Computer Networks 290 (2026) 112775

Contents lists available at ScienceDirect

Computer Networks

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

SereBench: Adversarial robustness benchmarking for security and reliability event classification in networked railway cyber–physical systems☆

Alessandra Rizzardi [a,*], Raffaele Della Corte [b], Jesús F. Cevallos-Moreno [a], Simona De Vivo [b], Sabrina Sicari [a], Domenico Cotroneo [c], Alberto Coen-Porisini [a]

[a] Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, Via O. Rossi 9, Varese, 21100, Italy
[b] Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, Università degli Studi di Napoli Federico II, Via Claudio 21, Naples, 80125, Italy
[c] Department of Computer Science, University of North Carolina at Charlotte, 9201 University City Blvd, Charlotte, NC 28223, USA

ARTICLE INFO

Keywords:
Industrial IoT
Cyber–Physical system
Railway
Adversarial machine learning
Federated learning
Anomaly detection
Robustness benchmarking

ABSTRACT

This paper presents SereBench, an open-source security and reliability benchmarking framework for testing the adversarial robustness and federated resilience of AI-based attack and anomaly classification in networked railway Cyber–Physical Systems (CPS). In this work, SereBench is instantiated and evaluated through a pneumatic braking subsystem case study, where pressure dynamics provide a physically interpretable basis for controlled robustness evaluation. The framework provides a simulation-based environment emulating a railway CPS system, enabling controllable class semantics and verifiable ground truth. This environment supports controlled adversarial robustness benchmarking by progressively stressing the decision boundaries of Machine Learning (ML) models. To this end, SereBench includes a train simulator that produces telemetry, preserving explicit cause–effect relations among signals and operating modes, allowing perturbations to be interpreted as changes in system behavior rather than as arbitrary feature edits. Adversarial robustness benchmarking can be executed in multi-vehicle settings, allowing the study of federated resilience effects, i.e., how robustness-related behavior changes across vehicles under model aggregation. The experimental campaign considers multiple classifier architectures and federated aggregation strategies, targeting the classification of security and reliability events under controlled robustness evaluation conditions. Specifically, the study explores configurations combining federated learning, noise-augmented training, controlled Gaussian boundary stress, and HSJA-based hard-label black-box adversarial evaluation. The experimental campaign also includes a focused FL-flow sensitivity analysis in which controlled message loss and delivery delay are applied to uplink and downlink model exchanges. The obtained results show that SereBench can support adversarial robustness benchmarking of ML models under controlled, braking-oriented railway CPS conditions.

1. Introduction

Railway systems are no longer isolated mechanical and electrical machines. Nowadays, the railway system has been transformed into a **Cyber–Physical System (CPS)** by integrating software components, communication networks, and physical equipment. Under this condition, safety–critical interventions, such as braking, traction control, or train protection, rely on data exchanged among different digital systems. This new architecture offers clear advantages, but also introduces new risks. Indeed, due to connectivity, faults and attacks are no longer limited to local components but can spread through software and networks, impacting the train's behavior [1].

To manage this growing complexity, **Machine Learning (ML)** techniques are increasingly adopted to analyze telemetry streams and support anomaly and attack detection in Cyber–Physical Systems and Industrial IoT environments, particularly for time-series monitoring tasks [2], and have recently been explored also in railway-oriented and federated settings [3,4]. In practice, these systems are fed time-series data from on-board subsystems (e.g., pressure, current, voltage, velocity, or diagnostic signals) that are mapped to discrete decisions. Although classification accuracy is commonly reported as the primary indicator of performance, accuracy alone is not sufficient to validate

☆ This article is part of a Special issue entitled: 'Artificial Intelligence for Security in Networked CPS' published in Computer Networks.

* Corresponding author.
E-mail addresses: alessandra.rizzardi@uninsubria.it (A. Rizzardi), raffaele.dellacorte2@unina.it (R. Della Corte), jf.cevallosmoreno@uninsubria.it (J.F. Cevallos-Moreno), simona.devivo@unina.it (S. De Vivo), sabrina.sicari@uninsubria.it (S. Sicari), d.cotroneo@charlotte.edu (D. Cotroneo), alberto.coenporisini@uninsubria.it (A. Coen-Porisini).

https://doi.org/10.1016/j.comnet.2026.112775
Received 28 February 2026; Received in revised form 18 July 2026; Accepted 8 September 2026
Available online 24 September 2026
1389-1286/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

deployment. This concern is not limited to railway CPS: in the broader networking community, recent surveys on network intrusion detection and federated settings emphasize the shift toward distributed, data-driven pipelines and realistic threat models, stressing robustness-centric evaluation beyond average accuracy [5]. Railway environments are inherently dynamic. Sensors drift, components age, operational modes change, and environmental conditions vary. All these factors modify the statistical properties of the data a detector sees. A model that performs well only under the conditions observed during training may degrade silently in legitimate but unfamiliar situations. For safety–critical systems, this behavior is unacceptable and may lead to severe consequences. In this context, model robustness becomes paramount, namely the ability to preserve reliable decisions when inputs deviate from nominal conditions, whether due to natural variability, faults, or deliberate interference [6,7].

Assