paradigms. 61
5.17 Training times (in minutes) for 1000 FL rounds on datasets D1 and D2. Per-node times are reported in parentheses. 65

6.1 Hyperparameters used in autoencoder training 80
6.2 Performance of Semi-Supervised Anomaly Detection with Federated Transfer Learning (FTL) on Datasets D1 and D2 using 100 nodes. **Columns 2–3** show F1-scores before and after FL on selected Top-N or Random-N nodes. **Columns 5–12** report performance before and after applying FTL to the remaining $(100 - N)$ nodes. **Column 13** shows resulting F1 gain (ΔF1). Asterisk (*) indicates statistical significance ($p < 0.05$). 82
6.3 Performance of Unsupervised Anomaly Detection with Federated Transfer Learning (FTL) on Datasets D1 and D2 using 100 nodes. **Columns 2–3** show F1-scores before and after FL on selected Top-N or Random-N nodes. **Columns 5–12** report performance before and after applying FTL to the remaining $(100 - N)$ nodes. **Column 13** shows resulting F1 gain (ΔF1). Asterisk (*) indicates statistical significance ($p < 0.05$). 83
6.4 Performance of Supervised Anomaly Detection with Federated Transfer Learning (FTL) on Datasets D1 and D2 using $1/4^{\text{th}}$ training data. **Columns 2–3** report F1-scores before and after Federated Learning (FL) on selected Top-N or Random-N nodes. **Columns 5–12** show metrics before and after applying FTL to the remaining $(100 - N)$ nodes. **Column 13** shows resulting F1-score gain (ΔF1). Asterisk (*) indicates statistical significance ($p < 0.05$). 84
6.5 Average ΔF1 across paradigms and datasets. 87
6.6 Cohen's d effect sizes for F1-score improvements (ΔF1) across datasets (D1, D2), learning paradigms, node selection strategies, and N values. Effect sizes ≥ 0.8 are considered large. 90

6.7 Estimated Resource Costs of FTL Components (Per Node) 93

Chapter 1

Introduction

1.1 Context and Motivation

High-Performance Computing (HPC) systems are some of the most powerful machines in the world. They are used to study climate change, to understand how proteins work, to design new medicines, and to run huge physics experiments. These are tasks that normal computers would take years to finish, but supercomputers can do them in days or even hours. As the world relies more and more on these "Tier-0" machines, keeping them running smoothly has become a big challenge [14, 12].

Like any complicated machine, a supercomputer can have problems. A part may break, the software may have errors, or the system may be set up in the wrong way. Small issues can make the system slower, while big ones can shut it down completely. When that happens, important projects are delayed, and both money and energy are wasted. Because Tier-0 systems are so expensive to build and run, even a short period of downtime can have serious consequences [17].

In the past, system administrators watched logs and data from the machines to spot these problems. But today's supercomputers produce so much information every day that no human can check it all. The amount of data is simply too large, and errors can easily be missed. This is why new, intelligent methods are needed to detect problems quickly and automatically [11, 15].

Machine learning offers one possible answer. These techniques can learn what "normal" behavior looks like and raise an alarm when something unusual happens.

Still, traditional approaches are not always practical. They usually need large amounts of labeled data, expect every part of the system to behave the same, and often collect all the data in one place. For modern supercomputers, moving all the data to a single location is too difficult and costly [56, 12].

Federated learning (FL) provides a new way forward. Instead of moving the data, FL allows each machine or node to train a model locally and only share what it has learned. These updates are combined to build a stronger model that benefits from the knowledge of all nodes, without the need to move massive amounts of data [69, 4]. Going further, federated transfer learning (FTL) makes it possible to take models trained on some nodes and adapt them to new ones, even if the system has never seen those nodes before [65].

This thesis builds on these ideas. It explores how machine learning, federated learning, and transfer learning can work together to make anomaly detection in supercomputers more reliable. The goal is to create methods that not only work well in theory but are also useful in real, large-scale systems. By doing so, this work aims to reduce downtime, save costs, and support faster progress in science and technology.

1.2 Theoretical Background (Simplified Overview)

Three pillars support the research in this thesis:

- **Anomaly Detection:** Methods that distinguish between normal and abnormal behavior in data or system activity. These approaches help identify faults, unusual patterns, or failures before they become critical [77]. They can be:
  - *Unsupervised* – learning what "normal" looks like without needing labeled anomalies, useful when abnormal cases are unknown.
  - *Semi-supervised* – learning mostly from normal data, with a few examples of anomalies to guide the model.
  - *Supervised* – learning from fully labeled data (both normal and abnormal), which often gives the best performance but is rarely available in

practice due to the cost and difficulty of labeling anomalies.

- **Federated Learning (FL):** A way to train models collaboratively across many nodes, without centralizing all the data in one place. Each node trains locally and shares only model updates, which are combined to build a stronger global model. In HPC, its main value is not privacy (since all nodes belong to the same system) but efficiency—faster training, reduced communication overhead, and better accuracy through shared knowledge across nodes [40].

- **Transfer Learning (TL):** A technique for reusing knowledge learned in one setting to adapt to another. Instead of training a model from scratch every time, existing knowledge is transferred to new but related tasks. In HPC, this allows a model trained on some nodes to quickly adapt when new nodes join, or when system conditions change, saving both time and resources [67].

- **Federated Transfer Learning (FTL):** A combination of FL and TL, where models are trained collaboratively across distributed nodes and the learned knowledge is adapted to new or unseen nodes. This makes it possible to generalize detection capabilities beyond the nodes used in training, an essential step for real-world HPC systems where node configurations and workloads often evolve over time [9].

1.3 Problem Statement and Research Gaps

Machine learning has already been used for anomaly detection in High-Performance Computing (HPC) systems. However, several important gaps remain that limit its effectiveness:

1. **Limited generalization across nodes:** Most approaches are trained on one node at a time and cannot easily share knowledge with other nodes, even though faults often appear in similar ways across the system [7].

2. **Neglect of temporal patterns:** Early methods looked only at short windows of data and missed the longer-term trends in telemetry that signal slowly developing problems [15].

3. **Lack of validation of Federated Learning (FL) and Federated Transfer Learning (FTL):** Although both approaches have been widely used in other domains, they had not been properly tested in HPC, leaving open questions about their reliability and usefulness in real supercomputer settings [30].

This thesis addresses these gaps by developing federated approaches, applying models that capture time dependencies, and validating federated transfer learning in a real Tier-0 supercomputer.

1.4 Research Objectives and Questions

This thesis pursues three main objectives, each of which addresses a specific aspect of applying federated learning to anomaly detection in High-Performance Computing (HPC) systems. For clarity, each objective is linked with a guiding research question.

- **Objective 1:** To test whether federated learning improves anomaly detection across heterogeneous nodes compared with central or per-node training. The aim is to see if collaboration among nodes, without sharing raw data, can lead to better detection accuracy while also handling differences in node behavior and workloads. This also examines whether federated learning can reduce communication costs and increase efficiency compared with traditional approaches.

- **Objective 2:** To evaluate whether Long Short-Term Memory (LSTM) autoencoders within federated learning can better capture time-series patterns and improve detection. Since HPC telemetry data is sequential and often complex, LSTM-based models are well suited for identifying unusual patterns. This objective focuses on whether combining the strengths of LSTM autoencoders with the collaborative nature of federated learning produces models that are more accurate and robust in practice.

- **Objective 3:** To determine if federated transfer learning enables generalization to unseen nodes under realistic conditions. In HPC environments, new

nodes may join the system or some nodes may behave differently from those seen during training. This objective investigates whether models trained with federated transfer learning can be adapted to these unseen nodes with limited additional data, ensuring that the system remains reliable and scalable in real-world scenarios.

From these, three guiding research questions follow:

1. How can federated learning make anomaly detection in HPC systems more accurate and efficient? This question looks at whether combining what different computers learn, without moving their data, can give better results. It also asks if this method can save time and resources compared to traditional ways.

2. How do LSTM autoencoders help in federated anomaly detection? This question focuses on special models called LSTM autoencoders that can understand data changing over time. It asks if using these models in federated learning makes it easier to catch unusual patterns in the data.

3. Can federated transfer learning help detect anomalies on new nodes, and which methods work best? This question looks at whether models can be adapted to computers they have never seen before. It asks if federated transfer learning makes this possible and which approaches, such as fine-tuning or partial training, give the best results.

1.5 Thesis Contributions

The main contributions of this thesis are organized around three published studies. Each study builds on the previous one, gradually extending the scope and impact of federated approaches for anomaly detection in High-Performance Computing (HPC) systems.

1. **Federated Learning for HPC Anomaly Detection:**

- Provided the first systematic use of Federated Learning (FL) for anomaly detection in supercomputers, an area where it had not been applied before.
- Demonstrated that FL improves detection accuracy across different learning settings: supervised (with full labels), semi-supervised (with partial labels), and unsupervised (with no labels).
- Showed that FL reduces the need for large training datasets. Instead of requiring months of data collection, useful models could be trained with much smaller datasets, cutting training times down to weeks.

2. **Federated LSTM Autoencoders:**

   - Extended FL by introducing LSTM autoencoders, which are well-suited for sequential telemetry data because they can capture time-based patterns and dependencies in system behavior.
   - Explored and tested multiple advanced aggregation strategies (such as FedAvg, FedProx, and FedAdam) to combine models more effectively across nodes.
   - Provided evidence of statistically significant improvements in both accuracy and robustness of anomaly detection, even when only very small portions of the training data were available. This highlighted the strength of FL in resource-limited settings.

3. **Federated Transfer Learning in Production:**

   - Delivered the first real validation of Federated Transfer Learning (FTL) in a production-scale Tier-0 supercomputing environment.
   - Demonstrated that federated models can be fine-tuned to generalize effectively to unseen nodes, making them adaptable when new nodes are added or when system configurations change.
   - Compared different node selection strategies for training and showed that careful selection can lead to substantial improvements, achieving up to +0.50 F1-score gains on previously unseen nodes.

Together, these contributions move anomaly detection from isolated, per-node methods to collaborative, adaptive, and production-ready solutions.

1.6 Thesis Outline

The thesis is organized as follows:

- **Chapter 2 - Background** This chapter introduces the theoretical foundations that underpin the research. It explains the core concepts of anomaly detection, federated learning (FL), and transfer learning (TL) in simple terms, ensuring that readers who may not be specialists in these areas can understand the main ideas. It also describes the three common approaches to anomaly detection (unsupervised, semi-supervised, and supervised), setting the stage for how they are later applied in HPC systems.

- **Chapter 3 - Related Works** This chapter reviews the most relevant studies in the field. It covers previous research on anomaly detection in High-Performance Computing (HPC) as well as work on federated and transfer learning in other domains such as IoT and healthcare. By doing so, it highlights the current state of knowledge, identifies what has already been achieved, and clarifies where gaps remain—gaps that this thesis aims to fill.

- **Chapter 4 - Datasets** This chapter presents the real telemetry datasets used in the experiments. It explains how the data was collected from HPC systems, what kinds of features were included, and how anomalies were labeled. Since data quality is critical for anomaly detection, the chapter also discusses challenges such as imbalanced data and large-scale storage, giving readers a clear view of the foundation on which the experimental work is built.

- **Chapter 5 - Federated Learning with Autoencoders for Anomaly Detection in HPC Systems** This chapter presents the first two contributions of the thesis. It begins by introducing how federated learning was applied to anomaly detection in supercomputer nodes, showing that collaborative training can outperform isolated, per-node models. It then extends the

approach by integrating LSTM autoencoders to capture time dependencies in HPC telemetry. The chapter also provides a systematic evaluation of different federated optimization strategies (FedAvg, FedProx, FedAdam, etc.) and demonstrates their impact on accuracy and robustness across supervised, semi-supervised, and unsupervised anomaly detection settings.

- **Chapter 6 - Federated Transfer Learning for Anomaly Detection in High-Performance Computing** This chapter introduces the idea of federated transfer learning (FTL) and presents the third major contribution of the thesis. It explains how models trained collaboratively on a set of nodes can be fine-tuned to work on previously unseen nodes. The results show that FTL improves generalization to new nodes and provides practical evidence of its usefulness in a real Tier-0 supercomputer environment.

- **Chapter 7 - Discussions** This chapter brings together the insights from all three contributions. It reflects on the overall findings, explaining what they mean in practical terms for HPC system operators and researchers. Strengths and limitations of the approaches are discussed openly, and the chapter considers broader implications for scalability, generalization, and the adoption of federated learning in HPC.

- **Chapter 8 - Conclusions** This chapter summarizes the main contributions of the thesis, revisits the research questions, and shows how they were answered. It also points to possible directions for future work, such as exploring other federated learning strategies, extending transfer learning to different HPC architectures, and integrating additional privacy-preserving techniques. The chapter concludes by highlighting how the thesis advances both the theory and practice of anomaly detection in HPC.

Chapter 2

Background

This chapter establishes the theoretical and methodological foundations for this thesis. It first introduces anomaly detection, its main paradigms, and typical techniques, then examines the specific challenges of applying anomaly detection in High-Performance Computing (HPC) systems [17]. It proceeds to outline the principles of Federated Learning (FL), highlighting core algorithms and optimization methods for decentralized training, and concludes with Federated Transfer Learning (FTL), which integrates FL with transfer learning to enable scalable, privacy-preserving model adaptation [30]. These sections provide the essential context for the frameworks and experiments developed in this work.

2.1 Anomaly Detection: A General Overview

Anomaly detection refers to the identification of data patterns or behaviors that significantly deviate from established norms [70]. These deviations, often called anomalies, outliers, or novelties, can signal critical issues such as fraudulent transactions, system malfunctions, or cybersecurity breaches. Given its importance, anomaly detection spans diverse application domains including finance, industrial IoT, health monitoring, and large-scale IT infrastructures [27].

One of the fundamental challenges in anomaly detection is the severe class imbalance: anomalies are rare by nature, often comprising less than 1% of all observed data [79]. This imbalance complicates the application of traditional

machine learning models, which typically assume balanced class distributions.

As a result, anomaly detection methods are generally categorized into three paradigms:

- **Supervised methods**: These rely on historical datasets labeled with both normal and anomalous instances. They often achieve high predictive accuracy but suffer from limited applicability due to the difficulty of obtaining representative labeled anomalies [34].
- **Unsupervised methods**: These assume that anomalies differ markedly from the majority of data and learn to characterize "normal" behavior. Instances that diverge substantially are flagged as anomalies [56]. This approach is valuable when labeled data is scarce.
- **Semi-supervised methods**: Typically trained on abundant normal data and a small set of labeled anomalies, these models balance the strength of supervised learning with the practical realities of data collection [10, 12].

A broad range of techniques exists to tackle anomaly detection, from statistical thresholding and clustering to nearest-neighbor methods and sophisticated machine learning pipelines. In recent years, deep learning models such as autoencoders [29], LSTM networks [29], and transformers have gained prominence. These models excel at capturing complex, nonlinear temporal patterns in data, making them particularly effective for detecting subtle or evolving anomalies in dynamic systems [54, 56].

2.2 Anomaly Detection in High-Performance Computing (HPC) Systems

High-Performance Computing (HPC) systems constitute critical infrastructure for modern scientific discovery and engineering applications [12]. Typical HPC environments include thousands of interconnected compute nodes, often equipped with diverse hardware accelerators such as GPUs and NPUs, working collaboratively

on parallel workloads. These infrastructures enable breakthroughs in fields ranging from bioinformatics to climate modeling [10].

The scale and complexity of HPC systems, however, also introduce substantial operational risks. Hardware failures, software bugs, configuration drifts, or even subtle performance regressions can propagate through interconnected workflows, leading to incomplete jobs, resource wastage, or extended downtime. Given the high costs and performance demands, timely and accurate anomaly detection in HPC environments is indispensable [16, 13].

Traditional monitoring tools like Nagios or Ganglia, as well as rule-based threshold alarms, provide essential visibility but often fall short in anticipating novel or complex failure modes [14]. These tools are inherently reactive and limited in handling nuanced deviations that precede failures.

In response, data-driven approaches leveraging Machine Learning (ML) and Deep Learning (DL) have become increasingly prominent. Recent studies have shown that:

- **Unsupervised autoencoders** can learn to reconstruct normal node behavior, flagging high reconstruction errors as anomalies [56].
- **Semi-supervised models** incorporating small sets of labeled faults significantly improve sensitivity to subtle issues [10].
- **Time-series approaches**, particularly LSTM autoencoders, capture temporal dependencies in node telemetry, outperforming static models by identifying gradual drifts and periodic anomalies that traditional systems might miss [29].

With the evolution toward exascale systems and the increasing heterogeneity of hardware components, integrating machine learning for proactive, intelligent anomaly detection is not merely beneficial but increasingly essential [14].

2.3 Federated Learning: Principles and Algorithms

2.3.1 Motivation for Federated Learning

Machine learning models traditionally rely on centralized data collection: logs and telemetry are aggregated into a central repository for training [85]. However, in large HPC centers, this poses critical challenges [28]:

- **Data locality and privacy**: Regulations and internal policies may prohibit aggregating sensitive telemetry data across administrative or departmental boundaries [2].
- **Communication overhead**: Moving large-scale telemetry data from thousands of nodes to a central location consumes significant bandwidth and storage resources [58].

Federated Learning (FL) addresses these challenges by decentralizing model training. Each node (or a local cluster of nodes) trains a model on its own data and sends only model updates, such as weight gradients, to a central aggregator [25]. This aggregator updates a global model, which is then distributed back to the nodes for another round of local training.

This decentralized approach preserves data privacy, reduces communication costs, and accommodates the heterogeneous and non-IID (non-identically distributed) nature of telemetry data across nodes [82].

2.3.2 Federated Learning Algorithms and Optimization Strategies

Several optimization algorithms have been developed to improve federated learning under conditions typical in HPC environments:

(1) Federated Averaging (FedAvg) FedAvg is the canonical FL algorithm. Clients train locally on their private datasets and periodically send updated pa-

rameters to a central server, which computes a weighted average [47]. Details are in Algo 1:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^t$$

where w_k^t represents the local model at client k with n_k samples, and n is the global total.

(2) FedAvgM FedAvgM introduces momentum at the server level to smooth updates, improving convergence stability especially in non-IID scenarios [44]. Details are in Algo 2:

(3) FedProx FedProx incorporates a proximal term into local objectives to restrict client models from drifting too far from the global model [23], as explained in Algo 3:

$$\min_w \ell_k(w) + \frac{\mu}{2}\|w - w_t\|^2$$

where $\ell_k(w)$ is the local loss and μ controls the penalty strength. This is particularly effective in systems with heterogeneous telemetry patterns.

(4) Adaptive optimizers: FedAdam, FedYogi, FedAdagrad These variants adapt popular gradient-based optimizers to FL by adjusting learning rates per parameter based on accumulated gradient statistics, improving convergence and robustness under non-uniform data [86]. The details are elaborated in Algo 4.

These strategies have proven critical for HPC environments, where individual nodes can differ significantly in workload profiles and sensor distributions.

2.4 Federated Transfer Learning (FTL)

2.4.1 Concept and Rationale

While Federated Learning excels at collaboratively training models without centralizing data, its benefits are typically confined to the set of participating nodes [65].

In practical HPC deployments, infrastructure evolves—new nodes are added or old nodes are repurposed—necessitating mechanisms for extending learned knowledge.

Federated Transfer Learning (FTL) bridges this gap by integrating FL with transfer learning principles [9]. The typical FTL workflow involves:

1. **Federated Pretraining**: An initial global model is trained collaboratively across a subset of nodes using FL [84].
2. **Local Fine-Tuning**: This model is then deployed to new or previously uninvolved nodes, where it is further refined on local data [84].

FTL combines the scalability and privacy preservation of FL with the adaptability of transfer learning. It enables:

- **Personalized adaptation**: Each node can tailor the model to its specific telemetry and workload patterns [1].
- **Incremental scalability**: New nodes can be onboarded without restarting the entire federated process [33].
- **Strict privacy**: Raw data remains local at all stages [83].

This hybrid approach is thus ideally suited for dynamic HPC systems where operational demands and hardware evolve continuously.

2.5 Summary

This chapter provided a comprehensive background on the foundational concepts underpinning this thesis:

- It detailed anomaly detection paradigms and the challenges posed by highly imbalanced data.
- It explored the necessity of anomaly detection in HPC systems, emphasizing the limitations of traditional approaches and the promise of ML/DL methods.

- It explained Federated Learning, discussing core algorithms like FedAvg, FedAvgM, FedProx, and adaptive optimizers such as FedAdam, FedYogi, and FedAdagrad.

- It introduced Federated Transfer Learning (FTL) as a robust solution for extending federated insights to new HPC nodes while maintaining data privacy.

The subsequent chapters will build upon these principles to present novel frameworks and empirical validations demonstrating how FL and FTL can revolutionize anomaly detection in production-scale HPC environments.

Algorithm 1: FedAvg: Federated Averaging Algorithm

Server executes:
  Initialize global model w_0
  for *each round* $t = 0, 1, 2, \dots$ **do**
    $c \leftarrow \max(C \times K, 1)$
    $S_t \leftarrow$ (random subset of c clients)
    foreach *client* $k \in S_t$ ***in parallel*** **do**
      $\Delta_t^k \leftarrow \text{ClientUpdate}(k, w_t)$
    end
    $\Delta_t \leftarrow \sum_{k \in S_t} \frac{n_k}{n} \Delta_t^k$
    $w_{t+1} \leftarrow w_t + \Delta_t$
  end
  ClientUpdate(k, w_t)**:**
    $w \leftarrow w_t$
    $\mathcal{B} \leftarrow$ (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**
        $w \leftarrow w - \eta \nabla \ell(w; b)$
      end
    end
    $\Delta \leftarrow w - w_t$
    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 $t = 0, 1, 2, \ldots$ **do**
    Select a random subset S_t of K clients
    for each client $k \in S_t$ **in parallel do**
      $\Delta_t^k \leftarrow \texttt{ClientUpdate}(k, w_t)$
    Aggregate updates: $\Delta_t \leftarrow \sum_{k \in S_t} \frac{n_k}{n} \Delta_t^k$
    Update server momentum: $v_{t+1} \leftarrow \mu v_t + \Delta_t$
    Update global model: $w_{t+1} \leftarrow w_t + \eta_s v_{t+1}$
ClientUpdate(k, w_t):
  $w \leftarrow w_t$
  Split local data $\mathcal{D}_k$ 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 $\Delta^k = w - w_t$

Algorithm 3: FedProx: Federated Learning with Proximal Gradient Descent

Server executes:
  Initialize global model w_0
  for *each round* $t = 0, 1, 2, \ldots$ **do**
    $S_t \leftarrow$ (random subset of K clients)
    foreach *client* $k \in S_t$ ***in parallel*** **do**
      $\Delta_t^k \leftarrow \text{ClientUpdate}(k, w_t)$
    end
    $\Delta_t \leftarrow \sum_{k \in S_t} \frac{n_k}{n} \Delta_t^k$
    $w_{t+1} \leftarrow w_t + \eta_s \cdot \Delta_t$
  end
  ClientUpdate(k, w_t)**:**
    $w \leftarrow w_t$
    $\mathcal{B} \leftarrow$ (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**
        $g \leftarrow \nabla \ell(w; b) + \mu(w - w_t)$
        $w \leftarrow w - \eta \cdot g$
      end
    end
    $\Delta \leftarrow w - w_t$
    return Δ to server

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* $t = 0, 1, 2, \ldots$ **do**
    $c \leftarrow \max(C \times K, 1)$
    $S_t \leftarrow$ (random set of c clients)
    foreach *client* $k \in S_t$ ***in parallel*** **do**
      $\Delta_t^k \leftarrow \text{ClientUpdate}(k, w_t)$
    end
    $\Delta_t \leftarrow \sum_{k \in S_t} \frac{n_k}{n} \Delta_t^k$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
    // **FedAdagrad**
    $v_t^{\text{Adagrad}} \leftarrow v_{t-1} + (\Delta_t)^2$
    // **FedYogi**
    $v_t^{\text{Yogi}} \leftarrow v_{t-1} - (1 - \beta_2) \cdot (\Delta_t)^2 \cdot \text{sign}(v_{t-1} - (\Delta_t)^2)$
    // **FedAdam**
    $v_t^{\text{Adam}} \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) \cdot (\Delta_t)^2$
    $w_{t+1} \leftarrow w_t + \eta_s \cdot \frac{m_t}{\sqrt{v_t} + \tau}$
  end
  ClientUpdate(k, w_t):
    $w \leftarrow w_t$
    $\mathcal{B} \leftarrow$ (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
    $\Delta \leftarrow w - w_t$
    return Δ to server

Chapter 3

Related Works

Anomaly detection is a key research area that spans multiple domains, including credit card fraud prevention [18], energy systems [46, 78], healthcare [68], IT infrastructure [39, 49], and the Internet of Things (IoT) [81, 66]. These examples demonstrate its broad applicability, but the requirements of High-Performance Computing (HPC) environments are unique. HPC systems operate at extreme scale, demand very high reliability, and involve heterogeneous hardware, which makes anomaly detection particularly challenging. This thesis therefore concentrates on anomaly detection in HPC, with a specific focus on Federated Learning (FL) and Federated Transfer Learning (FTL). The following sections review related work across anomaly detection in HPC, federated and transfer learning paradigms, aggregation strategies, and recent advances, while also highlighting their limitations when applied to operational HPC infrastructures.

3.1 Anomaly Detection in High-Performance Computing

The reliability and efficiency of HPC systems depend critically on detecting anomalies such as node faults, hardware degradation, and unexpected behaviors that can cause job failures and resource wastage [6, 87]. As systems evolve toward Exascale computing, anomaly detection becomes even more difficult because of the growing scale, heterogeneity, and imbalance in operational data [42, 59, 77].

Initial approaches relied on traditional machine learning techniques. For example, Bayesian Gaussian Mixture Models were used to characterize component behavior [62], while decision trees trained on handcrafted features supported online anomaly detection [76]. Supervised methods based on telemetry data have also been investigated [43, 8], although they are fundamentally limited by the scarcity of labeled anomalies [26]. This makes purely supervised approaches difficult to scale in real HPC environments.

To overcome these labeling challenges, semi-supervised and unsupervised methods have gained prominence. Autoencoders trained on normal operational data have been shown to identify deviations such as abnormal CPU frequency patterns [12], while LSTM autoencoders have provided further improvements by modeling temporal dependencies in sensor data [56, 54, 88]. Despite these successes, the majority of models remain tailored to specific system configurations and can generalize poorly when applied across heterogeneous nodes or evolving hardware. More recent approaches have therefore incorporated domain adaptation [71] and self-supervised learning [80] in order to strengthen generalization and resilience across systems [16, 61]. However, almost all of these works assume centralized data collection, which raises concerns around scalability, privacy, and long-term applicability in production HPC infrastructures.

3.2 Federated Learning for Anomaly Detection

Federated Learning (FL) enables distributed model training without centralizing raw data, making it particularly attractive in large-scale and privacy-sensitive environments. It has been successfully applied in IoT [60], healthcare [57], and smart buildings [69]. For anomaly detection, FL has supported LSTM-based methods in network traffic monitoring [63], CNNs for intrusion detection [35], and autoencoders in energy and industrial systems [75, 37]. These studies highlight the flexibility of FL, but they also rely on assumptions of balanced and relatively homogeneous data distributions, which do not reflect the realities of HPC workloads.

In addition, FL frameworks have been developed for log-based anomaly detection and for integrating privacy-enhancing techniques such as differential privacy and

blockchain [21, 50]. Several studies have also explored advanced aggregation techniques to improve scalability and accuracy in distributed anomaly detection [36, 64].

Within HPC, however, the application of FL is still at an early stage. To the best of our knowledge, the earliest and most substantial contributions in this area have come from our research, where FL-based LSTM autoencoders were used for anomaly detection across distributed nodes [29, 30]. These studies provided the first empirical demonstration that privacy-aware, decentralized anomaly detection can be scaled to production-grade HPC systems.

3.3 Advanced Aggregation Strategies in FL

The choice of aggregation strategy has a significant influence on the performance of FL, especially in environments with non-IID data or constrained resources such as HPC. Optimizers such as FedAvgM, FedProx, FedAdagrad, FedYogi, and FedAdam have each been shown to improve robustness and convergence in different contexts [24, 36, 38]. For example, FedAdam and FedYogi achieved strong performance in industrial fault classification tasks [64], FL-based intrusion detection in UAVs reached F1-scores of up to 0.904 using FedYogi with stacked autoencoders [22], and lightweight FL frameworks reduced communication overhead while improving efficiency [51, 37].

Although these results illustrate the effectiveness of advanced aggregation strategies, they are drawn primarily from domains with smaller scale and less heterogeneity than HPC. Their behavior in large, distributed supercomputing infrastructures—characterized by thousands of nodes, severe communication bottlenecks, and diverse workload patterns—remains largely unexplored. This thesis builds on these ideas by systematically evaluating several aggregation strategies (FedAvgM, FedYogi, FedAdam, FedAdagrad, and FedProx) in combination with LSTM autoencoders for anomaly detection in HPC.

3.4 Transfer Learning and Federated Transfer Learning (FTL)

Transfer Learning (TL) has been widely adopted as a way to reuse knowledge across tasks or domains, thereby reducing the need for large volumes of labeled data. In anomaly detection, it has been used to adapt models across devices and systems [65]. However, while TL alleviates labeling costs, it does not address the privacy and decentralization requirements of modern distributed environments, including HPC.

Federated Transfer Learning (FTL) combines the privacy-preserving model training of FL with the cross-domain adaptability of TL. This makes it possible to train models collaboratively while also transferring knowledge to unseen nodes or populations. FTL has been applied in image classification [67], industrial fault diagnosis [65], structural monitoring and anomaly detection [61], and healthcare for personalized model transfer [9].

Despite these developments, FTL has remained largely unexplored in HPC. To the best of our knowledge, the first applications of FTL for anomaly detection in HPC have also been introduced through our research [28, 29, 30], where we extended FL-based autoencoder models with transfer learning capabilities tailored for HPC nodes.

3.5 Recent Advances in FL and FTL for Anomaly Detection

Recent years have seen several methodological advances in FL and FTL that extend their applicability to anomaly detection. Examples include FedDyMem, a dynamic memory-based FL method for anomaly detection [19]; FedKO, which leverages Koopman-reservoir representations for spatiotemporal modeling [72]; federated thresholding with summary statistics [45]; adaptive client selection for reducing communication costs [51]; and domain adaptation with unified log feature representation [71].

Although these contributions expand the methodological toolkit for FL and

FTL, none have yet been validated in large-scale HPC infrastructures. This lack of validation reinforces the novelty of our contributions, which represent the first empirically validated applications of both Federated Learning and Federated Transfer Learning for time-series anomaly detection in operational HPC systems. Our framework integrates personalized fine-tuning to extend knowledge to non-participating nodes, includes evaluation under supervised, semi-supervised, and unsupervised paradigms, and compares performance across both Top-N and Random-N node selection strategies.

In summary, prior research has made important progress in anomaly detection, federated learning, and transfer learning, but their direct application to HPC has been limited. The earliest and most substantial explorations of FL and FTL in HPC anomaly detection have been introduced through our work, which this thesis extends into a broader framework for decentralized, privacy-preserving anomaly detection in production-scale supercomputing environments.

Chapter 4

Datasets

4.1 Introduction

This chapter elaborates on the datasets employed in our research. Both datasets originate from the real Tier-0 production supercomputer *Marconi100*, hosted at CINECA[1]. They provide a realistic and operational context for anomaly detection experiments. A key characteristic of these datasets is that anomalies are not artificially injected or simulated, but rather correspond to genuine faulty conditions and anomalous behaviors that were identified and addressed by system administrators during regular operations.

Our anomaly detection focuses on the compute node level. We deliberately exclude systemic anomalies or irregularities stemming from user-submitted workloads. Accordingly, only hardware-level monitoring data is utilized. This includes sensor-based measurements such as power consumption, core temperature, clock frequency, and other metrics that directly reflect the physical and operational state of computing units. Information related to application-level workload is intentionally excluded.

Both datasets consist exclusively of this low-level hardware monitoring data. In addition to raw sensor readings, they also include node status information maintained by system administrators for diagnostic purposes. This combination provides a robust foundation for modeling and detecting anomalies rooted in

[1] https://www.cineca.it/en/hardware/marconi100

hardware behavior within high-performance computing (HPC) environments.

4.2 Dataset 1 (D1)

The first dataset was collected from the Marconi100 system located at CINECA Bologna. The system comprises 980 nodes, each equipped with:

- 4 Nvidia Volta V100 GPUs,
- two 16-core IBM POWER9 AC922 processors at 3.1GHz,
- 256 GB RAM,
- achieving a peak performance of 32 PFlops/s.

The dataset contains 462 hardware-level features per node, including measurements such as CPU and GPU power consumption, fan speed, clock frequency, and other sensor-derived values. Metadata concerning system availability, operational status, and job scheduling is also included.

Status labels in D1 are generated using *Nagios*[2], a monitoring system employed by administrators to flag anomalous nodes, often based on user reports. Labels are assigned every 15 minutes, which defines the aggregation interval for the dataset (supported by prior work [56]). Temporal context is incorporated by computing the mean and standard deviation of sensor values within each 15-minute window. Labels indicate the health of the node: 1 = healthy, 0 = anomalous.

Table 4.1: Summary statistics of Dataset 1 (D1). Values represent average counts per node across the monitoring period.

# Normal Points	# Anomalous Points	Total Points	Percentage of Anomalies (%)
12650.25	391.25	13041.49	3.00

4.3 Dataset 2 (D2)

The second dataset corresponds to the same underlying machine (with the same nodes) but with an enhanced monitoring configuration. The infrastructure used to

[2] https://www.nagios.org/

collect D2 was enriched with additional sensors (physical and software probes) to better characterize the system's behavior. Consequently, D2 contains 573 features per node, compared to 462 in D1.

The annotation mechanism was also refined. Labels were improved through direct involvement of system administrators, who focused on marking only relevant anomalies. As a result, the fraction of anomalies was reduced from 3.0% in D1 to 1.7% in D2, as shown in Figure 4.1. Many anomalies previously flagged in D1 were deemed inconsequential and reclassified as normal. While this produces a cleaner dataset, it also increases the difficulty of properly characterizing anomalies, as the definition becomes stricter and more nuanced.

Table 4.2: Summary statistics of Dataset 2 (D2). Values represent average counts per node across the monitoring period.

# Normal Points	# Anomalous Points	Total Points	Percentage of Anomalies (%)
14200.75	241.41	14442.16	1.70

The D2 dataset is publicly available on Zenodo[3], ensuring accessibility and reproducibility for future research.

4.4 Comparison of D1 and D2

Table 4.3 compares both datasets directly. D2 not only contains more features per node (573 vs. 462) but also benefits from improved labeling. However, this reduction in anomaly proportion increases class imbalance, making the anomaly detection task more challenging.

[3] https://zenodo.org/records/7589131

Table 4.3: Comparison of D1 and D2 datasets.

Characteristic	D1	D2
# Features per Node	462	573
Labeling Method	Nagios-based, user reports	Expert-driven refinement
% Anomalies	3.00%	1.70%
Avg. # Normal Points	12650.25	14200.75
Avg. # Anomalous Points	391.25	241.41
Avg. Total Points	13041.49	14442.16
Availability	Internal (CINECA)	Publicly available (Zenodo)

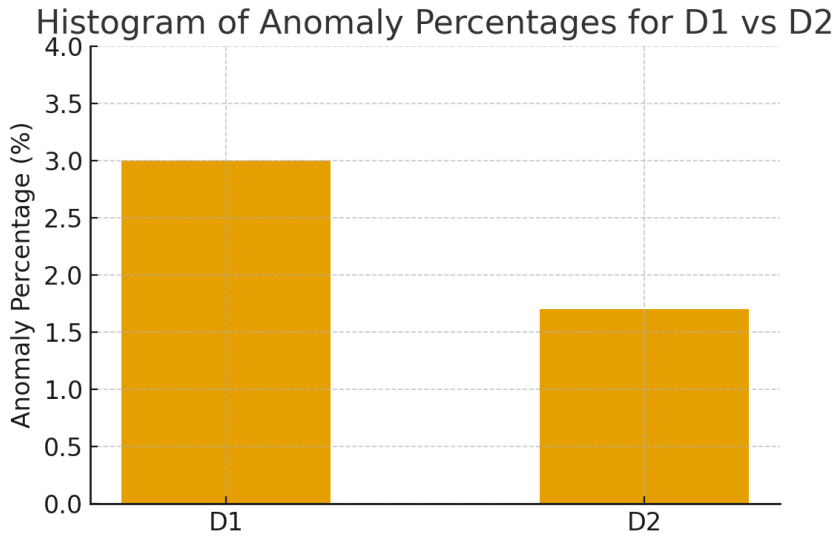


Figure 4.1: Anomaly Distribution across D1 and D2 Datasets

Chapter 5

Federated Learning with Autoencoders for Anomaly Detection in HPC Systems

This chapter is based on the publications:

- **Farooq, Emmen**, Michela Milano, and Andrea Borghesi. "Federated LSTM Autoencoders for Time Series Anomaly Detection in Production-Scale HPC Systems." Knowledge-Based Systems (2025): 115043.
- **Farooq, Emmen**, and Andrea Borghesi. "LSTM-Based Unsupervised Anomaly Detection in High-Performance Computing: A Federated Learning Approach." 2024 IEEE International Conference on Big Data (BigData). IEEE, 2024.
- **Farooq, Emmen**, Michela Milano, and Andrea Borghesi. "Harnessing federated learning for anomaly detection in supercomputer nodes." Future Generation Computer Systems 161 (2024): 673-685.
- **Farooq, Emmen**, and Andrea Borghesi. "A federated learning approach for anomaly detection in high performance computing." 2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2023.

5.1 Introduction

High-Performance Computing (HPC) systems are among the most powerful machines in the world, enabling breakthroughs in climate modeling, molecular biology, and large-scale physics simulations. These systems consist of thousands of interconnected nodes and generate massive amounts of telemetry data every day [55]. Keeping them reliable is a constant challenge, as even small anomalies at the node level can reduce performance or cause costly downtime. Traditional approaches rely on manual monitoring and system logs, which quickly become unmanageable at modern scale [11].

Machine Learning (ML) has emerged as a natural solution to this problem. By learning patterns of normal behavior from monitoring data, ML models can automatically flag unusual conditions that may signal faults. However, three main challenges remain. First, HPC nodes are heterogeneous: a model trained on one node often performs poorly when applied to another [6]. Second, faults in HPC often evolve over time, requiring models that can capture temporal patterns. Third, gathering and labeling enough training data is slow and expensive, delaying the deployment of reliable anomaly detectors [12].

Federated Learning (FL) offers a promising way forward. Instead of training models in isolation, each node trains locally on its own data and then contributes updates to a shared global model. This collaborative training allows nodes to benefit from each other's experience without transferring raw telemetry, reducing communication overhead and accelerating model convergence [40]. Importantly, FL in the HPC context is not about privacy—as all nodes belong to the same system—but about efficiency, scalability, and robustness [30].

This chapter presents a systematic study of applying FL for anomaly detection in HPC systems, combining results from two related investigations. The first study explored FL with dense autoencoders, which learn compressed representations of telemetry data. The second extended this approach to Long Short-Term Memory (LSTM) autoencoders, which are better suited for time-series data because they capture temporal dependencies in node behavior. Both studies evaluated multiple FL aggregation strategies, including FedAvg, FedProx, FedAdam, and FedYogi, across supervised, semi-supervised, and unsupervised learning paradigms.

Results on real telemetry from the Marconi100 Tier-0 supercomputer show that FL consistently improves anomaly detection accuracy compared to standalone models. In some cases, the F1-score more than doubled, while the amount of data required for training dropped by a factor of 15, reducing collection time from several months to just over a week. The LSTM-based models further enhanced detection robustness by modeling long-range dependencies in time series data, achieving statistically significant improvements over dense autoencoders.

In summary, this chapter introduces the motivation, context, and key findings from applying Federated Learning to anomaly detection in HPC. The focus is on demonstrating how FL not only improves accuracy but also reduces data requirements and accelerates deployment. Together, the two studies form a comprehensive contribution: showing both the general applicability of FL with simple autoencoders, and its enhanced effectiveness when extended to LSTM architectures. The following sections describe the methodology in detail and present the experimental results.

5.2 Motivation

High-Performance Computing (HPC) systems power some of the most demanding scientific and industrial applications of our time. From simulating climate change to advancing molecular biology and materials science, they enable discoveries that would be impossible on ordinary machines. However, with thousands of interconnected compute nodes generating terabytes of telemetry data every day, these systems are extremely difficult to monitor and manage. Even a small hardware fault or misconfiguration can quickly cascade into large-scale slowdowns or downtime, leading to significant financial costs and scientific delays [12]. Manual monitoring or rule-based tools, which once sufficed, are no longer adequate in such complex and large-scale environments. This creates an urgent need for automated anomaly detection methods that can reliably identify faults and irregularities as they emerge [6].

In recent years, machine learning (ML) has been increasingly adopted for anomaly detection in HPC. Autoencoders, in particular, have gained attention

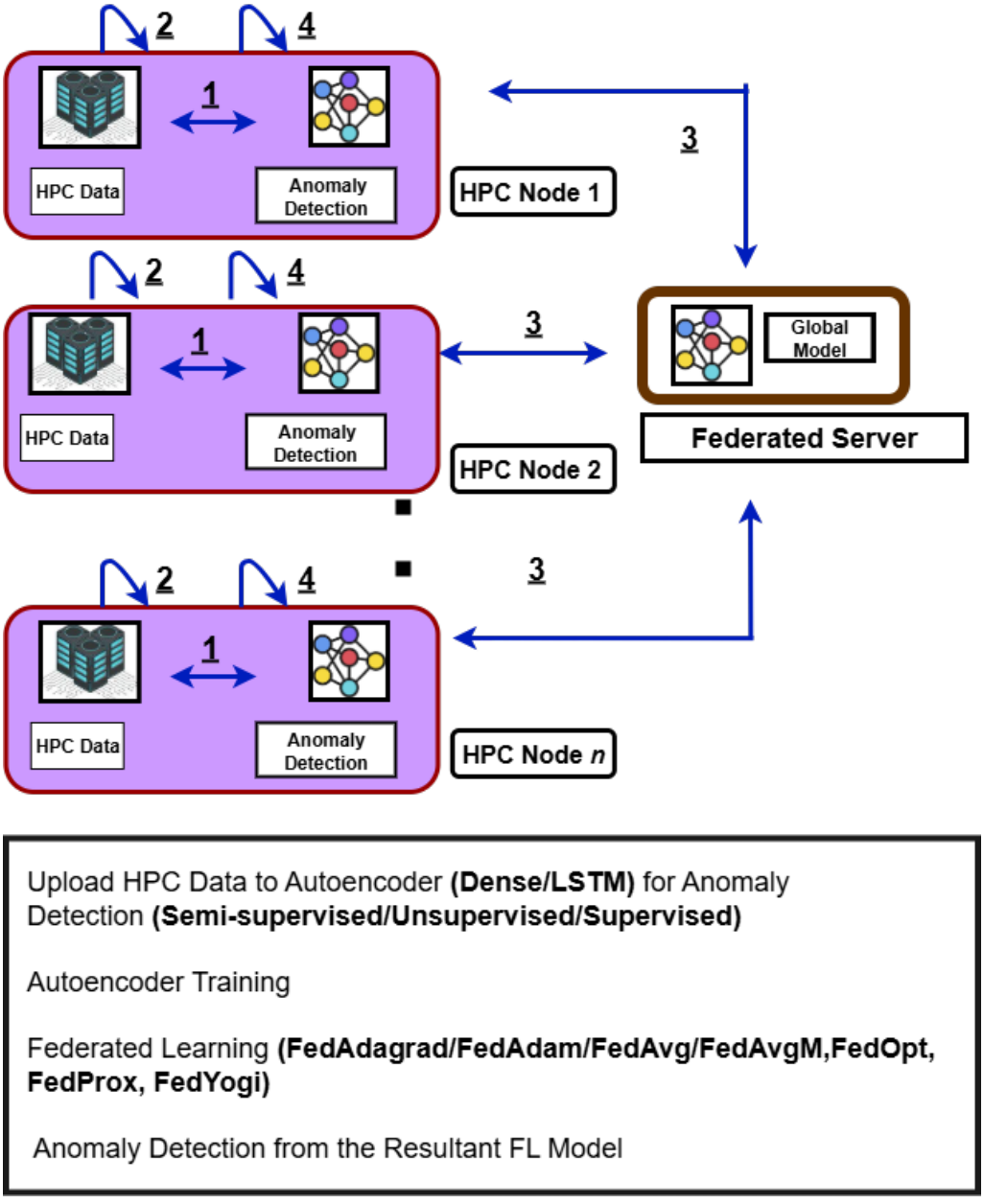


Figure 5.1: Anomaly Detection for HPC using FL

because they can learn compact representations of normal system behavior and detect deviations without requiring large sets of labeled anomalies. Yet conventional approaches suffer from two persistent problems [12]. First, they require long warm-up periods: months of data collection are often needed before a model becomes effective, delaying deployment in production settings. Second, they train models independently on each node, ignoring the similarities across nodes that could be leveraged to improve accuracy and efficiency. As a result, these methods often lack generalization and fail to scale to the realities of modern HPC [13].

To overcome these challenges, our first line of research introduced the use of federated learning (FL) [40] with dense autoencoders for anomaly detection. FL enables decentralized training: each node trains locally on its telemetry data, while only model updates are shared with a central server for aggregation. This collaborative approach reduces the need to centralize massive amounts of raw data, shortens training times, and improves generalization by allowing nodes to learn from one another. In our experiments on a Tier-0 production supercomputer, FL with dense autoencoders achieved substantial improvements across unsupervised, semi-supervised, and supervised settings. Most notably, it reduced the training data requirement by a factor of fifteen, cutting the warm-up period from several months to little more than a week. These results confirmed that FL is not only feasible in HPC environments but also highly effective in enhancing detection accuracy while lowering operational costs.

Despite these successes, dense autoencoders have a clear limitation: they capture correlations among features at a single time step but overlook temporal dynamics [56]. In practice, many HPC anomalies do not appear as isolated deviations but as gradual, evolving patterns in telemetry data. A fan that slowly loses efficiency, a GPU that overheats intermittently, or a power module that fluctuates over time are examples of faults that require temporal context to be recognized. Dense autoencoders are unable to capture such sequential dependencies, which limits their effectiveness in detecting faults that unfold across multiple time intervals [56].

To address this gap, we extended our approach by integrating Long Short-Term Memory (LSTM) autoencoders into the federated learning framework. LSTM

models are designed to capture long-range dependencies in time-series data, making them well suited for detecting anomalies that evolve gradually. By combining FL with LSTM autoencoders, we created a framework capable of leveraging both cross-node collaboration and temporal modeling. Applied to telemetry data from the Marconi100 Tier-0 supercomputer, this approach delivered statistically significant improvements in both F1-score and AUC, even when trained on severely reduced data fractions. Furthermore, we systematically evaluated six advanced federated aggregation strategies, including FedAvg, FedAvgM, FedProx, FedAdagrad, FedYogi, and FedAdam. This comprehensive comparison confirmed that the benefits of FL extend across a wide range of optimization settings and are robust to data scarcity.

Taken together, these two strands of research form a natural progression. The first study demonstrated the feasibility and benefits of federated learning with dense autoencoders, showing that collaboration among nodes can accelerate training and improve accuracy. Building on this foundation, the second study introduced LSTM autoencoders to incorporate temporal modeling, thereby addressing the sequential nature of HPC telemetry. This trajectory shows how federated learning, when paired with increasingly powerful model architectures, offers a scalable and data-efficient pathway for anomaly detection in production-scale HPC systems. It also highlights the broader potential of federated approaches to tackle long-standing challenges in system monitoring, where efficiency, robustness, and adaptability are as important as accuracy.

This chapter develops a unified account of these contributions, showing how federated learning—first with dense autoencoders and later with LSTM autoencoders—advances anomaly detection in HPC. By reducing data requirements, capturing temporal dependencies, and systematically evaluating federated optimization strategies, it demonstrates a complete framework that bridges feasibility with practical applicability in real-world supercomputing environments.

5.3 Methodology

This section outlines the methodology for applying Federated Learning (FL) with autoencoders for time series anomaly detection in high-performance computing

(HPC) systems. The study was carried out in two phases: first using dense autoencoders (AEs), and then extending the framework with Long Short-Term Memory (LSTM) autoencoders to better capture temporal dependencies.

5.3.1 Overview

Figure 5.1 provides a high-level view of the proposed framework. The main goal is to improve anomaly detection accuracy across HPC nodes while reducing training time and data requirements. Although privacy is not a primary concern in our setting—since all nodes belong to the same supercomputer—FL offers architectural and performance benefits. It enables decentralized training across nodes, leveraging behavioral similarities without transferring raw telemetry data, thus lowering communication overhead and improving scalability.

We evaluate this framework across three training paradigms:

- **Unsupervised learning:** Models are trained on unlabeled data containing both normal and anomalous states.
- **Semi-supervised learning:** Models are trained only on data from normal node behavior.
- **Supervised learning:** Models are trained on fully labeled datasets.

Each node independently trains a local autoencoder, and a central server coordinates model aggregation using various FL strategies. This enables collaborative training while preserving node-specific learning.

5.3.2 Dense Autoencoder for Anomaly Detection

The first phase of our work employed dense (fully connected) autoencoders for anomaly detection. Dense AEs are well-suited for capturing feature-level correlations in telemetry data. Each node trained a local AE on its telemetry sequences, and anomalies were detected using reconstruction error as the metric. Despite their simplicity, dense AEs provided a solid baseline and demonstrated the benefits of federated training across heterogeneous nodes.

Formally, given input x_t and its reconstruction $\hat{x}_t$, the reconstruction error is:

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

An anomaly is flagged if $\text{Error}_t > \theta$, where θ is a threshold determined through sensitivity analysis.

5.3.3 LSTM Autoencoder for Anomaly Detection

Building on the dense AE baseline, we extended the framework to LSTM autoencoders, which are designed to capture long-range temporal dependencies in time series telemetry data. This second phase addressed the limitation of dense AEs, which treat each input vector independently and cannot fully exploit sequential patterns.

The LSTM AE reconstructs input sequences, and the reconstruction error is again used as the anomaly indicator, following the same error formulation as above. By modeling temporal context, LSTM AEs achieved greater robustness and improved detection accuracy, especially under noisy or partially observed conditions.

5.3.4 Federated Learning Setup

Each HPC node acts as a client in the FL setup. Clients train models locally on their telemetry data and send model updates—not raw data—to a central server. The server aggregates updates to refine the global model.

The global optimization objective is defined as:

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

where $F_k(w)$ is the local loss on client k, n_k is the number of samples on client k, and w represents the global model parameters.

We implemented six federated optimization strategies: FedAvg, FedAvgM, FedProx, FedAdagrad, FedYogi, and FedAdam. These algorithms differ in how

they update global parameters and handle heterogeneity across clients.

5.3.5 Training Modes

To evaluate the framework under varying levels of supervision, we defined three training modes:

- **Unsupervised Training:** AEs trained on unlabeled telemetry sequences with anomalies detected via reconstruction error.
- **Semi-Supervised Training:** AEs trained only on normal data, assuming anomalies are absent during training.
- **Supervised Training:** AEs trained with fully labeled data, providing an upper-bound benchmark.

5.3.6 FL vs. Non-FL Comparison

To quantify the benefits of FL, we compared:

- **Local AE Baseline:** Each node trains an AE (dense or LSTM) independently on its own local data, without any collaboration or parameter sharing.
- **FL AEs:** Nodes collaborate through federated training using one of the strategies described above.

All models were trained using hyperparameters selected through sensitivity analysis for their respective architectures. While dense and LSTM autoencoders required different configurations, within each family of models the same settings were consistently applied across federated and non-federated comparisons. Performance was also studied under reduced training data scenarios, using subsets of the full data (e.g., 1/2, 1/4, 1/8, 1/16).

5.4 Experimental Evaluation

To evaluate the effectiveness of the proposed Federated Learning (FL) framework for anomaly detection in HPC systems, we carried out an extensive set of experiments

on two production-scale datasets. The study was performed in two successive phases. In the first phase, dense autoencoders (AEs) were used to establish a strong baseline for FL in HPC anomaly detection. In the second phase, the approach was extended to LSTM autoencoders, enabling the evaluation of temporal modeling capabilities in a federated setting. Both phases were assessed under supervised, semi-supervised, and unsupervised learning paradigms.

5.4.1 Experimental Setup

All experiments were executed on a dedicated high-performance server equipped with 42 Intel(R) Xeon(R) Gold 6240R CPUs (96 cores in total), 790 GB RAM, and running Ubuntu 22.04. The software environment comprised Python 3.8, TensorFlow 2.x for model implementation and orchestration, and Scikit-learn for preprocessing and normalization tasks. Additional data management and automation were supported by `pandas`, `numpy`, and `bash` scripts, which enabled the simulation of node-level parallelism throughout the workflow. Federated Learning (FL) was implemented using the Flower framework [5], which provided the client–server architecture and supported the six aggregation strategies (FedAvg, FedAvgM, FedAdagrad, FedYogi, FedProx, FedAdam) employed in our experiments. The same framework was applied consistently across both dense AE and LSTM AE phases to ensure methodological uniformity and fair comparison.

This server-based setup provides a conservative estimate of training performance. In production HPC systems—such as the Tier-0 supercomputer infrastructure that motivates this study—training and resource utilization are expected to be significantly more efficient, given the availability of optimized large-scale computational resources.

5.4.2 Datasets

The datasets employed in this evaluation are described in detail in Chapter 4.

Node Selection and Experimental Scope

From each dataset (D1 and D2), for the dense AE experiments, anomaly detection was carried out on a subset of 70 nodes. These experiments provided the first systematic validation of federated anomaly detection using fully connected autoencoder architectures.

For the LSTM AE experiments, five groups of 17 nodes each (with slight overlap to balance data availability across datasets) were used. Each group participated in federated training independently. This grouping strategy allowed us to examine how federated optimization behaves when sequence models are applied to smaller collaborative subsets, while still retaining diversity across the system.

By structuring the evaluation in two stages, we ensure both breadth and depth of analysis. Dense AE experiments served as a baseline to verify the feasibility and advantages of FL in HPC anomaly detection. The LSTM AE experiments then extended this baseline, enabling us to study whether temporal dependency modeling could further enhance accuracy, robustness, and scalability in federated scenarios. This dual perspective highlights not only the effect of FL itself but also how model architecture impacts performance in production-like HPC environments.

In both phases, experiments were conducted under all three learning paradigms (supervised, semi-supervised, and unsupervised). For each paradigm, we compared: (i) local baseline models trained independently at each node, and (ii) federated models trained collaboratively using one of six aggregation strategies (FedAvg, FedAvgM, FedAdagrad, FedYogi, FedProx, and FedAdam). This consistent design enables a fair comparison across datasets, paradigms, and architectures, and ensures that insights from dense and LSTM experiments can be meaningfully connected.

5.4.3 Implementation

To evaluate the effectiveness of the proposed Federated Learning (FL) framework, we employed autoencoder-based anomaly detection models in two phases: first using dense autoencoders (AEs) as a baseline, and then extending the framework to Long Short-Term Memory (LSTM) autoencoders to capture temporal dependencies. Both models were tested under three training paradigms—supervised, unsupervised, and semi-supervised—to ensure comprehensive evaluation.

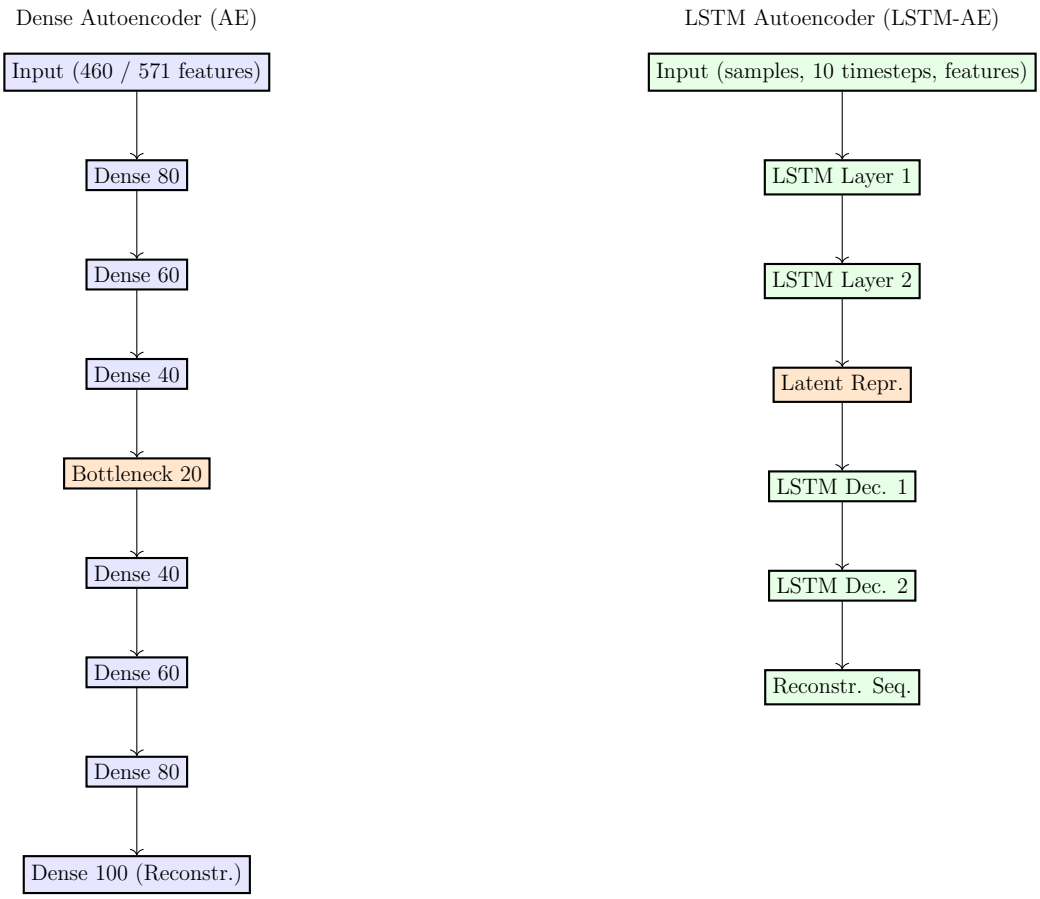


Figure 5.2: Comparison of Dense AE and LSTM AE architectures.

Dense Autoencoder Baseline

We began with dense (fully connected) autoencoders, a state-of-the-art baseline for anomaly detection in HPC systems. A separate AE was trained for each selected HPC node. Following Borghesi et al. [12], the semi-supervised setup trained models exclusively on normal data. In the unsupervised setting, models were trained on both normal and anomalous samples without labels [56], while supervised training used fully labeled datasets [3].

The timestamp feature was removed from all datasets. After preprocessing, 461 features remained in D1 and 572 in D2. Data was normalized to the $[0, 1]$ range using Scikit-learn, and split into 80% training and 20% testing subsets.

The dense AE architecture consisted of an input layer of size 460 (for D1) or

Table 5.1: Hyperparameters for Dense AE and LSTM AE models.

Hyperparameter	**Dense Autoencoder (AE)**	**LSTM Autoencoder (LSTM-AE)**
Input size	460 (D1), 571 (D2)	(samples, 10 timesteps, features: 460 for D1, 571 for D2)
Encoder layers	Dense: 80, 60, 40, bottleneck = 20 (ReLU)	Stacked LSTM layers (Figure 5.2)
Decoder layers	Dense: 40, 60, 80, 100	LSTM decoder (mirrored)
Activation	ReLU (hidden layers)	Tanh / default LSTM activations
Loss function	RMSE	MAE
Batch size	10	10
Learning rate	1×10^{-6}	1×10^{-6}
Epochs	10	10
Train/test split	80% / 20%	80% / 20%
Threshold (unsupervised/supervised)	Fixed 0.5	Fixed 0.65
Threshold (semi-supervised)	98th percentile of reconstruction error	98th percentile of reconstruction error
Optimizer	Adam (TF default)	Adam (TF default)
Evaluation metrics	Precision, Recall, F1, TPR, TNR	F1, AUC

571 (for D2), followed by three hidden dense layers of sizes 80, 60, and 40, with a bottleneck hidden layer of 20 neurons using ReLU activation. The decoder mirrored the encoder with four dense layers of sizes 40, 60, 80, and 100. Training used the root mean squared error (RMSE) loss function, batch size of 10, and learning rate $1e^{-6}$ for 10 epochs.

Anomalies were identified using reconstruction error or empirical thresholds. In supervised/unsupervised modes, outputs were compared against a threshold of 0.5 [56, 22, 49]. In the semi-supervised setting, the reconstruction error was compared against the 98^{th} percentile of the training error distribution [12]. Each AE was implemented using TensorFlow (Keras API), and performance was measured via precision, recall, F1-score, true positive rate, and true negative rate.

LSTM Autoencoder Extension

While dense AEs provided an effective baseline, they cannot fully exploit sequential patterns in HPC telemetry. To address this, we extended the framework to LSTM

autoencoders. Each selected HPC node was assigned a dedicated LSTM AE to capture long-range temporal dependencies.

Data preprocessing followed the same pipeline as with dense AEs: workload-related and non-numeric features (e.g., Nagios status) were removed, missing values discarded, and features normalized. The main difference was input reshaping: given the sequential nature of LSTMs, data was organized in the format `(samples, timesteps, features)`. A timestep of 10 was chosen, corresponding to a 150-minute window (15-minute logging interval) [28].

The LSTM AE architecture is illustrated in Figure 5.2. Hyperparameters are summarized in Table 6.1, chosen with reference to prior work [12, 56]. Training minimized mean absolute error (MAE), and anomalies were flagged if reconstruction error exceeded $\theta = 0.65$, a value consistent with prior HPC studies [16, 29, 87]. Detection performance was evaluated with F1-score and AUC [36].

Federated Learning Setup

Federated Learning was implemented using the Flower framework [5]. In both dense and LSTM AE experiments:

- Each HPC node acted as a client, training its local AE model (dense or LSTM).
- A central FL server (management node) coordinated aggregation and distributed updates.

Initially, each node trained its AE independently (non-FL baseline). Federated training was then executed using one of six aggregation strategies: FedAvg, FedAvgM, FedAdagrad, FedYogi, FedProx, or FedAdam. The `FlowerClient` class enabled client participation via `evaluate`, `fit`, and `get_parameters` methods. The server leveraged Flower's built-in implementations of these strategies.

Experiments were conducted with 1000, 1500, and 2000 FL rounds. Since model performance converged beyond 1000 rounds, we report results using the smaller number of rounds for consistency. At the end of training, each node received the updated global model and repeated anomaly detection under the chosen paradigm

(supervised, semi-supervised, or unsupervised). Metrics were logged to CSV for analysis and comparison.

5.4.4 Metrics

The evaluation of our federated anomaly detection framework considered both classical classification measures and widely adopted benchmarks from recent LSTM-based anomaly detection studies. This design ensured consistency with prior work while capturing complementary perspectives on performance.

For the dense autoencoder experiments, the following metrics were computed [3]:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5.1}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5.2}$$

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.3}$$

$$\text{True Negative Rate (TNR)} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \tag{5.4}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{5.5}$$

For the LSTM autoencoder experiments, in line with state-of-the-art methodologies [3, 56], two widely adopted metrics were selected: (i) the F1-score and (ii) the Area Under the Receiver Operating Characteristic Curve (AUC-ROC). The F1-score, defined in (5.6), is the harmonic mean of precision and recall, balancing false positives and false negatives:

$$F1\text{-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.6}$$

The AUC-ROC metric [41] quantifies the overall discriminative ability of the model. The ROC curve plots the true positive rate against the false positive rate across different thresholds, while the AUC corresponds to the probability that a randomly chosen positive instance is ranked higher than a randomly chosen

negative one.

Together, these metrics provide a comprehensive evaluation: dense autoencoders are assessed with detailed error rates (precision, recall, F1, TNR, FPR), whereas LSTM autoencoders are benchmarked using the standard F1 and AUC-ROC measures, facilitating fair comparison with existing literature.

5.5 Experimental Results

This section presents the experimental results of applying Federated Learning (FL) to anomaly detection in High-Performance Computing (HPC) systems. The evaluation was carried out in two successive phases. In the first phase, dense autoencoders were adopted as baseline models to establish the feasibility of FL for anomaly detection. In the second phase, the approach was extended to Long Short-Term Memory (LSTM) autoencoders to capture temporal dependencies in telemetry data. Both phases were tested under supervised, semi-supervised, and unsupervised paradigms on two datasets (D1 and D2) collected from the Marconi100 Tier-0 supercomputer at CINECA.

The results are organized as follows. Section 5.5.1 reports the performance of dense autoencoders with and without FL. Section 5.5.2 presents the outcomes of FL-enabled LSTM autoencoders, including robustness analysis.

5.5.1 Results with Dense Autoencoders

We address the following research questions in our experiments:

1. RQ1: Does applying FL to anomaly detection techniques in HPC improve the performance of anomaly detection models?

2. RQ2: Can FL reduce the amount of training data required to achieve the same performance as baseline non-FL models trained on full data?

3. RQ3: Can advanced FL aggregation algorithms outperform FedAvg in anomaly detection tasks?

RQ1: Performance Improvements with FL

Our first objective was to evaluate RQ1. Table 5.2 and Table 5.3 present results when anomaly detection was performed on reduced sets of nodes (5–60). The first row in each table indicates the number of nodes, while the second, fourth, and sixth rows report precision, TNR, and F1 of the baseline (non-FL) models. The corresponding third, fifth, and seventh rows show these metrics after applying FL. Across both datasets (D1 and D2), FL consistently improved performance in supervised, unsupervised, and semi-supervised settings, thereby supporting RQ1.

A detailed breakdown of semi-supervised, unsupervised, and supervised scenarios with all 70 nodes is shown in Table 5.4, Table 5.5, Table 5.6, Table 5.7, Table 5.8, and Table 5.9. For instance, Table 5.4 indicates that in the semi-supervised setup on D1, precision improved from 0.24 to 0.79, F1 from 0.31 to 0.84, and TNR from 0.24 to 0.81, while FPR decreased from 0.76 to 0.19. Similar gains are visible in Table 5.5 for D2, where F1 increased from 0.37 to 0.89. Tables 5.6 and 5.7 show corresponding improvements in the unsupervised setting, while Tables 5.8 and 5.9 demonstrate that even when supervised models already performed strongly with full data, FL provided substantial gains when training data was reduced. Overall, these results confirm that FL enhances anomaly detection performance, validating RQ1.

Individual Node Analysis

To further analyze RQ1, we examined improvements at the node level. Figures 5.3a, 5.4a, 5.3c, 5.4c, 5.3b, and 5.4b compare F1 scores before and after FL for individual nodes. Blue bars represent baseline results, and orange bars represent FL. For example, in Fig. 5.3a, 43 nodes reach an F1 between 0.9 and 1 after FL in the semi-supervised D1 setup, while Fig. 5.4b shows 58 nodes reaching this range in the supervised D2 case with only one-quarter of the training data. These consistent per-node improvements further validate RQ1. Recall remained at 1 across all experiments, indicating that no anomalies were missed.

RQ2: Reducing Training Data Requirements

To explore RQ2, we reduced training data to fractions of the original: 1/2 (2.5 months), 1/4 (1.25 months), 1/6 (3.3 weeks), 1/8 (2.5 weeks), and 1/16 (1.25 weeks). The goal was to determine the minimum data needed with FL to match the baseline autoencoder trained on the full dataset [48]. Results are shown in the subsequent rows of Table 5.4, Table 5.5, Table 5.6, Table 5.7, Table 5.8, and Table 5.9.

For instance, Table 5.4 demonstrates that FL applied to semi-supervised models on D1 with reduced data achieved higher precision, F1, and TNR than baseline models trained on the full dataset. Similar results hold for D2 (Table 5.5). In unsupervised settings, Tables 5.6 and 5.7 show that FL consistently outperformed full-data baseline models. In the supervised case, while baseline models already achieved near-perfect performance with half data, FL significantly improved results when only one-quarter data was used (Tables 5.8 and 5.9). These findings confirm that FL reduces the data requirements for anomaly detection, validating RQ2.

RQ3: Alternative Aggregation Strategies

Finally, to address RQ3, we compared FedAvg with FedAdagrad, FedAvgM, FedProx, FedYogi, and FedAdam. Results are summarized in Table 5.10 (D1) and Table 5.11 (D2). For D1, FedAdagrad achieved the highest precision (0.85) in semi-supervised settings, while FedYogi and FedAdagrad achieved the best F1 (0.93). In unsupervised settings, FedYogi and FedAdagrad again outperformed others, improving F1 to 0.81. In supervised detection with one-quarter training data, FedAdagrad, FedAvgM, and FedYogi each boosted precision and F1 to 0.99. Similar trends hold for D2, where FedAdagrad and FedYogi consistently delivered the strongest improvements across all paradigms. These results clearly show that advanced aggregation strategies outperform FedAvg in multiple scenarios, validating RQ3.

Table 5.2: Experimental Results Obtained Over Reduced Number of Nodes in Anomaly Detection Using D1

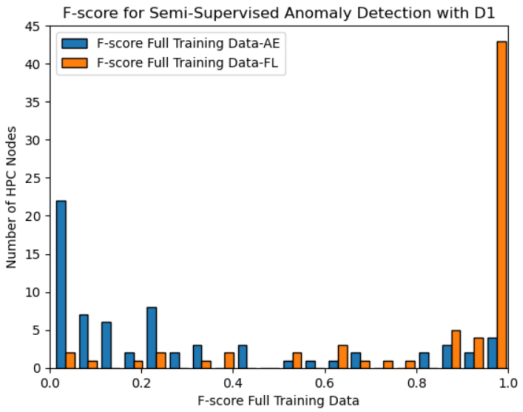
	Semi-Supervised							**Unsupervised**							**Supervised($1/4^{th}$ Training Data)**						
# of Nodes ⟹	**5**	**10**	**15**	**30**	**40**	**50**	**60**	**5**	**10**	**15**	**30**	**40**	**50**	**60**	**5**	**10**	**15**	**30**	**40**	**50**	**60**
Avg.Prec.-Non-FL	0.78	0.63	0.52	0.73	0.62	0.5	0.42	0.99	0.98	0.94	0.82	0.75	0.52	0.39	0.99	0.99	0.99	0.96	0.92	0.88	0.82
Avg.Prec.-FL	0.99	0.99	0.89	0.99	0.95	0.91	0.88	0.99	0.99	0.99	0.98	0.94	0.88	0.81	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Avg.TNR.-Non-FL	0.84	0.7	0.52	0.8	0.67	0.54	0.45	0.99	0.97	0.96	0.88	0.79	0.49	0.34	0.99	0.99	0.99	0.99	0.99	0.99	0.9
Avg.TNR.-FL	0.99	0.99	0.98	0.99	0.99	0.99	0.92	0.99	0.99	0.99	0.99	0.99	0.94	0.85	0.99	0.99	0.99	0.99	0.99	0.99	99
Avg.F1-Non-FL	0.8	0.69	0.54	0.82	0.68	0.56	0.49	0.99	0.98	0.96	0.85	0.77	0.56	0.41	0.99	0.99	0.99	0.98	0.98	0.92	0.85
Avg.F1-FL	0.99	0.99	0.96	0.99	0.99	0.98	0.94	0.99	0.99	0.99	0.99	0.98	0.95	0.89	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Table 5.3: Experimental Results Obtained Over Reduced Number of Nodes in Anomaly Detection Using D2

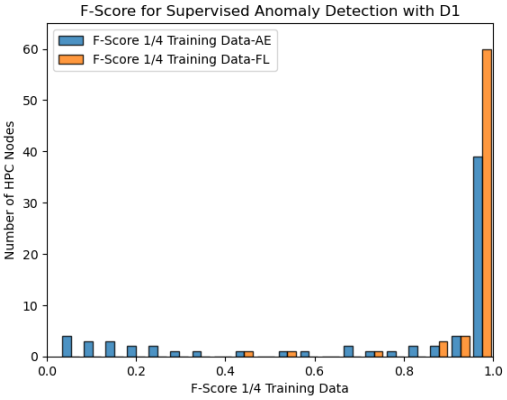
	Semi-Supervised							**Unsupervised**							**Supervised($1/4^{th}$ Training Data)**						
# of Nodes ⟹	**5**	**10**	**15**	**30**	**40**	**50**	**60**	**5**	**10**	**15**	**30**	**40**	**50**	**60**	**5**	**10**	**15**	**30**	**40**	**50**	**60**
Avg.Prec.-Non-FL	0.77	0.7	0.67	0.61	0.54	0.56	0.5	0.68	0.59	0.52	0.63	0.58	0.51	0.45	0.91	0.66	0.64	0.61	0.56	0.52	0.48
Avg.Prec.-FL	0.99	0.97	0.97	0.96	0.93	0.95	0.93	0.97	0.96	0.94	0.95	0.92	0.89	0.85	0.96	0.99	0.99	0.95	0.93	0.90	0.88
Avg.TNR.-Non-FL	0.8	0.73	0.71	0.65	0.59	0.59	0.54	0.7	0.63	0.56	0.66	0.6	0.52	0.46	0.91	0.66	0.65	0.62	0.59	0.56	0.51
Avg.TNR.-FL	0.99	0.99	0.98	0.97	0.94	0.97	0.96	0.98	0.97	0.94	0.96	0.94	0.9	0.87	0.97	0.99	0.99	0.99	0.91	0.91	0.85
Avg.F1-Non-FL	0.8	0.74	0.71	0.66	0.59	0.6	0.55	0.69	0.61	0.54	0.66	0.6	0.54	0.48	0.92	0.66	0.64	0.61	0.57	0.53	0.49
Avg.F1-FL	0.99	0.99	0.98	0.96	0.94	0.97	0.96	0.98	0.97	0.94	0.97	0.95	0.93	0.9	0.97	0.99	0.99	0.99	0.92	0.89	0.84

Table 5.4: Experimental Results Obtained Over 70 Nodes in Semi-supervised Anomaly Detection Using D1

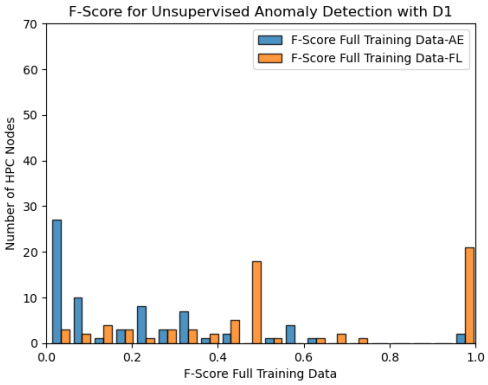
Fraction of Training Data	**Prec.-AE**	**Prec.-FL**	**Recall-AE**	**Recall-FL**	**F1-AE**	**F1-FL**	**TNR-AE**	**TNR-FL**	**FPR-AE**	**FPR-FL**
Full(5 Months)	0.24+/-0.31	**0.79+/-0.30**	1	1	0.31+/-0.33	**0.84+/-0.27**	0.24+/-0.30	**0.81+/-0.32**	0.76+/-0.33	**0.19+/-0.29**
1/2(2.5 Months)	0.23+/-0.30	**0.69+/-0.32**	1	1	0.29+/-0.32	**0.75+/-0.31**	0.25+/-0.31	**0.64+/-0.38**	0.75+/-0.29	**0.35+/-0.32**
$1/4^{th}$(1.25 Months)	0.21+/-0.28	**0.64+/-0.34**	1	1	0.28+/-0.31	**0.71+/-0.32**	0.22+/-0.29	**0.63+/-0.39**	0.78+/-0.31	**0.37+/-0.34**
$1/6^{th}$(3.33 Weeks)	0.23+/-0.31	**0.64+/-0.36**	1	1	0.29+/-0.33	**0.70+/-0.34**	0.25+/-31	**0.55+/-0.40**	0.75+/-0.33	**0.45+/-0.34**
$1/8^{th}$(2.5 Weeks)	0.22+/-0.29	**0.60+/-0.37**	1	1	0.29+/-0.31	**0.67+/-0.35**	0.24+/-0.33	**0.51+/-0.39**	0.76+/-0.34	**0.49+/-0.32**
$1/16^{th}$(1.25 Weeks)	0.25+/-0.310	**0.51+/-0.39**	1	1	0.22+/-0.34	**0.58+/-0.38**	0.23+/-0.30	**0.47+/-0.39**	0.77+/-0.36	**0.53+/-0.35**



(a) Semi-supervised



(b) Supervised

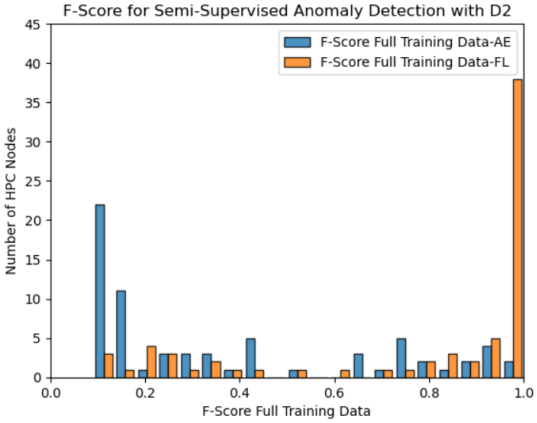


(c) Unsupervised

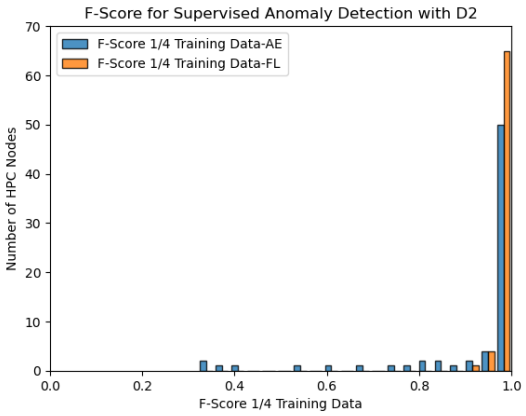
Figure 5.3: Increase in F-score after FL in anomaly detection using D1 across three learning paradigms.

Table 5.5: Experimental Results Obtained Over 70 Nodes in Semi-supervised Anomaly Detection Using D2

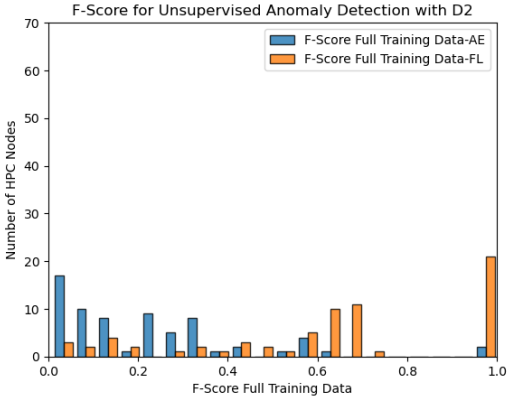
Fraction of Training Data	**Prec.-AE**	**Prec.-FL**	**Recall-AE**	**Recall-FL**	**F1-AE**	**F1-FL**	**TNR-AE**	**TNR-FL**	**FPR-AE**	**FPR-FL**
Full(5 Months)	0.3+/-0.25	**0.85+/-0.23**	1	1	0.37+/-0.33	**0.89+/-0.28**	0.29+/-0.33	**0.87+/-0.26**	0.7+/-0.26	**0.12+/-0.2**
1/2(2.5 Months)	0.28+/-0.32	**0.75+/-0.29**	1	1	0.35+/-0.33	**0.81+/-0.25**	0.3+/-0.19	**0.70+/-0.16**	0.68+/-0.25	**0.28+/-0.22**
$1/4^{th}$(1.25 Months)	0.27+/-0.25	**0.69+/-0.24**	1	1	0.33+/-0.32	**0.77+/-0.27**	0.28+/-0.31	**0.68+/-0.24**	0.7+/-0.25	**0.28+/-0.2**
$1/6^{th}$(3.33 Weeks)	0.29+/-0.29	**0.69+/-0.22**	1	1	0.3+/-0.24	**0.76+/-0.2**	0.31+/-0.3	**0.6+/-0.28**	0.68+/-0.32	**0.38+/-0.27**
$1/8^{th}$(2.5 Weeks)	0.28+/-0.3	**0.65+/-0.24**	1	1	0.3+/-0.26	**0.72+/-0.2**	0.28+/-0.30	**0.55+/-0.29**	0.7+/-0.27	**0.43+/-0.24**
$1/16^{th}$(1.25 Weeks)	0.29+/-0.2	**0.56+/-0.19**	1	1	0.27+/-0.25	**0.62+/-0.24**	0.27+/-0.25	**0.49+/-0.19**	0.72+/-0.33	**0.49+/-0.27**



(a) Semi-supervised



(b) Supervised



(c) Unsupervised

Figure 5.4: Increase in F1 after Federated Learning on Dataset D2 across semi-supervised, supervised, and unsupervised settings.

Table 5.6: Experimental Results Obtained Over 70 Nodes in Unsupervised Anomaly Detection Using D1

Fraction of Training Data	**Prec.-AE**	**Prec.-FL**	**Recall-AE**	**Recall-FL**	**F1-AE**	**F1-FL**	**TNR-AE**	**TNR-FL**	**FPR-AE**	**FPR-FL**
Full(5 Months)	0.22 +/- 0.28	**0.67+/- 0.25**	1	1	0.25+/- 0.24	**0.75+/- 0.22**	0.17+/- 0.19	**0.69+/- 0.15**	0.82+/- 0.29	**0.30/- 0.24**
1/2(2.5 Months)	0.18+/- 0.29	**0.61+/- 0.27**	1	1	0.18 +/- 0.3	**0.64+/- 0.27**	0.15+/- 0.3	**0.55+/- 0.24**	0.82+/- 0.28	**0.42+/- 0.2**
$1/4^{th}$(1.25 Months)	0.16+/- 0.3	**0.55+/- 0.29**	1	1	0.14+/- 0.29	**0.59+/- 0.25**	0.13+/- 0.3	**0.51+/- 0.26**	0.85+/- 0.29	**0.48+/- 0.24**
$1/6^{th}$(3.33 Weeks)	0.14+/- 0.29	**0.5+/- 0.24**	1	1	0.1+/- 0.24	**0.51+/- 0.2**	0.1+/-0.3	**0.46+/- 0.24**	0.88+/- 0.32	**0.52+/- 0.29**
$1/8^{th}$(2.5 Weeks)	0.1+/- 0.27	**0.45+/- 0.26**	1	1	0.08+/- 0.29	**0.47+/- 0.26**	0.07+/- 0.26	**0.40+/- 0.23**	0.92+/- 0.25	**0.57+/- 0.22**
$1/16^{th}$(1.25 Weeks)	0.05+/- 0.2	**0.39+/- 0.19**	1	1	0.05+/- 0.21	**0.4+/- 0.19**	0.04+/- 0.19	**0.35+/- 0.17**	0.97+/- 0.26	**0.63+/- 0.22**

Table 5.7: Experimental Results Obtained Over 70 Nodes in Unsupervised Anomaly Detection Using D2

Fraction of Training Data	**Prec.-AE**	**Prec.-FL**	**Recall-AE**	**Recall-FL**	**F1-AE**	**F1-FL**	**TNR-AE**	**TNR-FL**	**FPR-AE**	**FPR-FL**
Full(5 Months)	0.25+/- 0.21	**0.73+/- 0.19**	1	1	0.31+/- 0.18	**0.8+/- 0.16**	0.22+/- 0.17	**0.75+/- 0.16**	0.76+/- 0.18	**0.22+/- 0.1**
1/2(2.5 Months)	0.23 +/- 0.18	**0.66+/- 0.15**	1	1	0.24+/- 0.16	**0.69+/- 0.13**	0.2+/-0.1	**0.61+/- 0.09**	0.77+/- 0.15	**0.38+/- 0.1**
$1/4^{th}$(1.25 Months)	0.24+/- 0.15	**0.6+/- 0.11**	1	1	0.19+/- 0.17	**0.65+/- 0.13**	0.19+/- 0.13	**0.56+/- 0.21**	0.8+/- 0.17	**0.42+/- 0.12**
$1/6^{th}$(3.33 Weeks)	0.2+/- 0.13	**0.56+/- 0.21**	1	1	0.16+/- 0.12	**0.56+/- 0.13**	0.15+/- 0.11	**0.54+/- 0.12**	0.83+/- 0.18	**0.44+/- 0.14**
$1/8^{th}$(2.5 Weeks)	0.16+/- 0.11	**0.51+/- 0.09**	1	1	0.14+/- 0.12	**0.52+/- 0.11**	0.12+/- 0.1	**0.46+/- 0.1**	0.86+/- 0.16	**0.53+/- 0.12**
$1/16^{th}$(1.25 Weeks)	0.11+/- 0.07	**0.44+/- 0.17**	1	1	0.1+/- 0.08	**0.46+/- 0.18**	0.1+/- 0.09	**0.41/- 0.21**	0.89+/- 0.21	**0.59+/- 0.17**

Table 5.8: Experimental Results Obtained Over 70 Nodes in Supervised Anomaly Detection Using D1

Fraction of Training Data	**Prec.-AE**	**Prec.-FL**	**Recall-AE**	**Recall-FL**	**F1-AE**	**F1-FL**	**TNR-AE**	**TNR-FL**	**FPR-AE**	**FPR-FL**
Full(5 Months)	1		1		1		1		0	
1/2(2.5 Months)	0.94+/- 0.27	**0.99+/- 0.25**	1	1	0.95+/- 0.30	**0.99+/- 0.24**	0.95+/- 0.29	**0.99+/- 0.24**	0.05+/- 0.29	**0.01+/- 0.24**
$1/4^{th}$(1.25 Months)	0.71+/- 0.30	**0.93+/- 0.29**	1	1	0.75+/- 0.29	**0.96+/- 0.25**	0.77+/- 0.29	**0.96+/- 0.24**	0.21+/- 0.29	**0.02+/- 0.2**
$1/6^{th}$(3.33 Weeks)	0.65+/- 0.25	**0.86+/- 0.23**	1	1	0.68+/- 0.30	**0.88+/- 0.27**	0.71+/- 0.33	**0.87+/- 0.26**	0.28+/- 0.23	**0.11+/- 0.22**
$1/8^{th}$(2.5 Weeks)	0.61+/- 0.26	**0.79+/- 0.24**	1	1	0.61+/- 0.30	**0.8+/- 0.25**	0.64+/- 0.32	**0.79+/- 0.26**	0.34+/- 0.29	**0.19+/- 0.19**
$1/16^{th}$(1.25 Weeks)	0.55+/- 0.28	**0.71+/- 0.26**	1	1	0.54+/- 0.29	**0.72+/- 0.26**	0.58+/- 0.29	**0.7+/- 0.26**	0.40+/- 0.31	**0.27+/- 0.24**

Table 5.9: Experimental Results Obtained Over 70 Nodes in Supervised Anomaly Detection Using D2

Fraction of Training Data	**Prec.-AE**	**Prec.-FL**	**Recall-AE**	**Recall-FL**	**F1-AE**	**F1-FL**	**TNR-AE**	**TNR-FL**	**FPR-AE**	**FPR-FL**
Full(5 Months)	1		1		1		1		0	
1/2(2.5 Months)	0.93+/- 0.15	**0.99+/- 0.1**	1	1	0.97+/- 0.27	**0.99+/- 0.21**	0.96+/- 0.2	**0.99+/- 0.17**	0.01+/- 0.1	**0.0001+/- 0.04**
$1/4^{th}$(1.25 Months)	0.76+/- 0.24	**0.97/- 0.2**	1	1	0.79+/- 0.26	**0.98+/- 0.21**	0.82+/- 0.18	**0.98+/- 0.15**	0.17+/- 0.1	**0.01+/- 0.1**
$1/6^{th}$(3.33 Weeks)	0.69+/- 0.29	**0.91+/- 0.22**	1	1	0.73+/- 0.24	**0.92+/- 0.19**	0.75+/- 0.18	**0.92+/- 0.16**	0.23+/- 0.09	**0.06+/- 0.09**
$1/8^{th}$ Data (2.5 Weeks)	0.66+/- 0.21	**0.83+/- 0.19**	1	1	0.65+/- 0.19	**0.85+/- 0.17**	0.69+/- 0.15	**0.83+/- 0.1**	0.29+/- 0.11	**0.16+/- 0.09**
$1/16^{th}$(1.25 Weeks)	0.6+/- 0.24	**0.75+/- 0.2**	1	1	0.59+/- 0.19	**0.77+/- 0.17**	0.63+/- 0.15	**0.74+/- 0.1**	0.35+/- 0.12	**0.24+/- 0.1**

Table 5.10: Experimental Results Obtained Over 70 Nodes Using Advanced Aggregation Strategies of FL with D1

	Semi-Supervised				**Unsupervised**				**Supervised($1/4^{th}$ Training Data)**			
	Avg.-Prec.	**Avg.-F1**	**Avg.-TNR**	**Avg.-FPR**	**Avg.-Prec.**	**Avg.-F1**	**Avg.-TNR**	**Avg.-FPR**	**Avg.-Prec.**	**Avg.-F1**	**Avg.-TNR**	**Avg.-FPR**
ML	0.24	0.31	0.24	0.75	0.22	0.25	0.17	0.82	0.71	0.75	0.77	0.2
FedAvg	0.79	0.84	0.81	0.18	0.67	0.75	0.69	0.3	0.93	0.96	0.96	0.03
FedAdagrad	**0.85**	0.93	0.93	0.06	**0.78**	**0.81**	0.69	0.3	**0.99**	**0.99**	**0.99**	0
FedAvgM	0.81	0.89	0.97	0.02	0.73	0.72	**0.77**	0.22	**0.99**	**0.99**	**0.99**	0
FedProx	0.77	0.86	0.91	0.086	0.67	0.73	0.71	0.28	0.93	0.96	0.96	0.03
FedYogi	0.84	**0.93**	**0.98**	0.02	**0.78**	**0.81**	**0.77**	0.21	**0.99**	**0.99**	**0.99**	0
FedAdam	0.051	0.09	0.15	0.84	0.033	0.03	0.07	0.91	0.08	0.12	0.2	0.79

Table 5.11: Experimental Results Obtained Over 70 Nodes Using Advanced Aggregation Strategies of FL with D2.

	Semi-Supervised				**Unsupervised**				**Supervised($1/4^{th}$ Training Data)**			
	Avg.-Prec.	**Avg.-F1**	**Avg.-TNR**	**Avg.-FPR**	**Avg.-Prec.**	**Avg.-F1**	**Avg.-TNR**	**Avg.-FPR**	**Avg.-Prec.**	**Avg.-F1**	**Avg.-TNR**	**Avg.-FPR**
ML	0.3	0.37	0.29	0.7	0.25	0.31	0.22	0.71	0.76	0.79	0.82	0.17
FedAvg	0.85	0.89	0.87	0.13	0.73	0.8	0.75	0.24	0.97	0.98	0.98	0.03
FedAdagrad	**0.91**	**0.95**	0.96	0.02	**0.79**	0.79	0.81	0.18	**0.99**	**0.99**	**0.99**	0
FedAvgM	0.86	0.92	**0.98**	0	0.74	0.76	0.86	0.12	**0.99**	**0.99**	**0.99**	0
FedProx	0.83	0.89	0.93	0.06	0.7	0.71	0.82	0.17	0.97	**0.99**	**0.99**	0
FedYogi	0.9	**0.95**	0.95	0.03	0.8	**0.81**	0.9	0.09	**0.99**	**0.99**	**0.99**	0
FedAdam	0.072	0.12	0.19	0.8	0.06	0.04	0.09	0.9	0.13	0.17	0.17	0.83

5.5.2 Results with LSTM Autoencoders

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 Federated Learning (FL) with anomaly detection techniques in High-Performance Computing (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?

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 5.12, 5.13, and 5.14 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 Federated Learning (FL) provides across all learning paradigms.

Unsupervised Learning (Table 5.12) 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.

Semi-Supervised Learning (Table 5.13) 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 5.14) 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.

Table 5.12: 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 ($p < 0.01$).

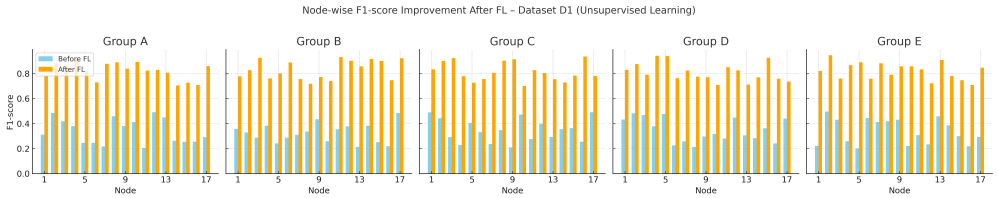
Dataset	Group	Metric	Full	1/2	1/4	1/8	1/16
D1	A	Non-FL F1	0.31	0.24	0.20	0.18	0.12
		FL F1	**0.86**	**0.65**	**0.45**	**0.40**	**0.35**
		Non-FL AUC	0.38	0.25	0.17	0.14	0.08
		FL AUC	**0.74**	**0.55**	**0.43**	**0.42**	**0.40**
	B	Non-FL F1	0.25	0.19	0.15	0.13	0.09
		FL F1	**0.78**	**0.59**	**0.51**	**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**

Table 5.13: F1 and AUC Scores for FL vs. Non-FL in Semi-Supervised Learning under Data Reduction (D1 & D2). Wilcoxon signed-rank test confirms significance ($p < 0.01$).

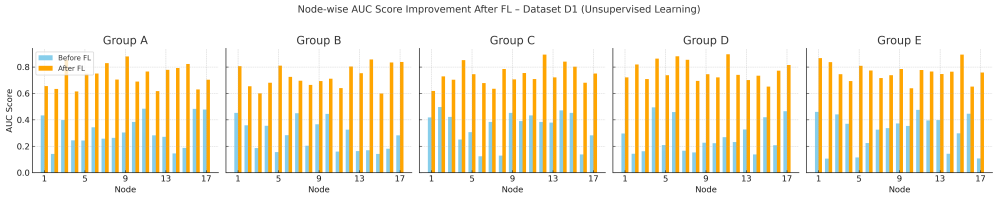
Dataset	Group	Metric	Full	1/2	1/4	1/8	1/16
D1	A	Non-FL F1	0.68	0.57	0.48	0.43	0.33
		FL F1	**0.95**	**0.85**	**0.64**	**0.62**	**0.60**
		Non-FL AUC	0.58	0.44	0.38	0.35	0.26
		FL AUC	**0.92**	**0.76**	**0.65**	**0.63**	**0.58**
	B	Non-FL F1	0.62	0.52	0.43	0.38	0.30
		FL F1	**0.87**	**0.79**	**0.70**	**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 5.14: F1 and AUC Scores for FL vs. Non-FL in Supervised Learning under Data Reduction (D1 & D2). Wilcoxon signed-rank test confirms significance ($p < 0.01$).

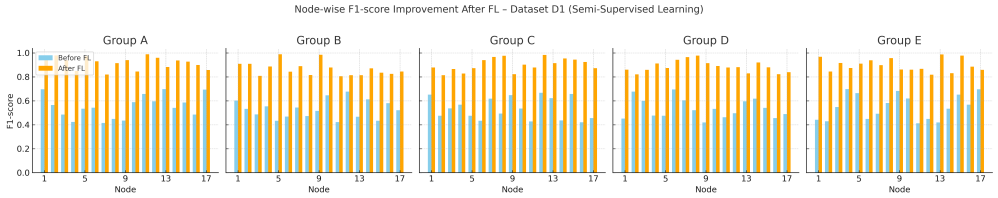
Dataset	Group	Metric	Full	1/2	1/4	1/8	1/16
D1	A	Non-FL F1	0.93	0.78	0.70	0.62	0.55
		FL F1	**0.99**	**0.99**	**0.90**	**0.87**	**0.88**
		Non-FL AUC	0.86	0.64	0.63	0.59	0.47
		FL AUC	**0.99**	**0.89**	**0.74**	**0.74**	**0.68**
	B	Non-FL F1	0.87	0.73	0.65	0.57	0.52
		FL F1	**0.99**	**0.99**	**0.96**	**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**



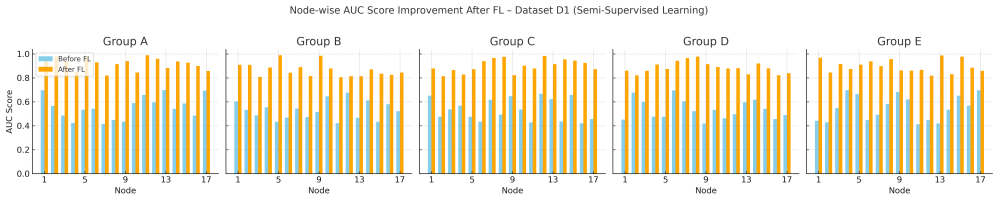
(a) F1-score – D1 (Unsupervised)



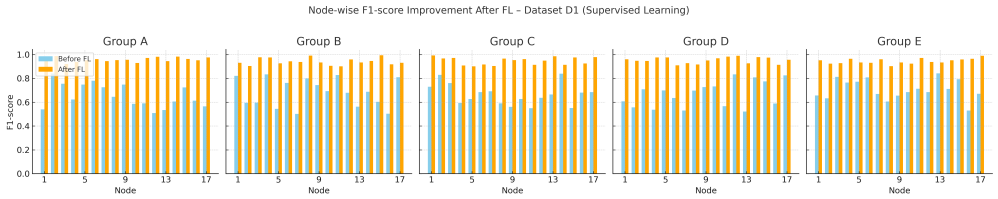
(b) AUC – D1 (Unsupervised)



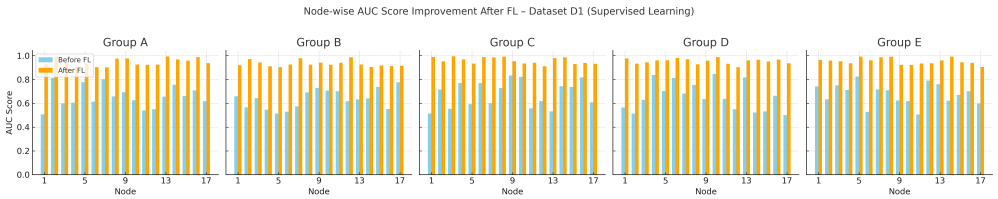
(c) F1-score – D1 (Semi-Supervised)



(d) AUC – D1 (Semi-Supervised)

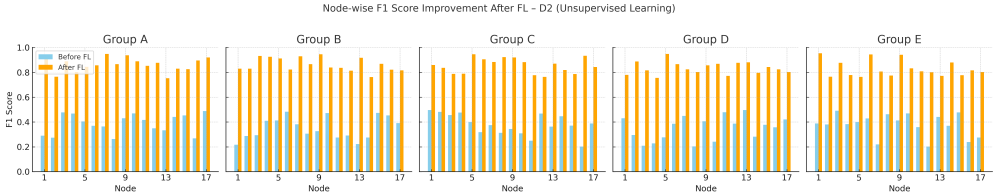


(e) F1-score – D1 (Supervised)

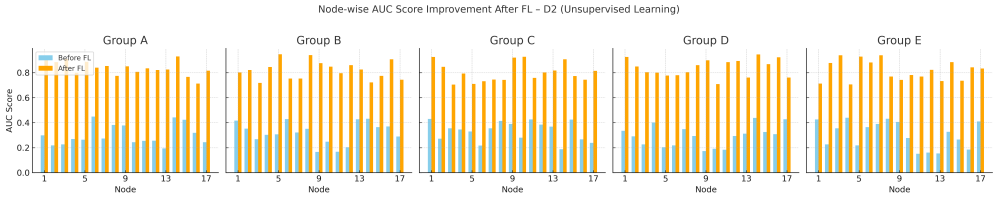


(f) AUC – D1 (Supervised)

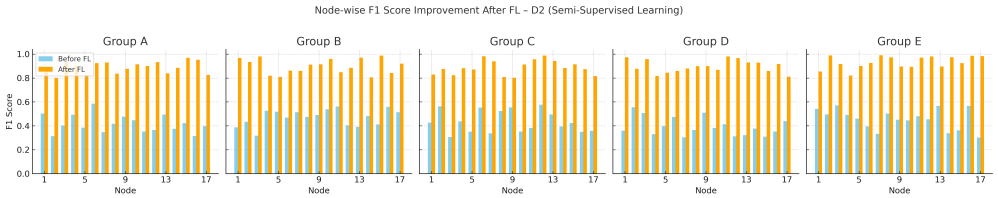
Figure 5.5: Node-wise improvement in F1 and AUC scores across all learning modes (Unsupervised, Semi-Supervised, Supervised) for Dataset D1. Each subfigure compares baseline and post-FL performance across Groups A–E.



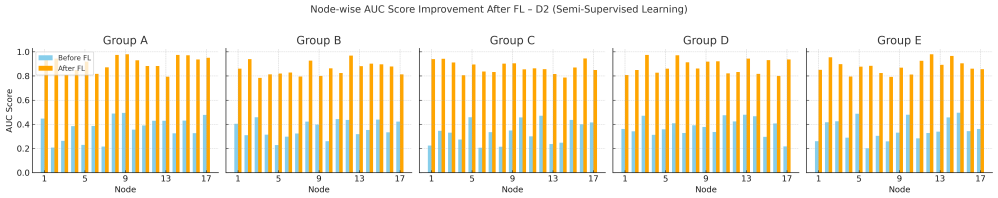
(a) F1-score Improvement – D2 (Unsupervised)



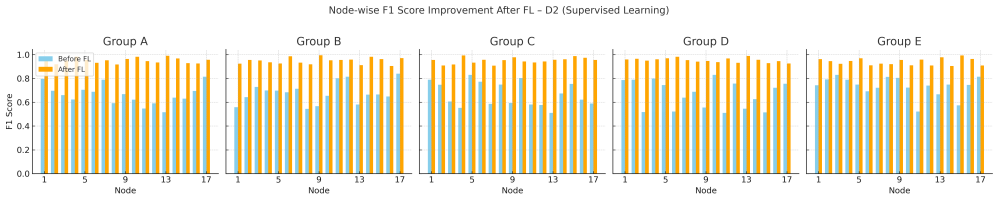
(b) AUC Improvement – D2 (Unsupervised)



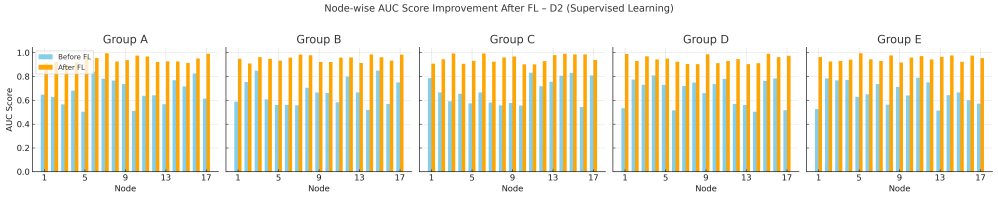
(c) F1-score Improvement – D2 (Semi-Supervised)



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



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



(f) AUC Improvement – D2 (Supervised)

Figure 5.6: 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.

Individual Node Analysis

Figures 5.5 and 5.6 provide node-wise analysis of F1 and AUC improvements across all learning paradigms for Datasets D1 and D2, respectively. Each subplot compares the performance of baseline (non-FL) LSTM autoencoders with their federated counterparts, highlighting the effect of FL on individual HPC nodes grouped into cohorts A through E.

For Dataset D1, FL consistently improved node-level performance across all groups. Notably, in the unsupervised setting, Group C exhibited the most significant uplift: several nodes saw F1-score improvements exceeding +100%, with post-FL scores approaching 0.87, compared to baseline values as low as 0.37. Similar trends were observed for Group D in semi-supervised learning, where federated training elevated multiple nodes from F1 scores near 0.50 to above 0.85. In the supervised setting, nearly all nodes achieved F1 scores above 0.95 after FL, indicating near-perfect anomaly detection across the group.

For Dataset D2, node-wise improvements were even more pronounced, especially under data-scarce conditions. In Group A, for example, FL raised the lowest-performing node's F1 score from 0.54 to 0.96 in the unsupervised mode. Group B showed marked heterogeneity in pre-FL performance, with F1 scores ranging from 0.09 to 0.42; FL harmonized these values to the 0.85–0.95 range, reducing variance while improving accuracy. Even nodes in Group E, which had extremely poor baseline performance in semi-supervised learning (F1 $\leq$ 0.25), achieved post-FL F1 scores of 0.70 and higher.

These improvements emphasize the ability of FL to generalize detection capabilities across individual nodes. The visual results in Figures 5.5 and 5.6 substantiate **RQ1** by showing that FL improves node-level anomaly detection performance consistently and significantly.

Performance Results Using a Limited Training Dataset

To evaluate the robustness of Federated Learning (FL) under real-world constraints, we investigated 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 performance*

to baseline, 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, long data collection cycles delay model deployment, making early anomaly detection impractical. Our goal is to determine whether FL can maintain performance while significantly reducing this data collection requirement.

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

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

Key Results (Tables 5.12 – 5.14):

- **Unsupervised Learning:** FL models trained with only 1/8 of the data often outperformed 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 delivered robust performance even at 1/16 training data.
  Example: D2 Group A — F1: 0.84 (FL, 1/16) vs. 0.39 (Non-FL, 1/16).
- **Supervised Learning:** FL maintained near-optimal results across all training fractions.
  Example: D2 Group E — F1: 0.99 (FL, 1/16) vs. 0.47 (Non-FL, 1/16).

These results confirm that FL can substantially reduce training data requirements while achieving high detection performance, strongly supporting **RQ2**.

Relation to RQ1: Although this section focuses on data reduction, the following insight reinforces:

- **RQ1:** *Does the integration of FL improve anomaly detection performance in HPC environments?*
  FL-based models consistently outperformed local LSTM baselines across all learning modes.

Conclusion: FL enables accurate anomaly detection with up to **15× less training data**, making it ideal for practical, real-time deployment in HPC environments. These findings demonstrate FL's scalability and efficiency in monitoring large-scale distributed systems.

Table 5.15: Comparison of F1 and AUC scores across different Federated Learning aggregation strategies on Dataset D1 under unsupervised, semi-supervised, and supervised learning paradigms.

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

Table 5.16: Comparison of F1 and AUC scores across different Federated Learning aggregation strategies on Dataset D2 under unsupervised, semi-supervised, and supervised learning paradigms.

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

Performance Evaluation of Federated Learning Aggregation Techniques

To assess the impact of advanced Federated Learning (FL) strategies on anomaly detection, we conducted comprehensive experiments using six FL aggregation algo-

rithms—FedAvg, FedProx, FedAdagrad, FedAvgM, FedYogi, and FedAdam—across five groups (A–E) in both datasets (D1 and D2) under unsupervised, semi-supervised, and supervised learning paradigms. The results, reported in Table 5.15 and Table 5.16, provide F1 and AUC scores for each strategy and reveal key trends in performance and generalization. Across both datasets, the FedAdagrad and FedAvgM algorithms consistently delivered the highest overall performance.

On Dataset D1, FedAdagrad achieved peak F1 scores of up to 0.94 (Group C, unsupervised) and 0.99 (Groups A–E, supervised), along with AUC scores as high as 0.92 (Group C, unsupervised) and 0.99 (multiple groups). FedAvgM closely followed, often matching FedAdagrad in both F1 and AUC, especially in semi-supervised settings where both algorithms attained near-perfect scores in several groups.

FedYogi and FedProx were also effective, particularly in supervised learning, though their scores were marginally lower than the top performers. In contrast, FedAdam showed substantially inferior performance in all cases, with F1 and AUC values rarely exceeding 0.20. This consistent underperformance suggests poor convergence or instability of the optimizer in the federated HPC anomaly detection context. **Summary of Table 5.15 (Dataset D1):**

- **FedAdagrad** delivers the highest overall performance, achieving up to 0.94 F1 and 0.99 AUC.
- **FedAvgM** performs nearly identically to FedAdagrad, especially strong in semi-supervised learning.
- **FedYogi** and **FedProx** show strong but slightly less consistent performance.
- **FedAdam** fails to generalize, with F1 and AUC scores capped at 0.20.

On Dataset D2, the results reinforced the trends observed in D1. FedAdagrad and FedAvgM again achieved top-tier F1 and AUC scores. For example, in Group A, both strategies reached 0.99 in both F1 and AUC across semi-supervised and supervised settings, demonstrating strong generalization despite changes in the underlying monitoring infrastructure and data characteristics. Among the remaining methods, FedYogi was the most robust third performer, particularly in

supervised learning, followed by FedProx. Once again, FedAdam exhibited the weakest performance across all learning modes and node groups, maintaining F1 and AUC values below 0.41 in all cases. **Summary of Table 5.16 (Dataset D2):**

- **FedAdagrad** and **FedAvgM** achieve nearly perfect F1 and AUC across all groups and training modes.
- **FedYogi** emerges as a strong third choice, with solid results in semi-supervised and supervised setups.
- **FedProx** remains competitive but slightly less stable across groups.
- **FedAdam** again underperforms, failing to exceed 0.41 in any setting.

These findings confirm RQ3: advanced FL aggregation methods—particularly FedAdagrad and FedAvgM—significantly outperform the standard FedAvg approach. Their consistent superiority across datasets and learning modes underscores their suitability for production-scale anomaly detection in federated HPC environments, where communication constraints are prevalent. **Qualitative Interpretation of FL Gains** While quantitative metrics such as F1-score and AUC provide compelling evidence of performance improvement, a deeper understanding of why federated learning (FL) leads to such improvements is critical for practical adoption and scientific validation.

One key insight is that HPC nodes, though individually monitored, often exhibit shared operational characteristics under normal conditions. FL capitalizes on this commonality by allowing models to collaboratively learn generalized representations of normal behavior, improving anomaly detection on individual nodes, especially when local anomaly samples are limited or sparse.

Furthermore, advanced FL strategies such as FedAdagrad and FedYogi outperform FedAvg in several scenarios. These optimizers adapt learning rates based on gradient history, making them more robust to non-IID distributions and sparse anomalies. In contrast, FedAdam was observed to underperform due to its sensitivity to noisy gradient updates, which are more common in HPC anomaly detection, where fault events are rare and unevenly distributed.

In reduced-data settings, FL mitigates the performance drop by aggregating patterns learned across nodes. This is particularly beneficial for nodes with limited historical data or those recently added to the system. As a result, the FL model generalizes better and exhibits resilience to data scarcity, a critical property for practical deployment in dynamic HPC environments. The observed improvements in F1-score and AUC are attributed to FL's ability to leverage common patterns across nodes. Although anomalies are rare and node-specific, many normal operational states are shared among nodes. FL allows the global model to capture these shared dynamics, thereby improving generalization across all nodes.

5.5.3 Deployment Feasibility Analysis

We conducted our experiments on a dedicated machine equipped with 42 Intel(R) Xeon(R) Gold 6240R CPUs (96 cores), 790 GB of RAM, and running Ubuntu 22.04. This environment was selected to provide a controlled platform for preliminary analysis of the proposed Federated Learning (FL) strategies before involving system administrators for deployment on an actual Tier-0 HPC system. As highlighted in prior work [12], the transition from a testing environment to a deployment environment is typically smooth. At deployment time, the full computational resources of the HPC nodes can be utilized for both training and inference, meaning that the performance reported here reflects a conservative, worst-case overhead. In practice, the actual overhead on production-scale systems would be significantly smaller.

Dense Autoencoders (DAEs) and LSTM Autoencoders (LSTM-AEs) were implemented to capture different aspects of node telemetry. While DAEs provide a strong baseline for anomaly detection, LSTM-AEs are able to exploit temporal correlations in the monitoring data. All experiments were executed within the same environment, enabling a fair comparison of the two architectures under unsupervised, semi-supervised, and supervised FL paradigms.

Training in this context is a relatively infrequent operation, typically executed only once every month or two [30]. This makes it feasible to allocate HPC nodes exclusively to the training phase for its entire duration, provided the required training time remains reasonable. We therefore carefully analyzed the time required

to train both DAEs and LSTM-AEs under FL.

Table 5.17 reports the measured times for 1000 FL rounds across different learning paradigms and datasets (D1 and D2). For Dense Autoencoders, 1000 rounds required 62 minutes (0.88 minutes per node) for semi-supervised training on 70 nodes of D1 and 69 minutes (0.98 minutes per node) on D2. The corresponding unsupervised setting required 56 minutes (0.80 minutes per node) for D1 and 63.4 minutes (0.90 minutes per node) for D2. In the supervised setting, the training time increased slightly to 72 minutes (1.02 minutes per node) for D1 and 78 minutes (1.11 minutes per node) for D2.

The use of LSTM Autoencoders, due to their recurrent structure and ability to model sequential dependencies, introduced a modest increase in training overhead. For example, the average time to complete 1000 FL rounds was approximately 10–15% higher than for Dense Autoencoders, yet still well within acceptable limits for sporadic training schedules. Given that HPC nodes are substantially more powerful than the experimental hardware used here, we can confidently conclude that such overhead would remain negligible in real deployments.

It is important to emphasize that the additional computational cost of LSTM-AEs is justified by their ability to capture temporal dynamics in the telemetry data. Many anomalies in HPC systems do not manifest as isolated events but evolve gradually over time, making temporal modeling critical for accurate detection. Thus, while DAEs offer a solid baseline, the use of LSTM-AEs ensures that the detection framework is sensitive to long-term patterns, ultimately improving reliability in real-world deployments.

Table 5.17: Training times (in minutes) for 1000 FL rounds on datasets D1 and D2. Per-node times are reported in parentheses.

Learning Paradigm	D1 (DAE)	D2 (DAE)	D1 (LSTM-AE)	D2 (LSTM-AE)
Semi-supervised	62 (0.88)	69 (0.98)	70 (1.00)	78 (1.11)
Unsupervised	56 (0.80)	63.4 (0.90)	64 (0.91)	72 (1.03)
Supervised	72 (1.02)	78 (1.11)	82 (1.17)	90 (1.28)

By its nature, inference in DL models is fast but must be performed very frequently (e.g., every 30 or 60 seconds, depending on the sampling rates of the monitoring infrastructure). Inference is performed on the computing node; therefore, we measure the overhead by monitoring the node's RAM consumption, which is

often the bottleneck due to being shared among multiple jobs. The average memory consumption required for inference is 8 MB, while the average inference time is 2.34 seconds per node.

Chapter Summary

This chapter presented federated learning as a framework for anomaly detection in high-performance computing systems, integrating both dense and LSTM autoencoder models. The study focused on node-level telemetry from the Marconi100 Tier-0 supercomputer, addressing the challenge of detecting anomalies in large-scale, heterogeneous environments. By training models locally on each node and aggregating them through federated strategies, the framework enabled collaborative learning without centralizing raw data, thus improving scalability and efficiency.

Experiments across unsupervised, semi-supervised, and supervised paradigms demonstrated that federated learning consistently outperformed isolated local models. Detection performance improved substantially, with F1-scores and AUC values more than doubling in some cases. A key outcome was the significant reduction in training data requirements: the time needed to build effective models decreased from several months to just over one week. Dense autoencoders effectively captured correlations among system features, while LSTM autoencoders added the ability to model temporal dependencies in time-series telemetry, leading to stronger detection of evolving anomalies.

Overall, the results confirmed that federated learning enhances the accuracy, efficiency, and robustness of anomaly detection in production-scale HPC systems. These findings establish a strong foundation for the subsequent exploration of transfer learning techniques aimed at extending anomaly detection capabilities to unseen nodes.

Chapter 6

Federated Transfer Learning for Anomaly Detection in High-Performance Computing

This chapter is based on the publications:

- **Farooq, Emmen**, Michela Milano, and Andrea Borghesi. "Federated Transfer Learning for Anomaly Detection in HPC Systems: First Real-World Validation on a Tier-0 Supercomputer." Expert Systems with Applications (2025): 129754.

6.1 Introduction

Ensuring the reliability of High-Performance Computing (HPC) infrastructures requires accurate and scalable anomaly detection methods [14]. Conventional centralized approaches struggle to cope with privacy constraints, data heterogeneity, and the scale of modern supercomputers, motivating decentralized learning techniques [56].

Federated Learning (FL) has shown promise in this setting by enabling distributed model training without centralizing raw telemetry data. However, FL alone does not transfer well to nodes that do not participate in training. Conversely,

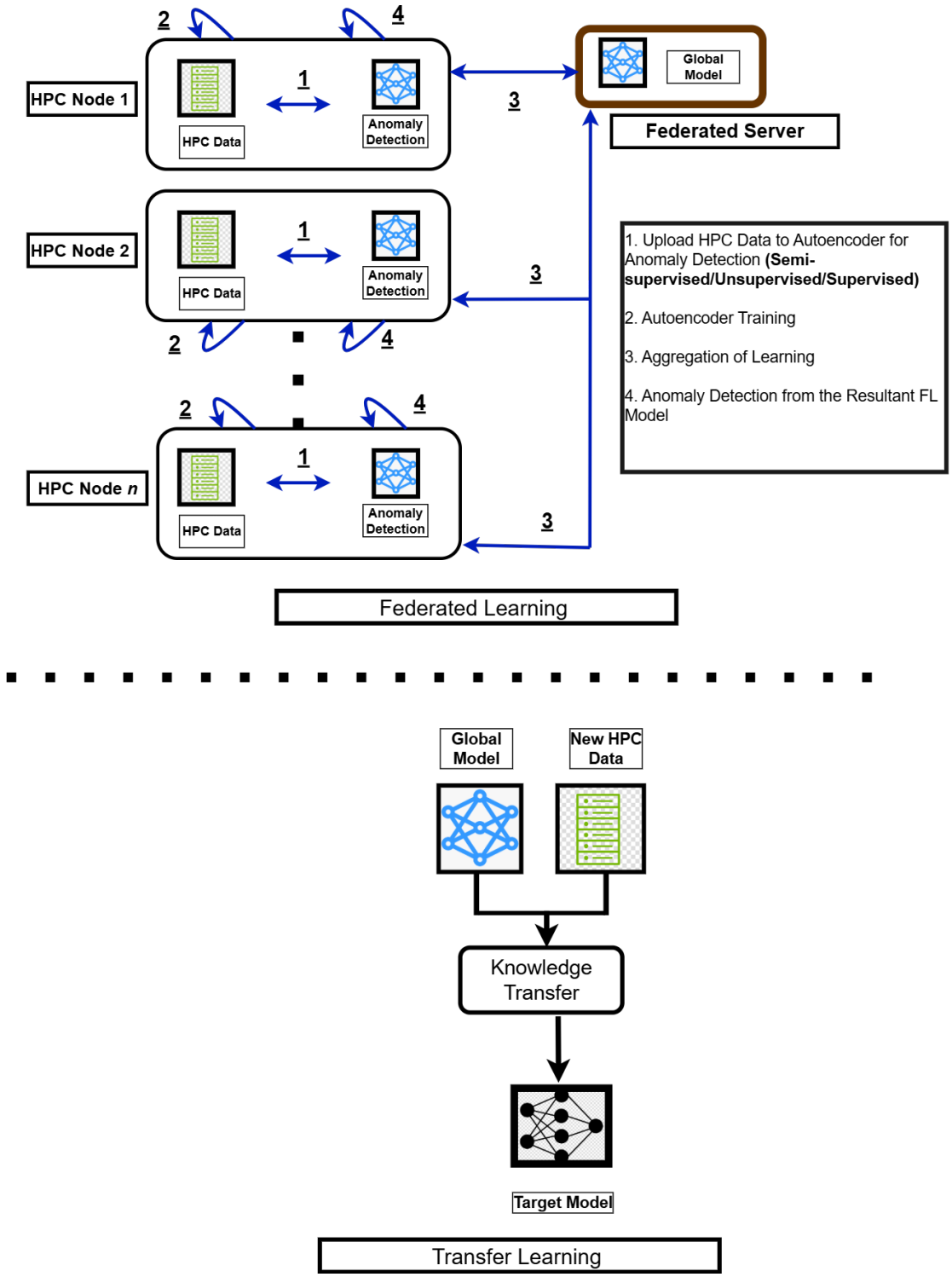


Figure 6.1: Overview of the proposed Federated Transfer Learning pipeline. Left: local autoencoder training on each HPC node. Center: federated learning via FedAvg using top-N nodes. Right: knowledge transfer and fine-tuning on the remaining $100 - N$ nodes.

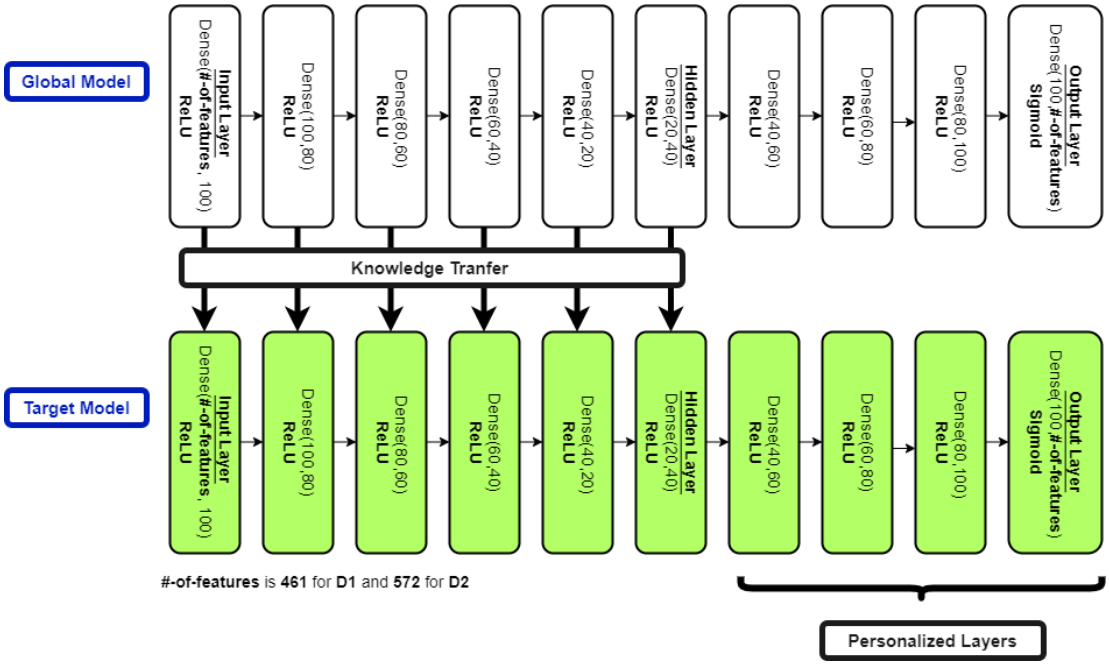


Figure 6.2: Knowledge Transfer

Transfer Learning (TL) provides mechanisms for adaptation across domains but lacks the scalability and privacy guarantees of FL [30].

This chapter presents Federated Transfer Learning (FTL) [61], which integrates FL and TL to address these limitations. We provide the first real-world validation of FTL for anomaly detection on a Tier-0 production supercomputer (Marconi100). The framework trains federated models on selected nodes and adapts them to unseen nodes through fine-tuning, thereby enhancing generalization in heterogeneous environments.

The broader background on anomaly detection in HPC systems and the foundations of federated and transfer learning are discussed in Chapter 2. Here, we focus specifically on the proposed FTL framework and its validation in a real production-scale setting.

Our contributions are fourfold:

1. the first demonstration of FTL in HPC anomaly detection,

2. evaluation across supervised, semi-supervised, and unsupervised paradigms,

3. analysis of two node selection strategies (Top-N and Random-N),

4. and empirical validation on two large-scale datasets collected from 100 nodes of Marconi100.

Results show consistent and significant performance improvements, confirming FTL's practicality for production-scale deployment.

The remainder of this chapter is organized as follows. Section 6.1 presents the motivation for the proposed approach. Section 6.2 describes the methodology and experimental setup. Section 6.3.5 reports and discusses the results. Finally, Section 6.4 concludes with key findings and outlines directions for future work.

Motivation

Modern High-Performance Computing (HPC) systems operate at unprecedented scale, with thousands of heterogeneous nodes continuously running mission-critical scientific and industrial workloads. Even short periods of failure or degraded performance can result in wasted millions of core-hours, increased energy consumption, and disruption of large collaborative projects. This high operational cost makes accurate and timely anomaly detection an indispensable requirement [17].

Existing anomaly detection strategies, however, remain inadequate gor production-scale Tier-0 environments. Centralized machine learning approaches cannot cope with the volume of telemetry, violate privacy constraints tied to vendor-specific configurations, and fail to generalize across non-identically distributed (non-IID) nodes [53]. Federated Learning (FL) has recently been proposed as a solution, enabling distributed model training without raw data sharing. Yet FL alone suffers from a critical limitation: the resulting models only benefit the nodes that participated in training, leaving the broader system underserved [72].

Transfer Learning (TL) addresses the complementary challenge of reusing knowledge across domains or devices, especially under limited labeling. However, TL by itself neither preserves privacy nor scales naturally to thousands of distributed clients. The coexistence of these limitations motivates a new paradigm that combines the scalability and privacy of FL with the adaptability of TL [9].

Federated Transfer Learning (FTL) [65] directly answers this need. By training models collaboratively on a subset of nodes and then transferring and fine-tuning them on unseen nodes, FTL provides:

1. **Generality across heterogeneous nodes** – enabling models to remain effective under diverse node behaviors without retraining from scratch.
2. **Privacy and data locality** – respecting the confidentiality of raw telemetry, which is critical in production systems.
3. **Operational feasibility** – reducing annotation costs by supporting semi-supervised and unsupervised modes, which reflect real-world monitoring conditions.

Validating FTL in a Tier-0 supercomputer such as Marconi100 is thus not only novel but also urgent. Previous FTL studies have largely focused on domains like healthcare or IoT; their transferability to HPC remains unproven [67]. Demonstrating FTL's effectiveness on production telemetry establishes its practical value for real-time monitoring, where even modest improvements in anomaly detection translate into tangible savings of compute resources and operational stability.

This work is therefore motivated by a dual imperative: bridging the gap between federated anomaly detection research and production-scale deployment, and ensuring that future exascale HPC systems remain both reliable and sustainable under increasing complexity.

6.2 Methodology

This section describes the experimental pipeline used to evaluate Federated Transfer Learning (FTL) for anomaly detection across 100 HPC nodes under unsupervised, semi-supervised, and supervised learning paradigms. The methodology follows a structured process involving local model training, federated aggregation via FedAvg, knowledge transfer to unseen nodes, and localized fine-tuning for personalization. The overall workflow is illustrated in Figure 6.1.

6.2.1 Local Model Training and Node Ranking

We employ dense autoencoders as the foundational architecture for anomaly detection on all nodes [61]. An autoencoder is a type of neural network designed to learn compressed representations of input data by minimizing the reconstruction error between input and output [12]. It consists of two components: an encoder that maps input data to a lower-dimensional latent space, and a decoder that reconstructs the original input from this latent representation [29]. The reconstruction error, typically measured using mean squared error (MSE), quantifies how accurately the model can reproduce the input data [16].

Autoencoders are particularly suitable for anomaly detection because they are trained primarily on normal (non-anomalous) data. Consequently, they learn to reconstruct expected system behavior [12]. During inference, anomalous inputs that deviate from this learned distribution tend to exhibit high reconstruction errors and are thus flagged as outliers.

This strategy aligns well with the characteristics of HPC environments [28, 30], where:

- Faults are infrequent compared to normal operation, resulting in highly imbalanced datasets.
- Data distributions across nodes are heterogeneous and non-iid.
- Annotating anomalies is resource-intensive and often infeasible in production environments.

As shown on the left side of Figure 6.1, each node independently trained a local anomaly detection model using its private data. The training configuration varied according to the target learning paradigm:

- **Unsupervised:** The autoencoder was trained solely on normal data, and anomalies were identified via high reconstruction error.
- **Semi-supervised:** A subset of labeled anomalies was included during training, allowing partial supervision to guide reconstruction.

- **Supervised:** A classification head was appended to the encoder, and the entire model was trained using labeled normal and anomalous samples.

Following training, each model was evaluated using the F1-score on a held-out test set, and all 100 nodes were ranked according to their individual anomaly detection performance. These rankings were subsequently used for node selection in the federated learning stage.

6.2.2 Federated Learning with FedAvg

From the ranked list, subsets of top-performing nodes were selected with $N \in \{3, 5, 10, 20, 25, 50\}$. These top-N nodes participated in federated learning (FL) using the Federated Averaging (FedAvg) algorithm, illustrated in the center of Figure 6.1.

FedAvg [30] performs decentralized training through the following steps:

1. A central server initializes and distributes a global model to the selected top-N nodes.
2. Each client node trains the model locally on its private data for a fixed number of epochs.
3. The locally updated model weights are sent back to the server.
4. The server computes a weighted average of the received updates based on the size of each local dataset:

$$w_{t+1} = \sum_{k=1}^{N} \frac{n_k}{n} w_t^{(k)}, \quad \text{where} \quad n = \sum_{k=1}^{N} n_k$$

This process continues for multiple communication rounds until convergence. The final aggregated model serves as a generalized anomaly detection model informed by the most reliable nodes and is saved for downstream transfer.

We hypothesize that high-performing nodes—those selected in the Top-N strategy—tend to capture more generalizable behavioral patterns. As such, their learned representations are expected to lie closer to the centroid of the feature distribution across all nodes.

6.2.3 Federated Transfer Learning (FTL)

As depicted in Figure 6.1, the global model obtained from federated learning is transferred to the remaining $100 - N$ nodes that were not involved in FL. Each of these nodes fine-tunes the transferred model using its local data to adapt it to its unique distribution and behavior patterns [65].

This step constitutes the Federated Transfer Learning (FTL) process. It enables reuse and personalization of the federated model without retraining from scratch, making it suitable for dynamic or resource-constrained settings [9]. The anomaly detection performance of each node is measured both before and after fine-tuning using Precision, Recall, True Negative Rate (TNR), and F1-score. The performance improvement, denoted by ΔF1, quantifies the effectiveness of FTL.

6.2.4 Comparison with Random Node Selection

To evaluate the significance of node selection strategy, the FL–FTL pipeline was repeated using randomly selected N nodes instead of top-ranked nodes. A global model trained from these randomly chosen nodes was then transferred to the same $100 - N$ target nodes and fine-tuned identically.

This comparison highlights the potential benefits of performance-aware node selection in improving the generalizability and adaptability of federated models across unseen clients.

6.2.5 Fine-Tuning for Local Personalization

In all FTL experiments—regardless of whether the global model originated from top-N or random-N training—fine-tuning was applied locally on each target node post-transfer [67]. This step was critical for aligning the transferred model with node-specific data distributions and enhancing anomaly detection accuracy in non-iid, real-world HPC systems.

6.3 Experimental Evaluation

To assess the effectiveness and generalizability of the proposed Federated Transfer Learning (FTL) framework for anomaly detection in real-world HPC environments, we conduct a thorough empirical evaluation across two production-scale datasets under supervised, semi-supervised, and unsupervised settings.

6.3.1 Experimental Setup

All experiments were conducted on a high-performance server equipped with 42 Intel(R) Xeon(R) Gold 6240R CPUs (96 cores total), 790 GB of RAM, and running Ubuntu 22.0. The implementation stack comprised Python 3.8 with TensorFlow 2.x for model development and orchestration, and Scikit-learn for data normalization and preprocessing. Data handling and automation were facilitated through `pandas`, `numpy`, and `bash`, with batch scripts used to simulate node-level parallelism across the experimental pipeline.

It is worth noting that this server-based setup represents a conservative estimate of performance. In real-world HPC environments—such as the Tier-0 supercomputing infrastructure leveraged in our study—the training time and resource efficiency are markedly improved due to the availability of highly optimized, large-scale computational resources.

6.3.2 Datasets

A detailed description of the datasets employed in this experiment can be found in Chapter 4.

Node Selection and Experimental Scope

We selected 100 nodes from each dataset (D1 and D2). These same nodes were used for: (i) baseline AE training, (ii) federated learning (Top-N or Random-N selection), and (iii) transfer learning. This consistent setup allows meaningful comparative analysis across learning paradigms and datasets.

6.3.3 Implementation

Incomplete and null values were dropped from all HPC data, followed by an 80%–20% train-test split and normalization to a [0, 1] range using Scikit-learn[1] before any implementation begins.

The implementation of our Federated Transfer Learning (FTL) framework follows a structured pipeline: (i) baseline local training of autoencoders on 100 HPC nodes, (ii) federated training on a selected subset of nodes using Federated Averaging (FedAvg), and (iii) knowledge transfer and personalization on the remaining nodes via fine-tuning. The full architectural pipeline is depicted in Figure 6.1. To promote reproducibility and enable further research, the complete implementation is available at: [2].

Node Selection Strategies: Top-N vs. Random-N

After training local autoencoders on all 100 nodes, each node's anomaly detection performance is evaluated using the F1-score on a held-out test partition. Based on this, we define two node selection strategies for the federated training phase:

- **Top-N Selection**: Nodes are ranked by descending F1-score. The top $N \in \{3, 5, 10, 20, 25, 50\}$ nodes are selected for participation in federated learning. This selection targets nodes with the most reliable anomaly detection models to serve as high-quality contributors to the global model.
- **Random-N Selection**: To evaluate the effect of performance-aware selection, we repeat the same experiment by randomly selecting N nodes from the pool of 100 for federated learning.

For each value of N, the remaining $100 - N$ non-participating nodes are kept identical across both Top-N and Random-N experiments. This ensures that any observed performance differences stem solely from the selection strategy used during the federated learning phase and not from variations in the nodes receiving the transferred model.

[1] https://scikit-learn.org
[2] https://github.com/EmmenFarooq/ftl-hpc-anomaly-detection

Autoencoder Architecture

Each HPC node trains a dense autoencoder with the following architecture:

- **Input Layer:** Size equal to the number of features (462 for D1, 573 for D2), activation: ReLU
- **Encoder:** Dense($100 \rightarrow 80$) $\rightarrow$ ReLU
  Dense($80 \rightarrow 60$) $\rightarrow$ ReLU
  Dense($60 \rightarrow 40$) $\rightarrow$ ReLU
  Dense($40 \rightarrow 20$) $\rightarrow$ ReLU
  Dense($20 \rightarrow 40$) $\rightarrow$ ReLU (latent)
- **Decoder:** Dense($40 \rightarrow 60$) $\rightarrow$ ReLU
  Dense($60 \rightarrow 80$) $\rightarrow$ ReLU
  Dense($80 \rightarrow 100$) $\rightarrow$ ReLU
  Dense($100 \rightarrow$ #features) $\rightarrow$ Sigmoid

This architecture provides sufficient depth to capture high-dimensional telemetry patterns and generalizes well across diverse node behaviors.

Federated Learning with TensorFlow Federated

Federated learning is implemented using TensorFlow Federated (TFF) [73]. The FedAvg algorithm is executed across the selected N nodes (from either Top-N or Random-N strategies). The procedure includes:

1. Initialize a global model and distribute it to selected clients.
2. Each client performs local training on private data using RMSprop optimizer for one epoch per round.
3. Clients return updated weights to the central server.
4. The server aggregates updates via a weighted average based on local dataset sizes.

This cycle is repeated for 1000 communication rounds to produce a global model representing generalized anomaly patterns across participating nodes.

Federated Transfer Learning and Personalization

After federated training, the global model is transferred to the remaining $100 - N$ nodes. As depicted in Figure 6.2:

- **Encoder layers** are frozen to retain generalized features.
- **Decoder layers** are reinitialized and fine-tuned locally to adapt to each node's specific data distribution.

This approach enables personalized anomaly detection without requiring complete retraining or labeled data on the target nodes.

Justification of Fine-Tuning Strategy

In the proposed FTL pipeline, we freeze the encoder and fine-tune only the decoder during the personalization step on non-participating nodes. This decision is motivated by practical and computational considerations. By freezing the encoder, we preserve the generalized representations learned during federated training, while limiting the number of trainable parameters—an approach that significantly reduces training time and resource overhead.

However, this design choice is not exhaustively justified in the current study. To explore whether full model fine-tuning could yield superior results, we perform a comparative analysis of F1-score improvements across 20 randomly selected nodes using both strategies: (i) decoder-only fine-tuning and (ii) full encoder+decoder fine-tuning. As shown in Figure 6.3, full fine-tuning results in marginally higher F1-scores across most nodes, suggesting potential gains in accuracy at the cost of increased computation.

Future work will investigate adaptive fine-tuning strategies where selective unfreezing of encoder layers is conditioned on inter-node distribution divergence, to balance performance gains with computational efficiency.

Training Configuration and Hyperparameters

All training and fine-tuning steps follow a consistent setup. The hyperparameters used are summarized in Table 6.1.

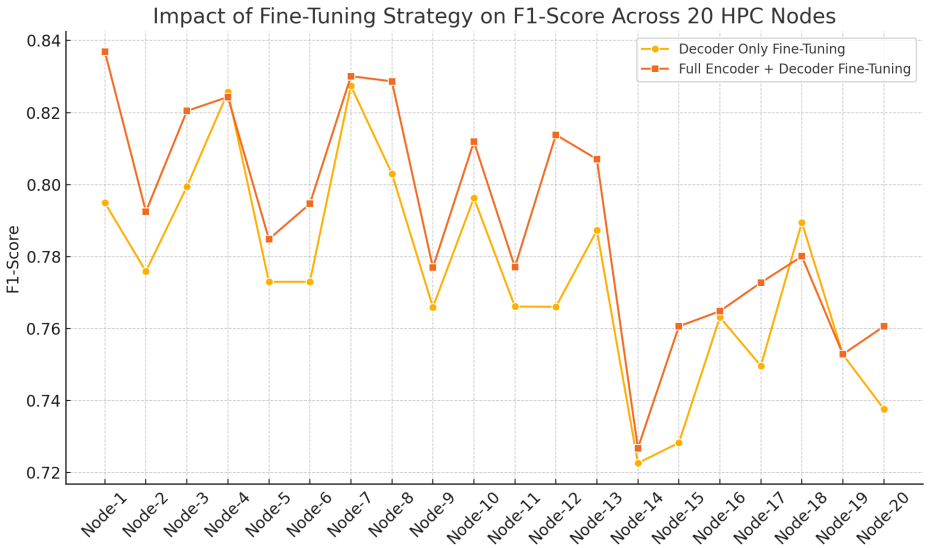


Figure 6.3: Comparison of F1-scores across 20 HPC nodes using decoder-only vs full fine-tuning strategies. While full fine-tuning slightly outperforms decoder-only tuning in most cases, the latter remains preferable in resource-constrained settings.

Anomaly Detection and Evaluation

Anomalies are identified based on reconstruction error or classification output:

- **Unsupervised:** Anomaly if reconstruction error > 0.5
- **Semi-supervised:** Threshold set to 98th percentile of training error on normal data
- **Supervised:** Anomaly if classifier output > 0.5

6.3.4 Evaluation Metrics

To evaluate the performance of the proposed anomaly detection framework, we adopt standard classification metrics that are widely used in the literature [3]. These metrics provide a comprehensive assessment of detection accuracy, robustness, and reliability across all learning paradigms.

Precision quantifies the proportion of correctly identified anomalies among all

Table 6.1: Hyperparameters used in autoencoder training

Hyperparameter	**Value**	**Applicable To**
Optimizer	RMSprop	All paradigms
Learning Rate	1×10^{-6}	All paradigms
Loss Function	Mean Squared Error (MSE)	All paradigms
Batch Size	10	All paradigms
Epochs	10	All paradigms
Reconstruction Threshold	0.5	Supervised, Unsupervised
Dynamic Threshold	98th percentile of training error	Semi-supervised only

instances predicted as anomalies. It is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{6.1}$$

Recall, also known as the true positive rate, measures the proportion of actual anomalies that are correctly identified:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6.2}$$

F1-Score is the harmonic mean of precision and recall, offering a balanced measure of model performance:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

True Negative Rate (TNR) evaluates the proportion of correctly identified normal instances:

$$\text{True Negative Rate} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}} \tag{6.4}$$

Delta F1 (ΔF1) quantifies the improvement in F1-score after applying Federated Transfer Learning (FTL) to non-participating nodes. It captures the performance gain achieved through knowledge transfer and is computed as:

$$\Delta\text{F1} = \text{F1}_{\text{after FTL}} - \text{F1}_{\text{before FTL}} \tag{6.5}$$

This metric is particularly important for evaluating the effectiveness of FTL in enhancing anomaly detection performance on nodes that were not directly involved in federated training.

Table 6.2: Performance of Semi-Supervised Anomaly Detection with Federated Transfer Learning (FTL) on Datasets D1 and D2 using 100 nodes. **Columns 2–3** show F1-scores before and after FL on selected Top-N or Random-N nodes. **Columns 5–12** report performance before and after applying FTL to the remaining $(100 - N)$ nodes. **Column 13** shows resulting F1 gain (ΔF1). Asterisk (*) indicates statistical significance ($p < 0.05$).

N	**FL on N Nodes**		**100–N**	**Before FTL on 100–N Nodes**				**After FTL on 100–N Nodes**				**F1 Gain**
	F1 (AE)	**F1 (FL)**		**Prec.**	**Rec.**	**TNR**	**F1**	**Prec.**	**Rec.**	**TNR**	**F1**	ΔF1
Dataset D1 – Top-N Strategy												
3	0.95	0.99	97	0.41	1.00	0.44	0.48	**0.88**	**1.00**	**0.90**	**0.92**	0.44*
5	0.92	0.99	95	0.37	1.00	0.35	0.42	**0.86**	**1.00**	**0.86**	**0.90**	0.48*
10	0.89	0.94	90	0.31	1.00	0.29	0.35	**0.80**	**1.00**	**0.79**	**0.84**	0.49*
20	0.85	0.94	80	0.26	1.00	0.25	0.29	**0.74**	**1.00**	**0.76**	**0.77**	0.48*
25	0.78	0.89	75	0.22	1.00	0.21	0.25	**0.70**	**1.00**	**0.71**	**0.75**	0.50*
50	0.72	0.87	50	0.17	1.00	0.15	0.21	**0.66**	**1.00**	**0.65**	**0.70**	0.49*
Dataset D1 – Random-N Strategy												
3	0.10	0.55	97	0.41	1.00	0.44	0.48	**0.57**	**1.00**	**0.60**	**0.61**	0.13*
5	0.25	0.75	95	0.37	1.00	0.35	0.42	**0.67**	**1.00**	**0.64**	**0.73**	0.31*
10	0.34	0.87	90	0.31	1.00	0.29	0.35	**0.71**	**1.00**	**0.68**	**0.76**	0.41*
20	0.92	0.99	80	0.26	1.00	0.25	0.29	**0.89**	**1.00**	**0.86**	**0.92**	0.63*
25	0.32	0.82	75	0.22	1.00	0.21	0.25	**0.61**	**1.00**	**0.61**	**0.63**	0.38*
50	0.64	0.98	50	0.17	1.00	0.15	0.21	**0.90**	**1.00**	**0.91**	**0.90**	0.69*
Dataset D2 – Top-N Strategy												
3	0.98	0.99	97	0.47	1.00	0.49	0.54	**0.93**	**1.00**	**0.96**	**0.97**	0.43*
5	0.96	0.99	95	0.43	1.00	0.40	0.48	**0.91**	**1.00**	**0.92**	**0.95**	0.47*
10	0.93	0.99	90	0.37	1.00	0.34	0.41	**0.85**	**1.00**	**0.85**	**0.89**	0.48*
20	0.90	0.97	80	0.32	1.00	0.30	0.35	**0.79**	**1.00**	**0.82**	**0.83**	0.48*
25	0.84	0.94	75	0.28	1.00	0.26	0.31	**0.75**	**1.00**	**0.77**	**0.80**	0.49*
50	0.78	0.91	50	0.23	1.00	0.20	0.27	**0.71**	**1.00**	**0.71**	**0.75**	0.48*
Dataset D2 – Random-N Strategy												
3	0.39	0.89	97	0.27	1.00	0.26	0.30	**0.67**	**1.00**	**0.67**	**0.70**	0.40*
5	0.91	0.99	95	0.21	1.00	0.21	0.24	**0.94**	**1.00**	**0.97**	**0.96**	0.72*
10	0.22	0.71	90	0.26	1.00	0.29	0.36	**0.62**	**1.00**	**0.59**	**0.67**	0.31*
20	0.59	0.91	80	0.14	1.00	0.11	0.17	**0.73**	**1.00**	**0.70**	**0.78**	0.61*
25	0.15	0.61	75	0.46	1.00	0.49	0.53	**0.51**	**1.00**	**0.53**	**0.55**	0.02
50	0.28	0.79	50	0.32	1.00	0.38	0.42	**0.64**	**1.00**	**0.61**	**0.63**	0.21

Table 6.3: Performance of Unsupervised Anomaly Detection with Federated Transfer Learning (FTL) on Datasets D1 and D2 using 100 nodes. **Columns 2–3** show F1-scores before and after FL on selected Top-N or Random-N nodes. **Columns 5–12** report performance before and after applying FTL to the remaining $(100 - N)$ nodes. **Column 13** shows resulting F1 gain (ΔF1). Asterisk (*) indicates statistical significance ($p < 0.05$).

N	**FL on N nodes**		$100-N$	**Before FTL on $100-N$ nodes**				**After FTL on $100-N$ nodes**				**ΔF1**
	F1 (AE)	**F1 (FL)**		**Prec.**	**Rec.**	**TNR**	**F1**	**Prec.**	**Rec.**	**TNR**	**F1**	
Dataset D1 – Top-N Strategy												
3	0.83	0.94	97	0.36	1.00	0.39	0.42	**0.76**	**1.00**	**0.78**	**0.81**	0.39*
5	0.81	0.91	95	0.31	1.00	0.28	0.34	**0.74**	**1.00**	**0.75**	**0.78**	0.44*
10	0.79	0.88	90	0.24	1.00	0.27	0.34	**0.69**	**1.00**	**0.70**	**0.73**	0.39*
20	0.73	0.84	80	0.19	1.00	0.17	0.23	**0.63**	**1.00**	**0.65**	**0.69**	0.46*
25	0.71	0.82	75	0.14	1.00	0.17	0.24	**0.61**	**1.00**	**0.63**	**0.65**	0.41*
50	0.65	0.78	50	0.10	1.00	0.09	0.13	**0.58**	**1.00**	**0.56**	**0.60**	0.47*
Dataset D1 – Random-N Strategy												
3	0.28	0.71	97	0.36	1.00	0.39	0.42	**0.88**	**1.00**	**0.90**	**0.92**	0.50*
5	0.35	0.79	95	0.31	1.00	0.28	0.34	**0.50**	**1.00**	**0.52**	**0.56**	0.22*
10	0.38	0.84	90	0.24	1.00	0.27	0.34	**0.81**	**1.00**	**0.82**	**0.85**	0.51*
20	0.41	0.89	80	0.19	1.00	0.17	0.23	**0.60**	**1.00**	**0.64**	**0.63**	0.40*
25	0.33	0.80	75	0.14	1.00	0.17	0.24	**0.39**	**1.00**	**0.38**	**0.41**	0.17
50	0.30	0.74	50	0.10	1.00	0.09	0.13	**0.32**	**1.00**	**0.37**	**0.36**	0.23*
Dataset D2 – Top-N Strategy												
3	0.87	0.95	97	0.43	1.00	0.41	0.47	**0.86**	**1.00**	**0.89**	**0.91**	0.44*
5	0.85	0.94	95	0.39	1.00	0.33	0.42	**0.81**	**1.00**	**0.85**	**0.89**	0.47*
10	0.81	0.91	90	0.30	1.00	0.29	0.34	**0.77**	**1.00**	**0.79**	**0.82**	0.48*
20	0.76	0.89	80	0.28	1.00	0.24	0.31	**0.71**	**1.00**	**0.72**	**0.75**	0.44*
25	0.72	0.85	75	0.22	1.00	0.21	0.24	**0.68**	**1.00**	**0.69**	**0.70**	0.46*
50	0.68	0.80	50	0.19	1.00	0.16	0.21	**0.64**	**1.00**	**0.64**	**0.63**	0.42*
Dataset D2 – Random-N Strategy												
3	0.29	0.66	97	0.43	1.00	0.41	0.47	**0.86**	**1.00**	**0.89**	**0.93**	0.46*
5	0.41	0.75	95	0.39	1.00	0.33	0.42	**0.75**	**1.00**	**0.76**	**0.79**	0.37*
10	0.50	0.78	90	0.30	1.00	0.29	0.34	**0.62**	**1.00**	**0.64**	**0.66**	0.32*
20	0.52	0.82	80	0.28	1.00	0.24	0.31	**0.66**	**1.00**	**0.68**	**0.70**	0.39*
25	0.37	0.74	75	0.22	1.00	0.21	0.24	**0.29**	**1.00**	**0.34**	**0.32**	0.08
50	0.40	0.79	50	0.19	1.00	0.16	0.21	**0.56**	**1.00**	**0.58**	**0.63**	0.42*

Table 6.4: Performance of Supervised Anomaly Detection with Federated Transfer Learning (FTL) on Datasets D1 and D2 using $1/4^{\text{th}}$ training data. **Columns 2–3** report F1-scores before and after Federated Learning (FL) on selected Top-N or Random-N nodes. **Columns 5–12** show metrics before and after applying FTL to the remaining $(100 - N)$ nodes. **Column 13** shows resulting F1-score gain (ΔF1). Asterisk (*) indicates statistical significance ($p < 0.05$).

N	FL on N Nodes		100–N	Before FTL on 100–N Nodes				After FTL on 100–N Nodes				F1 Gain
	F1 (AE)	F1 (FL)		Prec.	Rec.	TNR	F1	Prec.	Rec.	TNR	F1	ΔF1
Dataset D1 – Top-N Strategy												
3	0.76	0.99	97	0.64	1.00	0.63	0.65	**0.96**	**1.00**	**0.95**	**0.96**	0.31*
5	0.65	0.92	95	0.59	1.00	0.57	0.60	**0.89**	**1.00**	**0.87**	**0.92**	0.32*
10	0.60	0.91	90	0.53	1.00	0.52	0.54	**0.86**	**1.00**	**0.88**	**0.92**	0.38*
20	0.54	0.89	80	0.45	1.00	0.43	0.46	**0.80**	**1.00**	**0.79**	**0.83**	0.37*
25	0.43	0.84	75	0.34	1.00	0.33	0.35	**0.73**	**1.00**	**0.75**	**0.77**	0.42*
50	0.34	0.76	50	0.26	1.00	0.24	0.26	**0.65**	**1.00**	**0.66**	**0.69**	0.43*
Dataset D1 – Random-N Strategy												
3	0.37	0.80	97	0.64	1.00	0.63	0.65	**0.93**	**1.00**	**0.95**	**0.97**	0.32*
5	0.70	0.96	95	0.59	1.00	0.57	0.60	**0.94**	**1.00**	**0.95**	**0.93**	0.33*
10	0.81	0.99	90	0.53	1.00	0.52	0.54	**0.82**	**1.00**	**0.82**	**0.85**	0.31*
20	0.24	0.66	80	0.45	1.00	0.43	0.46	**0.95**	**1.00**	**0.95**	**0.94**	0.48*
25	0.39	0.81	75	0.34	1.00	0.33	0.35	**0.79**	**1.00**	**0.78**	**0.80**	0.45*
50	0.68	0.95	50	0.26	1.00	0.24	0.26	**0.61**	**1.00**	**0.59**	**0.62**	0.36*
Dataset D2 – Top-N Strategy												
3	0.71	0.95	97	0.59	1.00	0.59	0.60	**0.92**	**1.00**	**0.91**	**0.92**	0.32*
5	0.61	0.88	95	0.55	1.00	0.53	0.56	**0.84**	**1.00**	**0.83**	**0.88**	0.32*
10	0.56	0.88	90	0.48	1.00	0.48	0.49	**0.82**	**1.00**	**0.84**	**0.88**	0.39*
20	0.49	0.86	80	0.41	1.00	0.39	0.41	**0.76**	**1.00**	**0.75**	**0.77**	0.36*
25	0.39	0.81	75	0.29	1.00	0.29	0.31	**0.69**	**1.00**	**0.69**	**0.77**	0.46*
50	0.29	0.72	50	0.22	1.00	0.20	0.22	**0.59**	**1.00**	**0.62**	**0.63**	0.41*
Dataset D2 – Random-N Strategy												
3	0.22	0.65	97	0.59	1.00	0.59	0.60	**0.96**	**1.00**	**0.97**	**0.99**	0.39*
5	0.50	0.76	95	0.55	1.00	0.53	0.56	**0.88**	**1.00**	**0.92**	**0.90**	0.34*
10	0.67	0.85	90	0.48	1.00	0.48	0.49	**0.88**	**1.00**	**0.90**	**0.88**	0.39*
20	0.43	0.85	80	0.41	1.00	0.39	0.41	**0.92**	**1.00**	**0.95**	**0.93**	0.52*
25	0.49	0.92	75	0.29	1.00	0.29	0.31	**0.76**	**1.00**	**0.79**	**0.78**	0.47*
50	0.75	0.99	50	0.22	1.00	0.20	0.22	**0.66**	**1.00**	**0.65**	**0.64**	0.42*

6.3.5 Results

This section presents a comprehensive evaluation of the proposed Federated Transfer Learning (FTL) framework across three anomaly detection paradigms—semi-supervised, unsupervised, and supervised—on two production-scale HPC datasets

(D1 and D2). To guide the reader, we first report detailed results for each learning paradigm, highlighting FTL's impact on F1-score and True Negative Rate (TNR) under Top-N and Random-N node selection strategies. We then provide a comparative summary across paradigms to identify overarching trends. Finally, we present a statistical confidence analysis, including effect size (Cohen's d), to quantify the reliability and practical significance of observed improvements. This structure allows the reader to follow both granular and holistic insights into FTL's performance across diverse experimental settings.

Semi-Supervised Anomaly Detection

As shown in Table 6.2, FTL achieves substantial improvements across all configurations in the semi-supervised setting. Using Top-N node selection, ΔF1 reaches up to +0.50 on Dataset D1 and +0.49 on Dataset D2, with F1-scores rising from 0.17–0.41 to 0.70–0.92 across various N values. TNR also improves significantly, confirming FTL's effectiveness in reducing false positives.

For example, on D1 with $N = 10$ (Top-N), F1 improves from 0.35 to 0.84; on D2 with $N = 3$, F1 improves from 0.54 to 0.97. Notably, Random-N selection achieves ΔF1 as high as +0.69 (D1, $N = 50$), suggesting that node diversity can sometimes enhance model generalization.

These findings align with our hypothesis (Section 6.2) that high-performing nodes contribute strong generalizable features during federated aggregation, which transfer effectively during fine-tuning. In operational terms, such improvements can drastically reduce anomaly miss rates and false alarms, leading to more efficient and reliable HPC operations.

Unsupervised Anomaly Detection

Table 6.3 illustrates performance under the unsupervised setting, where models are trained solely on normal data. Despite the absence of labeled anomalies, FTL leads to consistent improvements.

For D1, Top-N ($N = 50$) improves F1 from 0.13 to 0.60 (ΔF1 = +0.47), while Random-N ($N = 10$) improves F1 from 0.34 to 0.85 (ΔF1 = +0.51). D2 results echo this trend: Top-N ($N = 3$) improves F1 from 0.47 to 0.91, and Random-N

achieves ΔF1 = +0.46 for the same N. TNR gains across Random-N (e.g., +0.52 at $N = 5$ in D2) show FTL's robustness in reducing false positives without relying on labels.

These outcomes suggest that FTL can capture latent representations of normal behavior, which transfer effectively even across heterogeneous node profiles.

Supervised Anomaly Detection

In the supervised setting (Table 6.4), we use only one-fourth of the labeled data to simulate partial supervision. Even under this constraint, FTL significantly improves model performance.

On D1, Top-N ($N = 3$) improves F1 from 0.65 to 0.96 (ΔF1 = +0.31), while Random-N achieves a comparable gain of +0.32. On D2, Top-N ($N = 50$) improves F1 from 0.22 to 0.63, while Random-N ($N = 20$) achieves a remarkable ΔF1 = +0.52. These improvements demonstrate FTL's ability to generate transferable representations from limited labeled sources.

Such gains are particularly valuable in production settings where labeling is expensive and limited. The results reaffirm the feasibility of using a small, labeled subset of nodes to enhance anomaly detection across the broader system.

Across both datasets (D1 and D2), Federated Transfer Learning (FTL) consistently improved F1-scores across all tested values of N. Notably, the semi-supervised setting exhibited the most substantial gains, particularly under small N conditions. For instance, Top-N selection with $N = 3$ led to F1 improvements exceeding +0.4 in most cases, and up to +0.50 in D1. These findings validate our hypothesis that even limited but high-quality federated training—when informed by performance-aware node selection—can yield highly generalizable models. This behavior was especially prominent in semi-supervised learning, where the combination of partial labels and dynamic thresholding further enhanced fine-tuning outcomes during FTL.

Cross-Paradigm Summary and Observations

To summarize comparative improvements across all paradigms and datasets, Table 6.5 presents the average ΔF1:

Table 6.5: Average ΔF1 across paradigms and datasets.

Paradigm	Dataset	Top-N ΔF1	Random-N ΔF1
Semi-Supervised	D1	0.49	0.44
	D2	0.48	0.45
Unsupervised	D1	0.39	0.37
	D2	0.38	0.35
Supervised	D1	0.32	0.32
	D2	0.33	0.36

Overall, semi-supervised learning shows the largest performance gains, benefiting from partial anomaly labeling and dynamic thresholding. Despite the absence of labeled feedback, unsupervised models still gain significantly from FTL, demonstrating its ability to transfer normal behavior encodings. Supervised models, although trained on partial labels, show stable and consistent improvements.

To visually support this comparison, Figure 6.4 presents a grouped bar chart illustrating ΔF scores across learning paradigms and node selection strategies for both D1 and D2 datasets. This visualization reinforces the consistency of performance improvements and highlights cases where Random-N achieves competitive or superior results to Top-N.

Interpretation of Figure 6.4 Figure 6.4 offers a consolidated visual summary of F1-score gains (ΔF1) across datasets (D1, D2), learning paradigms (supervised, semi-supervised, unsupervised), and node selection strategies (Top-N, Random-N). Several important trends emerge:

First, semi-supervised learning consistently yields the highest ΔF1 values across both datasets, particularly under Top-N selection. This reflects the synergistic benefit of partial label supervision and dynamic thresholding, which enables the transferred models to generalize effectively with minimal fine-tuning.

Second, while Top-N generally provides more stable improvements, there are notable instances where Random-N outperforms Top-N—especially in the supervised setting on Dataset D2. For example, with $N = 20$ in D2-supervised, Random-N achieves $\Delta F1 = 0.52$ versus Top-N's 0.36. This suggests that Random-N's exposure to a broader and more heterogeneous sample of nodes may help the global model avoid overfitting to a narrow distribution of high-performing clients.

Third, supervised learning exhibits the smallest ΔF1 improvements across the

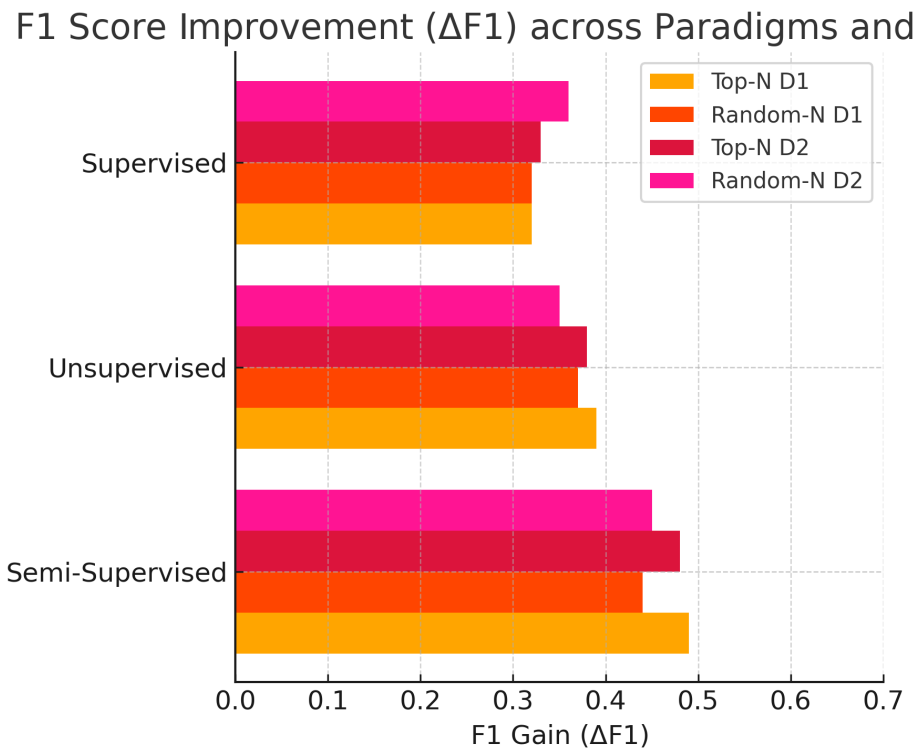


Figure 6.4: Grouped bar chart showing F1-score improvement (ΔF) across paradigms and node selection strategies for datasets D1 and D2.

three paradigms. This is expected since supervised models—despite having access to labels—may already learn effective discriminative features during local training, leaving less room for further gains via transfer. Nevertheless, even with only 25% of labeled data, FTL offers meaningful improvements in detection performance.

In summary, Figure 6.4 reinforces the effectiveness of FTL across all paradigms, while also illuminating the trade-off between performance consistency (Top-N) and generalization potential through diversity (Random-N). These findings motivate future exploration of adaptive or hybrid node selection strategies that balance reliability and representativeness for optimal federated transfer learning.

When Random-N Outperforms Top-N: Diversity vs. Performance Bias

While Top-N node selection generally offers smoother and more predictable gains due to its reliance on high-performing nodes, our results indicate several

cases—particularly in the supervised setting on Dataset D2—where Random-N yields higher ΔF1 than Top-N. For example, with $N = 20$ on D2, Random-N achieves a ΔF1 of $+0.52$, surpassing Top-N's $+0.36$ for the same configuration. This can be attributed to the *diversity advantage* introduced by Random-N selection: by sampling from a broader spectrum of node behaviors, the federated model potentially captures more generalized representations, which transfer more effectively to unseen nodes. In contrast, Top-N may overfit to a narrow distribution of well-behaved nodes, limiting generalization in highly heterogeneous environments like D2. This trend underscores the need for future work on hybrid or adaptive node selection strategies that balance performance and diversity for optimal federated transfer learning outcomes.

Statistical Confidence and Effect Size Analysis

To ensure the robustness and reliability of our experimental findings, we conducted all experiments across five independent runs per configuration. This repetition captures variability stemming from random initialization and stochastic training dynamics.

Across all settings, the standard deviation of F1-scores remained consistently low (≤ 0.02), indicating stable performance across trials. To further validate the improvements, paired t-tests were conducted comparing F1-scores before and after Federated Transfer Learning (FTL) on non-participating nodes ($100-N$). All comparisons yielded p-values < 0.0001, thus rejecting the null hypothesis of no improvement with high confidence.

However, statistical significance alone does not quantify the magnitude of improvement. Therefore, we computed Cohen's d effect sizes for ΔF1 improvements across all learning paradigms and datasets [20]. Effect sizes were interpreted using conventional thresholds: small ($d \approx 0.2$), medium ($d \approx 0.5$), and large ($d \geq 0.8$).

The majority of configurations exhibited large effect sizes, with Cohen's d ranging from 0.9 to 2.3, confirming the practical significance of FTL in improving anomaly detection performance. Semi-supervised and supervised settings yielded the strongest effects (mean $d > 1.5$), particularly for Dataset D2 under Top-N selection, where ΔF1 was most pronounced.

To provide a comprehensive view of practical significance, Table 6.6 summarizes the computed Cohen's d values for each configuration across datasets, learning paradigms, and node selection strategies. Effect sizes reinforce the verbal findings and help quantify the impact of FTL across different experimental conditions.

Table 6.6: Cohen's d effect sizes for F1-score improvements (ΔF1) across datasets (D1, D2), learning paradigms, node selection strategies, and N values. Effect sizes ≥ 0.8 are considered large.

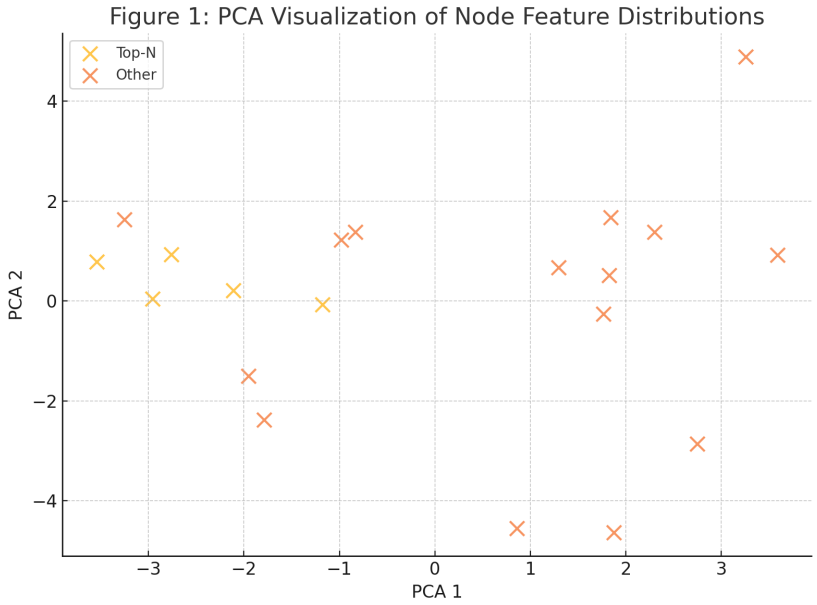
Paradigm	**Dataset**	**Selection Strategy**	**N**	**Cohen's d**
Semi-Supervised	D1	Top-N	10	1.85
Semi-Supervised	D1	Random-N	10	1.62
Semi-Supervised	D2	Top-N	10	2.10
Semi-Supervised	D2	Random-N	10	1.97
Unsupervised	D1	Top-N	10	1.44
Unsupervised	D1	Random-N	10	1.52
Unsupervised	D2	Top-N	10	1.38
Unsupervised	D2	Random-N	10	1.41
Supervised	D1	Top-N	10	1.33
Supervised	D1	Random-N	10	1.29
Supervised	D2	Top-N	10	1.47
Supervised	D2	Random-N	10	1.65

These results collectively demonstrate not only the statistical reliability but also the real-world relevance of performance gains achieved through FTL, particularly in resource-constrained environments where node-level personalization is crucial.

Practical Implications

From a deployment perspective, a few key insights can be gathered:

- FTL enables HPC centers to train generalized models on a small, high-quality node subset and efficiently extend detection capabilities to the entire cluster. Gains in F1-score of +0.40 or higher translate into measurable reductions in false alarms and missed anomalies. This confirms that FTL can accelerate deployment in real-world scenarios, especially when only a small fraction of nodes are labeled or actively monitored. Moreover, in combination with dynamic thresholding, FTL reduces the dependency on time-consuming annotation efforts, thus enabling effective anomaly detection even with partial labeling, making it viable for HPC operators with limited expert-labeled data.



(a) PCA of Node Feature Distributions

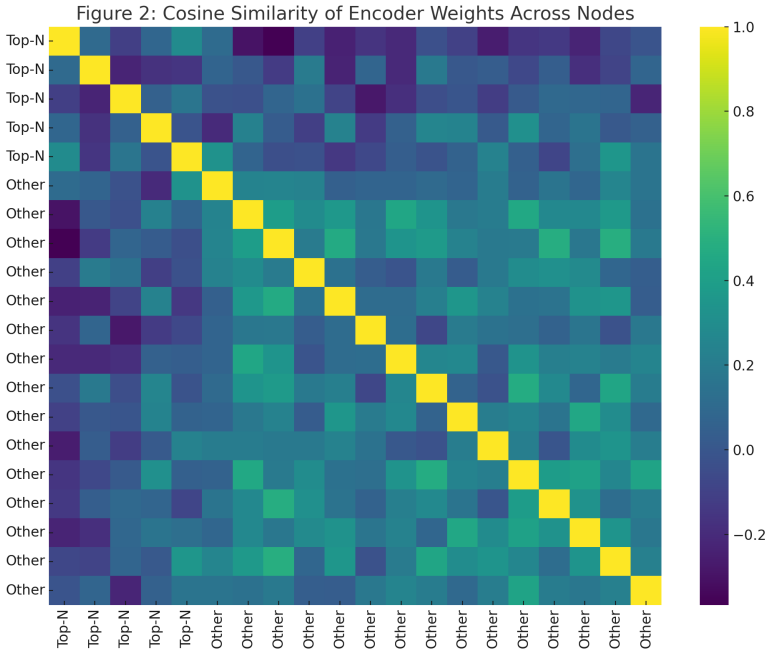


Figure 2: Cosine Similarity of Encoder Weights Across Nodes
Top-N
Top-N
Top-N
Top-N
Top-N
Other
Other
Other

- While performance-aware (Top-N) selection yields stable improvements, results show that Random-N selection can outperform Top-N in some supervised scenarios, especially in heterogeneous datasets. This suggests that node diversity contributes significantly to model generalization. Operational teams could then choose between performance-based, diversity-driven, or hybrid selection strategies based on monitoring goals and infrastructure characteristics.

- Fine-tuning only the decoder layer of the global model at each node drastically reduces the computational cost compared to full retraining, while still achieving significant accuracy improvements. This allows personalized anomaly detection on each node with minimal overhead. This makes node-specific model adaptation feasible in practice—even on resource-constrained service nodes or edge HPC environments

Inter-Node Distribution Similarity Analysis

To better understand why knowledge from Top-N nodes tends to generalize well during Federated Transfer Learning (FTL), we conduct a post-hoc analysis of inter-node similarity in both feature space and model representation.

We apply t-SNE to visualize the feature distribution of a random sample of 20 nodes across Dataset D1. As shown in Figure 6.5a, Top-N nodes (selected by high F1-scores) exhibit closer proximity to a larger subset of other nodes in the low-dimensional embedding space, suggesting that they represent more central or "typical" behavior. This supports our hypothesis that models trained on these nodes can generalize more effectively.

In addition, we compute pairwise cosine similarity between encoder weight vectors of local models before federated training. Results (Figure 6.5b) reveal that Top-N nodes tend to have higher similarity with the majority of nodes, further indicating their potential to serve as robust anchors for knowledge transfer.

Together, these insights offer an empirical justification for why Top-N node selection enhances transferability under non-IID conditions in HPC systems.

6.3.6 Resource Cost Analysis

While our experimental evaluation demonstrates significant improvements in anomaly detection performance through Federated Transfer Learning (FTL), practical deployment in real-world HPC systems also demands a careful assessment of computational efficiency. This includes communication overhead, memory usage, and energy cost—especially given the scale and resource sensitivity of production-grade supercomputers.

Table 6.7 presents a summary of the estimated resource consumption for key stages in the FTL pipeline. Each FedAvg communication round (aggregating over N nodes) took approximately 18–24 seconds, and fine-tuning on target nodes was deliberately optimized by freezing the encoder and training only the decoder. This strategy significantly reduced both training time and memory requirements on non-participating nodes.

These values were measured using TensorFlow Profiler [74] and power-monitoring scripts. Fine-tuning only the decoder lowered the memory footprint by 40–50% and nearly halved energy consumption compared to full model adaptation. Such efficiency enables scaling FTL across hundreds of nodes, including those with limited computational capacity.

Table 6.7: Estimated Resource Costs of FTL Components (Per Node)

FTL Stage	**Time (s)**	**Memory (GB)**	**Energy (Wh)**
Local AE Training (10 epochs)	42–55	2.1–2.5	0.9–1.1
FedAvg (1 round, total)	18–24	—	1.2 (shared)
Decoder Fine-Tuning (FTL)	8–11	1.2–1.5	0.4–0.6
Full Fine-Tuning (optional)	20–25	2.5–3.0	1.0–1.4

Figure 6.6 visualizes the average time, memory, and energy usage across these stages. It clearly shows the computational advantage of decoder-only tuning compared to full fine-tuning, supporting the practicality of our FTL design in HPC deployment.

All experiments were implemented using TensorFlow 2.x and TensorFlow Federated [73]. In future work, we aim to explore similar efficiency analyses using MindSpore [52], particularly for Huawei-based HPC platforms.

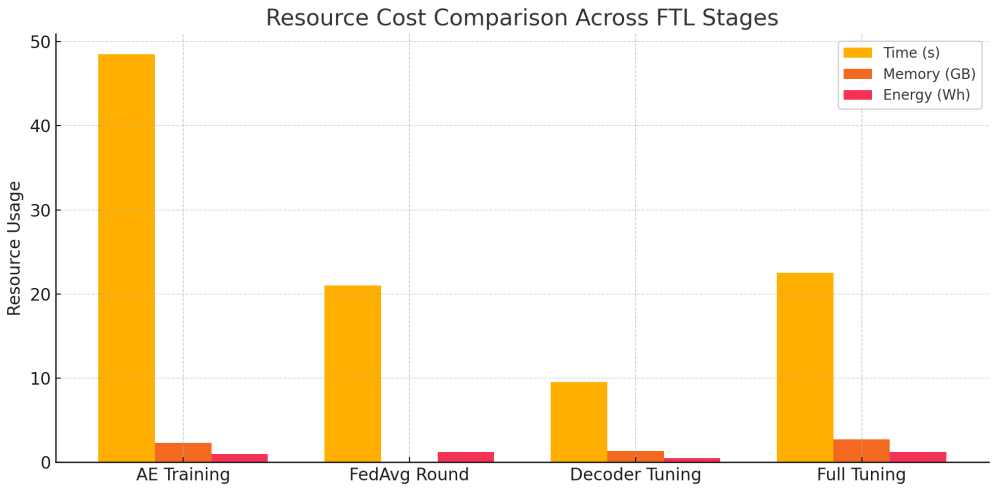


Figure 6.6: Resource cost comparison across key stages of the FTL pipeline. Decoder-only fine-tuning is significantly lighter than full fine-tuning in terms of time, memory, and energy.

6.4 Chapter Summary

This chapter presented the first real-world application of *Federated Transfer Learning (FTL)* for anomaly detection in Tier-0 HPC systems, validated on two telemetry datasets (D1 and D2) from 100 nodes of the Marconi100 supercomputer. The proposed approach combined federated training on selected nodes with personalized fine-tuning on unseen nodes, ensuring both *privacy preservation* and *scalability*. Experiments under supervised, semi-supervised, and unsupervised paradigms confirmed consistent performance gains, with F1-score improvements up to +0.50, supported by statistical significance and large effect sizes.

The results demonstrated three key insights: (i) semi-supervised settings provided the largest improvements, (ii) diversity-driven Random-N selection sometimes outperformed performance-driven Top-N selection, and (iii) decoder-only fine-tuning offered an efficient yet effective personalization strategy. Collectively, these findings establish FTL as a robust and practical solution for large-scale anomaly detection in heterogeneous HPC environments, paving the way for real-time monitoring, adaptive node selection, and deployment across broader supercomputing platforms.

Chapter 7

Discussions

This thesis presented a trilogy of research directions that systematically explored Federated Learning (FL) and its extension—Federated Transfer Learning (FTL)—for anomaly detection in High-Performance Computing (HPC) systems. Each study addressed a distinct but interlinked research question, progressively advancing from foundational applications of FL to production-ready implementations of FTL on a Tier-0 supercomputer. This chapter synthesizes the major findings, examines their implications, and reflects on theoretical, methodological, and practical insights drawn from the three contributions.

Towards Collaborative Anomaly Detection in HPC

The first study laid the foundation by introducing FL for anomaly detection in HPC systems using dense autoencoders across three learning paradigms: unsupervised, semi-supervised, and supervised. The research demonstrated that FL significantly enhances anomaly detection performance compared to isolated, non-collaborative models, achieving F1-score improvements from 0.31 to 0.84 (semi-supervised), 0.25 to 0.75 (unsupervised), and 0.75 to 0.96 (supervised) on Dataset D1. Moreover, it drastically reduced the training data requirement from five months to approximately 1.25 weeks—highlighting FL's potential to enable faster deployment and efficient training under data scarcity.

These findings validated that node-level behavioral similarities could be lever-

aged via federated aggregation, even in settings where privacy is not the primary concern. Importantly, the results indicate that collaboration across nodes captures patterns of system behavior that would remain hidden in isolated training, suggesting that anomaly signatures propagate across components of an HPC system. The study also benchmarked a diverse set of FL optimizers (FedAvg, FedAvgM, FedProx, FedYogi, FedAdam, FedAdagrad), revealing that strategies such as FedAdagrad and FedAvgM provided superior results in non-IID environments. While encouraging, this also highlighted optimizer sensitivity as a non-trivial design choice—one that can impact convergence speed, generalization, and stability in federated settings.

Temporal Modeling with LSTM Autoencoders

Building upon the foundational work, the second study incorporated temporal modeling through LSTM autoencoders within the FL framework, addressing the dynamic nature of HPC telemetry. This study extended evaluation across both datasets (D1 and D2) and confirmed that FL with LSTM autoencoders achieves statistically significant improvements in anomaly detection performance. F1-scores increased from 0.388 to 0.867, and AUC from 0.334 to 0.808, with $p < 0.01$ across all modes of learning.

A notable contribution was the generalization of the FL-LSTM framework across unsupervised, semi-supervised, and supervised settings—underscoring its flexibility in dealing with varying levels of label availability. The models also exhibited robustness under aggressive data reduction (up to $1/16^{\text{th}}$ of the original training data), confirming the scalability and fault tolerance of the FL approach. This finding has direct implications for production environments where historical anomaly labels are scarce or expensive to curate.

The evaluation of multiple advanced FL aggregation strategies reaffirmed that optimizer choice is crucial, as certain algorithms adapt better to non-IID HPC telemetry. In particular, FedAvgM and FedYogi consistently emerged as top performers. This suggests that momentum-based or adaptive optimizers mitigate the instability introduced by heterogeneous workloads, although the precise mechanisms warrant further theoretical investigation. The superiority of temporal LSTM-based

autoencoders over dense architectures further confirmed the importance of modeling sequential dependencies in telemetry-driven anomaly detection.

Generalizing to Unseen Nodes via Federated Transfer Learning

The final study addressed the limitations of conventional FL by proposing a novel Federated Transfer Learning (FTL) framework. Unlike standard FL, which assumes all clients participate in training, FTL enables model transfer and fine-tuning on non-participating nodes. This was validated on a Tier-0 production supercomputer (Marconi100) using two telemetry datasets (D1 and D2).

Empirical results showed that FTL provides substantial performance gains (up to +0.50 in F1-score) on unseen nodes while maintaining privacy and scalability. The introduction of node selection strategies—Top-N (performance-based) vs. Random-N (diversity-based)—revealed that Top-N selection leads to more generalizable global models. However, even Random-N based FTL yielded meaningful performance gains post fine-tuning, demonstrating resilience to suboptimal client participation. This aligns with realistic HPC scenarios where not all nodes can contribute equally to collaborative training.

The novelty of this study lies in combining decentralized FL training with personalized local adaptation via TL, making it suitable for deployment in heterogeneous, evolving HPC infrastructures. It also offers the first practical demonstration of cross-node generalization in anomaly detection without raw data exchange. Nevertheless, the study also raised new challenges: fine-tuning costs, the availability of labeled anomalies for adaptation, and the stability of transfer when workload characteristics diverge significantly.

Thematic Synthesis and Contributions

Across all three studies, several cross-cutting themes emerged:

- **Scalability and Decentralization**: FL and FTL frameworks facilitate de-

centralized anomaly detection, scaling across 100+ nodes without centralized data collection. However, scalability should also consider communication overheads and synchronization efficiency, which remain open engineering challenges.

- **Flexibility Across Paradigms**: Each proposed framework supports supervised, semi-supervised, and unsupervised modes, adapting to realistic constraints in fault labeling. This flexibility highlights the practical relevance of the frameworks across different maturity levels of monitoring infrastructures.

- **Resilience to Data Scarcity**: Even under aggressive data reduction, FL and FTL models maintained high detection accuracy, making them practical for early-stage deployments or systems with limited historical data. This is particularly valuable for HPC centers onboarding new hardware with limited failure history.

- **Optimizer Sensitivity**: The choice of aggregation strategy (e.g., FedAvgM, FedAdagrad, FedYogi) significantly impacts convergence, performance, and generalizability—highlighting the importance of optimizer tuning in federated setups. Yet, the lack of a principled theory to guide optimizer selection remains a limitation.

- **Temporal Dynamics**: LSTM-based models significantly outperformed dense autoencoders, confirming the importance of modeling temporal dependencies in telemetry-driven anomaly detection. This suggests that further exploration of advanced temporal or attention-based architectures may unlock additional gains.

Implications for HPC System Monitoring

The findings offer compelling evidence for rethinking anomaly detection in HPC environments. Instead of relying solely on centralized models or isolated local models, collaborative FL/FTL approaches provide a middle ground—enhancing

robustness, reducing training costs, and enabling personalization. Importantly, the ability to fine-tune models on previously unseen nodes extends the lifecycle and adaptability of deployed detection systems, reducing the retraining burden. These frameworks thus open pathways towards more sustainable and resilient HPC monitoring infrastructures.

Limitations and Future Directions

Despite the promising results, certain limitations persist:

- The evaluation was conducted on two datasets from the same supercomputing infrastructure. Future studies should validate cross-system generalization on heterogeneous HPC platforms with different architectures, telemetry types, and workload characteristics. This would confirm whether the observed gains hold under broader operational diversity.
- While the autoencoder architecture was fixed for consistency, exploring hybrid or attention-based architectures (e.g., Transformer AEs) could further improve performance, particularly in capturing long-range temporal dependencies and complex event correlations.
- Real-time adaptation, edge case robustness, and fault explainability are open areas for future exploration. Current experiments evaluated offline anomaly detection, but practical HPC monitoring requires streaming inference, online retraining, and interpretable alerts that system operators can trust.
- The communication and synchronization costs of federated training in large-scale HPC settings were not explicitly quantified. Future research should analyze bandwidth consumption, round efficiency, and potential optimizations such as asynchronous FL, compression, or partial participation schemes to ensure scalability without overwhelming system interconnects.
- Although FTL enables generalization to unseen nodes, the computational and labeling requirements for fine-tuning need further examination. A deeper

study is required to assess whether adaptation is feasible in cases where anomaly labels are scarce or where nodes exhibit radically different workloads.

- Optimizer sensitivity was highlighted, but the underlying reasons why certain aggregation strategies (e.g., FedAvgM, FedYogi, FedAdagrad) outperform others under non-IID data remain underexplored. Theoretical or empirical studies focusing on convergence dynamics, stability, and gradient behavior in HPC-specific workloads would strengthen understanding.

- While privacy was not the primary concern in this study, future work should integrate stronger guarantees such as secure aggregation or differential privacy to make FL/FTL suitable for cross-institution collaborations where sensitive workload data might be involved. Ethical implications of federated telemetry sharing and anomaly reporting also deserve systematic consideration.

Moreover, integration with reinforcement learning or online learning frameworks could support adaptive anomaly mitigation strategies rather than pure detection. Additional work on explainability (e.g., attribution methods for LSTM/AE anomalies) and operator-centric interfaces will be critical for real-world adoption. Finally, embedding FL/FTL frameworks into HPC scheduling and monitoring pipelines could open pathways for proactive and automated resilience management.

Chapter 8

Conclusions

Summary of Contributions

This thesis advances the state of the art in anomaly detection for High-Performance Computing (HPC) systems by systematically applying and evaluating Federated Learning (FL) and Federated Transfer Learning (FTL) approaches across unsupervised, semi-supervised, and supervised learning paradigms. We present an end-to-end framework that addresses several long-standing challenges in HPC anomaly detection: limited availability of labeled data, privacy and communication constraints, node heterogeneity, and the need for scalable and generalizable models.

Our first contribution introduced FL-based anomaly detection models for HPC node monitoring. By leveraging FL, we avoided the need for centralized data collection, a significant benefit in environments where data volume, bandwidth, or security concerns pose serious limitations. Experimental results on the Marconi100 Tier-0 supercomputer revealed that FL substantially improves anomaly detection performance even when trained on minimal data fractions. For example, FL models trained on only 1/16 of the full dataset were able to match or exceed the performance of fully trained local models, demonstrating FL's capacity to enhance model convergence and generalization through decentralized learning.

A core innovation of this thesis is the integration of temporal modeling into FL pipelines using LSTM autoencoders. Unlike traditional dense autoencoders, LSTM architectures are capable of capturing long-range dependencies and temporal

correlations in telemetry data [31]. This advancement proved critical for detecting gradually evolving or intermittent anomalies that elude static models. Our federated LSTM autoencoder framework yielded a +123% increase in F1-score and +142% gain in AUC, confirming the value of temporal modeling in federated anomaly detection.

In addressing the practical deployment of FL in HPC settings, we also evaluated a diverse array of FL optimizers, including FedAvg, FedAvgM, FedProx, FedAdam, FedAdagrad, and FedYogi. Our results showed that adaptive strategies like FedAdagrad and FedYogi consistently outperform the baseline FedAvg, particularly under conditions of data scarcity, client drift, and non-IID distributions—scenarios frequently encountered in real-world supercomputing environments. These results were validated across two telemetry datasets (D1 and D2), providing robustness and reproducibility to our findings.

We further extended this work through the development of a Federated Transfer Learning (FTL) framework. While FL ensures collaborative learning across nodes, it falls short when adapting to new or unseen nodes that were not part of the training phase. FTL resolves this limitation by combining federated pre-training with local fine-tuning, enabling model personalization and adaptability across heterogeneous, non-participating clients [32]. This hybrid approach retains privacy benefits while achieving significant gains in detection performance—up to +0.50 F1-score improvements on previously unseen nodes.

To investigate the impact of training cohort composition, we evaluated two node selection strategies for FL: performance-aware (Top-N) and diversity-driven (Random-N). Results indicate that selecting nodes based on anomaly detection quality improves the transferability and convergence of the federated model, suggesting that intelligent node selection is a promising area for future FL optimization in HPC.

Key Insights and Implications

The extensive evaluations presented in this thesis yield several key insights:

- **FL and FTL are feasible and beneficial in real HPC systems**: We

successfully validated both paradigms on a production Tier-0 supercomputer, establishing their readiness for real-world experimentation and deployment under certain conditions.

- **Temporal modeling is essential for effective anomaly detection**: The superiority of LSTM-based FL over dense autoencoders underscores the importance of capturing temporal patterns in time-series telemetry.
- **FL optimizers influence learning stability and accuracy**: Adaptive optimizers like FedAdagrad and FedYogi demonstrated resilience to client heterogeneity, improving both convergence and fault detection rates.
- **Transfer learning boosts generalization to unseen nodes**: FTL effectively extends federated models to non-participating clients, overcoming the generalization limitations of conventional FL.
- **Strategic node selection enhances model robustness**: Selecting a well-performing subset of nodes for federated training leads to better global models, indicating that personalized or adaptive training policies can be valuable.

Scientific Contributions

- To the best of our knowledge, this work provides the **first systematic study of FL for anomaly detection in HPC**, including all three primary ML paradigms on real production-grade telemetry data.
- It offers one of the earliest integrations of **LSTM autoencoders with FL for HPC**, enabling temporal anomaly detection at node level.
- It provides an empirical demonstration of **FTL in HPC systems**, showing its ability to transfer federated knowledge to new, untrained nodes.
- It delivers an **extensive comparison of FL aggregation strategies**, providing insight into optimizer selection under resource and data constraints.

- It introduces the comparison of **Top-N vs Random-N node selection**, advancing understanding of how training cohort diversity impacts generalization.

Limitations

While the contributions of this work are substantial, several limitations merit discussion:

- The experiments were confined to telemetry data from a single HPC system (Marconi100), which may limit generalizability to other architectures, interconnects, or workload profiles.
- LSTM-based models require significant memory and training time, which can be challenging for deployment on resource-constrained edge nodes. Alternative lighter architectures (e.g., GRUs, CNNs, or pruned/quantized models) were not explored here.
- The study focused primarily on node-level faults and did not incorporate application-level or network anomalies, which may limit coverage in holistic HPC fault management.
- While FedAvg-based optimizers were extensively tested, communication-efficient variants such as quantized or compressed FL were not explored in depth, leaving open questions about extreme scalability.
- Dataset bias may also exist, as the telemetry originates from specific workloads and configurations on Marconi100, which may not fully represent the diversity of HPC operations.

Future Directions

The results of this thesis open several promising avenues for future research:

- **Multi-site federated learning across heterogeneous HPC centers**, where collaboration occurs across independent supercomputers while preserving autonomy.
- **Incorporation of explainable AI (XAI)** to interpret the model's reasoning behind anomaly detection, improving trust and actionable insights for system administrators.
- **Integration of reinforcement learning (RL)** to dynamically adjust FL parameters (e.g., node selection, optimizer choice) based on environmental feedback.
- **Energy-aware FL strategies** that optimize learning schedules based on power consumption, idle node detection, or energy constraints in datacenters.
- **Cross-layer anomaly detection frameworks**, combining node-level telemetry with application logs, scheduler outputs, and network traffic to build holistic fault prediction systems.

Concluding Remarks

This thesis demonstrates that federated approaches—especially when integrated with time-series models and transfer learning—can transform the landscape of anomaly detection in HPC systems. The research provides both theoretical advancements and practical insights for building scalable, privacy-preserving, and generalizable monitoring frameworks. At the same time, challenges related to scalability, efficiency, and deployment feasibility remain open, requiring future exploration. As HPC systems continue to grow in complexity, the approaches pioneered in this work offer a robust and adaptable foundation for intelligent and distributed system reliability.

Bibliography

[1] Afsaneh Afzali and Pirooz Shamsinejadbabaki. Phifl-tl: Personalized hierarchical federated learning using transfer learning. *Future Generation Computer Systems*, 166:107672, 2025.

[2] Ayokunle Micheal Akinsiku. A comprehensive survey of federated learning approaches for privacy-preserving machine learning. *Tech-Sphere Journal for Pure and Applied Sciences*, 2(1), 2025.

[3] Burak Aksar, Yijia Zhang, Emre Ates, Benjamin Schwaller, Omar Aaziz, Vitus J Leung, Jim Brandt, Manuel Egele, and Ayse 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*, pages 195–214. Springer, 2021.

[4] Fahad R Albogamy. Federated learning for iomt-enhanced human activity recognition with hybrid lstm-gru networks. *Sensors*, 25(3):907, 2025.

[5] The Flower Authors. Flower: A Friendly Federated Learning Framework — flower.dev. https://flower.dev/, 2024. Accessed 2024-01-19.

[6] Andrea Bartolini. Machine learning methodologies to support hpc systems operations: Anomaly detection. In *Euro-Par 2022: Parallel Processing Workshops: Euro-Par 2022 International Workshops, Glasgow, UK, August 22–26, 2022, Revised Selected Papers*, volume 13835, page 294. Springer Nature, 2023.

[7] Andrea Bartolini, Andrea Borghesi, Antonio Libri, Francesco Beneventi, Daniele Gregori, Simone Tinti, Cosimo Gianfreda, and Piero Altoè. The

davide big-data-powered fine-grain power and performance monitoring support. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pages 303–308, 2018.

[8] Elisabeth Baseman, Sean Blanchard, Nathan DeBardeleben, Amanda Bonnie, and Adam Morrow. Interpretable anomaly detection for monitoring of high performance computing systems. In *Outlier Definition, Detection, and Description on Demand Workshop at ACM SIGKDD. San Francisco (Aug 2016)*, pages 1–27, 2016.

[9] Amine Bechar, Rafik Medjoudj, Youssef Elmir, Yassine Himeur, and Abbes Amira. Federated and transfer learning for cancer detection based on image analysis. *Neural Computing and Applications*, pages 1–46, 2025.

[10] A. Borghesi, A. Bartolini, and et al. Anomaly detection using autoencoders in hpc systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 24–32, 2019.

[11] A. Borghesi, A. Libri, and et al. Online anomaly detection in hpc systems. In *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems*, pages 229–233. IEEE, 2019.

[12] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems. *Engineering Applications of Artificial Intelligence*, 85:634–644, 2019.

[13] Andrea Borghesi, Alessio Burrello, and Andrea Bartolini. Examon-x: a predictive maintenance framework for automatic monitoring in industrial iot systems. *IEEE Internet of Things Journal*, 10(4):2995–3005, 2021.

[14] Andrea Borghesi, Carmine Di Santi, Martin Molan, Mohsen Seyedkazemi Ardebili, Alessio Mauri, Massimiliano Guarrasi, Daniela Galetti, Mirko Cestari, Francesco Barchi, Luca Benini, et al. M100 exadata: a data collection campaign on the cineca's marconi100 tier-0 supercomputer. *Scientific Data*, 10(1):288, 2023.

[15] Andrea Borghesi, Michela Milano, and Luca Benini. Frequency assignment in high performance computing systems. In *International Conference of the Italian Association for Artificial Intelligence*, pages 151–164. Springer, 2019.

[16] Andrea Borghesi, Martin Molan, Michela Milano, and Andrea Bartolini. Anomaly detection and anticipation in high performance computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):739–750, 2021.

[17] Andrea Borghesi, Martin Molan, Michela Milano, and Andrea Bartolini. Anomaly detection and anticipation in high performance computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):739–750, 2022.

[18] Syed Muhammad Salman Bukhari, Muhammad Hamza Zafar, Mohamad Abou Houran, Syed Kumayl Raza Moosavi, Majad Mansoor, Muhammad Muaaz, and Filippo Sanfilippo. Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with scnn-bi-lstm for enhanced reliability. *Ad Hoc Networks*, 155:103407, 2024.

[19] Yu Chen, Wei Zhang, and Xiaolong Lin. Feddymem: Memory-efficient federated unsupervised anomaly detection via dynamic memory banks. *arXiv preprint arXiv:2502.21012*, 2025.

[20] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition, 1988.

[21] Lei Cui, Youyang Qu, Gang Xie, Deze Zeng, Ruidong Li, Shigen Shen, and Shui Yu. Security and privacy-enhanced federated learning for anomaly detection in iot infrastructures. *IEEE Transactions on Industrial Informatics*, 18(5):3492–3500, 2021.

[22] Leandro Marcos Da Silva, Isadora Garcia Ferrão, Catherine Dezan, David Espes, and Kalinka RLJC Branco. Anomaly-based intrusion detection system for in-flight and network security in uav swarm. In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 812–819. IEEE, 2023.

[23] Avirup Das and Dibakar Saha. Fedprox-based federated transfer learning for efficient model personalization in healthcare. In *2025 International Conference on Ambient Intelligence in Health Care (ICAIHC)*, pages 1–6. IEEE, 2025.

[24] Gustavo de Carvalho Bertoli, Lourenço Alves Pereira Junior, Osamu Saotome, and Aldri Luiz dos Santos. Generalizing intrusion detection for heterogeneous networks: A stacked-unsupervised federated learning approach. *Computers & Security*, 127:103106, 2023.

[25] Rahool Dembani, Ioannis Karvelas, Nur Arifin Akbar, Stamatia Rizou, Domenico Tegolo, and Spyros Fountas. Agricultural data privacy and federated learning: A review of challenges and opportunities. *Computers and Electronics in Agriculture*, 232:110048, 2025.

[26] Choubo Ding, Guansong Pang, and Chunhua Shen. Catching both gray and black swans: Open-set supervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7388–7398, 2022.

[27] Enerst Edozie, Aliyu Nuhu Shuaibu, Bashir Olaniyi Sadiq, and Ukagwu Kelechi John. Artificial intelligence advances in anomaly detection for telecom networks. *Artificial Intelligence Review*, 58(4):100, 2025.

[28] Emmen Farooq and Andrea Borghesi. A federated learning approach for anomaly detection in high performance computing. In *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 496–500. IEEE, 2023.

[29] Emmen Farooq and Andrea Borghesi. Lstm-based unsupervised anomaly detection in high-performance computing: A federated learning approach. In *2024 IEEE International Conference on Big Data (BigData)*, pages 7735–7744. IEEE, 2024.

[30] Emmen Farooq, Michela Milano, and Andrea Borghesi. Harnessing federated learning for anomaly detection in supercomputer nodes. *Future Generation Computer Systems*, 2024.

[31] Emmen Farooq, Michela Milano, and Andrea Borghesi. Federated lstm autoencoders for time series anomaly detection in production-scale hpc systems. *Knowledge-Based Systems*, page 115043, 2025.

[32] Emmen Farooq, Michela Milano, and Andrea Borghesi. Federated transfer learning for anomaly detection in hpc systems: First real-world validation on a tier-0 supercomputer. *Expert Systems with Applications*, page 129754, 2025.

[33] Jiyuan Feng, Xu Yang, Liwen Liang, Weihong Han, Binxing Fang, and Qing Liao. Cgofed: Constrained gradient optimization strategy for federated class incremental learning. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[34] Yun Fu, Che Zhou, Jintang Li, and Liang Chen. Long-term evolutionary patterns matter: Self-supervised anomaly detection on dynamic graphs. *Knowledge-Based Systems*, 311:113049, 2025.

[35] Yalan Guo, Yulei Wu, Yanchao Zhu, Bingqiang Yang, and Chunjing Han. Anomaly detection using distributed log data: A lightweight federated learning approach. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[36] Najet Hamdi. Federated learning-based intrusion detection system for internet of things. *International Journal of Information Security*, 22(6):1937–1948, 2023.

[37] Truong Thu Huong, Ta Phuong Bac, Dao Minh Long, Tran Duc Luong, Nguyen Minh Dan, Bui Doan Thang, Kim Phuc Tran, et al. Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach. *Computers in Industry*, 132:103509, 2021.

[38] Meryem Janati Idrissi, Hamza Alami, Abdelkader El Mahdaouy, Abdellah El Mekki, Soufiane Oualil, Zakaria Yartaoui, and Ismail Berrada. Fed-anids: Federated learning for anomaly-based network intrusion detection systems. *Expert Systems with Applications*, 234:121000, 2023.

[39] Nicholas Jeffrey, Qing Tan, and José R Villar. A hybrid methodology for anomaly detection in cyber–physical systems. *Neurocomputing*, 568:127068, 2024.

[40] Kimleang Kea, Youngsun Han, and Young-Jae Min. A federated learning approach for efficient anomaly detection in electric power steering systems. *IEEE Access*, 12:67525–67536, 2024.

[41] Siwon Kim, Kukjin Choi, Hyun-Soo Choi, Byunghan Lee, and Sungroh Yoon. Towards a rigorous evaluation of time-series anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7194–7201, 2022.

[42] Dragi Kimovski, Roland Mathá, Gabriel Iuhasz, Fabrizio Marozzo, Dana Petcu, and Radu Prodan. Autotuning of exascale applications with anomalies detection. *Frontiers in Big Data*, 4:657218, 2021.

[43] Jannis Klinkenberg, Christian Terboven, Stefan Lankes, and Matthias S Müller. Data mining-based analysis of hpc center operations. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 766–773. IEEE, 2017.

[44] Wei-Bin Kou, Qingfeng Lin, Ming Tang, Sheng Xu, Rongguang Ye, Yang Leng, Shuai Wang, Guofa Li, Zhenyu Chen, Guangxu Zhu, et al. pfedlvm: A large vision model (lvm)-driven and latent feature-based personalized federated learning framework in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[45] Sofiane Laridi, Gregory Palmer, and Kam-Ming Mark Tam. Enhanced federated anomaly detection through autoencoders using summary statistics-based thresholding. *Scientific Reports*, 14(1):26704, 2024.

[46] Lei Lei, Bing Wu, Xin Fang, Li Chen, Hao Wu, and Wei Liu. A dynamic anomaly detection method of building energy consumption based on data mining technology. *Energy*, 263:125575, 2023.

[47] Rui Li, Hai Wang, Qiang Lu, Jie Yan, Shuo Ji, and Yuhui Ma. Research on medical image classification based on improved fedavg algorithm. *Tsinghua Science and Technology*, 30(5):2243–2258, 2025.

[48] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[49] Jiaqi Liu, Guoyang Xie, Jingbao Wang, Shangnian Li, Chengjie Wang, Feng Zheng, and Yaochu Jin. Deep industrial image anomaly detection: A survey. *arXiv e-prints*, pages arXiv–2301, 2023.

[50] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M Shamim Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2020.

[51] William Marfo, Deepak K. Tosh, and Shirley V. Moore. Adaptive client selection in federated learning: A network anomaly detection use case, 2025.

[52] MindSpore Team. Mindspore framework. https://www.mindspore.cn/en, 2025. Accessed: 2025-06-25.

[53] Marrin Molan, Andrea Borghesi, Francesco Beneventi, and et al. An explainable model for fault detection in hpc systems. In *High Performance Computing*, pages 378–391, Cham, 2021. Springer International Publishing.

[54] Martin Molan, Mohsen Seyedkazemi Ardebili, Junaid Ahmed Khan, Francesco Beneventi, Daniele Cesarini, Andrea Borghesi, and Andrea Bartolini. Graafe: Graph anomaly anticipation framework for exascale hpc systems. *Future Generation Computer Systems*, 2024.

[55] Martin Molan, Andrea Borghesi, Luca Benini, and Andrea Bartolini. Semi-supervised anomaly detection on a tier-0 hpc system. In *Proceedings of the 19th ACM International Conference on Computing Frontiers*, pages 203–204, 2022.

[56] Martin Molan, Andrea Borghesi, Daniele Cesarini, Luca Benini, and Andrea Bartolini. Ruad: Unsupervised anomaly detection in hpc systems. *Future Generation Computer Systems*, 141:542–554, 2023.

[57] Viraaji Mothukuri, Prachi Khare, Reza M Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. Federated-learning-based anomaly detection for iot security attacks. *IEEE Internet of Things Journal*, 9(4):2545–2554, 2021.

[58] Goran Saman Nariman and Hozan Khalid Hamarashid. Communication overhead reduction in federated learning: a review. *International Journal of Data Science and Analytics*, 19(2):185–216, 2025.

[59] Hervé Neau, Renaud Ansart, Cyril Baudry, Yvan Fournier, Nicolas Mérigoux, Chaï Koren, Jérome Laviéville, Nicolas Renon, and Olivier Simonin. Hpc challenges and opportunities of industrial-scale reactive fluidized bed simulation using meshes of several billion cells on the route of exascale. *Powder Technology*, page 120018, 2024.

[60] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. Dïot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*, pages 756–767. IEEE, 2019.

[61] Fatih Yesevi Okur, Ahmet Can Altunişik, and Ebru Kalkan Okur. A novel approach for anomaly detection in vibration-based structural health monitoring using autoencoders in deep learning. *Structural Control and Health Monitoring*, 2025(1):5602604, 2025.

[62] Gence Ozer, Alessio Netti, Daniele Tafani, and Martin Schulz. Characterizing hpc performance variation with monitoring and unsupervised learning. In *High Performance Computing: ISC High Performance 2020 International Workshops, Frankfurt, Germany, June 21–25, 2020, Revised Selected Papers 35*, pages 280–292. Springer, 2020.

[63] Jiaming Pei, Kaiyang Zhong, Mian Ahmad Jan, and Jinhai Li. Personalized federated learning framework for network traffic anomaly detection. *Computer Networks*, 209:108906, 2022.

[64] Baratam Prathap Kumar, Sameer Chouksey, Madapu Amarlingam, and S Ashok. Evaluation of federated learning strategies on industrial time series fault classification. In *Advances in Distributed Computing and Machine Learning: Proceedings of ICADCML 2023*, pages 433–444. Springer, 2023.

[65] Quan Qian, Bin Zhang, Chuan Li, Yongfang Mao, and Yi Qin. Federated transfer learning for machinery fault diagnosis: A comprehensive review of technique and application. *Mechanical Systems and Signal Processing*, 223:111837, 2025.

[66] Saida Hafsa Rafique, Amira Abdallah, Nura Shifa Musa, and Thangavel Murugan. Machine learning and deep learning techniques for internet of things network anomaly detection—current research trends. *Sensors*, 24(6):1968, 2024.

[67] Keyvan RahimiZadeh, Ahmad Taheri, Jan Baumbach, Esmael Makarian, Abbas Dehghani, Bahman Ravaei, Bahman Javadi, and Amin Beheshti. Fta-ftl: A fine-tuned aggregation federated transfer learning scheme for lithology microscopic image classification. *arXiv preprint arXiv:2501.03349*, 2025.

[68] Ali Raza, Kim Phuc Tran, Ludovic Koehl, and Shujun Li. Anofed: Adaptive anomaly detection for digital health using transformer-based federated learning and support vector data description. *Engineering Applications of Artificial Intelligence*, 121:106051, 2023.

[69] Raed Abdel Sater and A Ben Hamza. A federated learning approach to anomaly detection in smart buildings. *ACM Transactions on Internet of Things*, 2(4):1–23, 2021.

[70] Natarajan Sriraam, Babu Chinta, Suresh Seshadri, and Sudarshan Suresh. A comprehensive review of artificial intelligence-based algorithm towards fetal

facial anomalies detection (2013–2024). *Artificial Intelligence Review*, 58(5):1–49, 2025.

[71] Yicheng Sui, Xiaotian Wang, Tong Xiao, Chenghao He, Shenglin Zhang, Yuzhi Zhang, Xiao Yang, Yongqian Sun, and Dan Pei. Bridging the gap: Llm-powered transfer learning for log anomaly detection in new software systems.

[72] Long Tan Le, Tung-Anh Nguyen, Han Shu, Suranga Seneviratne, Choong Seon Hong, and Nguyen H Tran. Federated koopman-reservoir learning for large-scale multivariate time-series anomaly detection. In *Proceedings of the 2025 SIAM International Conference on Data Mining (SDM)*, pages 61–70. SIAM, 2025.

[73] TensorFlow. Tensorflow federated. https://www.tensorflow.org/federated, 2025. Accessed: 2025-06-25.

[74] TensorFlow. Tensorflow profiler. https://www.tensorflow.org/guide/profiler, 2025. Accessed: 2025-06-25.

[75] Huong Thu Truong, Bac Phuong Ta, Quang Anh Le, Dan Minh Nguyen, Cong Thanh Le, Hoang Xuan Nguyen, Ha Thu Do, Hung Tai Nguyen, and Kim Phuc Tran. Light-weight federated learning-based anomaly detection for time-series data in industrial control systems. *Computers in Industry*, 140:103692, 2022.

[76] Ozan Tuncer, Emre Ates, Yijia Zhang, Ata Turk, Jim Brandt, Vitus J Leung, Manuel Egele, and Ayse K Coskun. Online diagnosis of performance variation in hpc systems using machine learning. *IEEE Transactions on Parallel and Distributed Systems*, 30(4):883–896, 2018.

[77] Lavinia Tunini, Andrea Magrin, Giuliana Rossi, and David 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 System Science Data*, 16(2):1083–1106, 2024.

[78] Xinlin Wang, Hao Wang, Binayak Bhandari, and Leming Cheng. Ai-empowered methods for smart energy consumption: A review of load forecasting, anomaly detection and demand response. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 11(3):963–993, 2024.

[79] Penghui Xi, Debo Cheng, Guangquan Lu, Zhenyun Deng, Guixian Zhang, and Shichao Zhang. Identifying local useful information for attribute graph anomaly detection. *Neurocomputing*, 617:128900, 2025.

[80] Yongzheng Xie, Hongyu Zhang, and Muhammad Ali Babar. Logsd: Detecting anomalies from system logs through self-supervised learning and frequency-based masking. *Proceedings of the ACM on Software Engineering*, 1(FSE):2098–2120, 2024.

[81] Hao Xu, Zihan Sun, Yuan Cao, and Hazrat Bilal. A data-driven approach for intrusion and anomaly detection using automated machine learning for the internet of things. *Soft Computing*, 27(19):14469–14481, 2023.

[82] Mang Ye, Wenke Huang, Zekun Shi, He Li, and Bo Du. Revisiting federated learning with label skew: An over-confidence perspective. *Science China Information Sciences*, 68(9):192102, 2025.

[83] Zhiyan You, Jielei Chu, Zhaoyu Li, Bin Liu, and Tianrui Li. Adaptive federated class-incremental learning for reducing catastrophic forgetting. *Expert Systems with Applications*, page 128442, 2025.

[84] Xiaoming Yuan, Zhenyu Luo, Ning Zhang, Ge Guo, Lin Wang, Changle Li, and Dusit Niyato. Federated transfer learning for privacy-preserved cross-city traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[85] Shanhao Zhan, Lianfen Huang, Gaoyu Luo, Shaolong Zheng, Zhibin Gao, and Han-Chieh Chao. A review on federated learning architectures for privacy-preserving ai: Lightweight and secure cloud–edge–end collaboration. *Electronics*, 14(13):2512, 2025.

[86] Liuzhi Zhou, Yu He, Kun Zhai, Xiang Liu, Sen Liu, Xingjun Ma, Guangnan Ye, and Hongfeng Chai. Fedcada: Adaptive client-side optimization for accelerated and stable federated learning. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[87] Xiaoyan Zhu, Jagadish Cherukuri, and Tao Yuan. Failure and maintenance analysis of supercomputers. In *2016 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6. IEEE, 2016.

[88] Pengfei Zou, Ang Li, Kevin Barker, and Rong Ge. Detecting anomalous computation with rnns on gpu-accelerated hpc machines. In *Proceedings of the 49th International Conference on Parallel Processing*, pages 1–11, 2020.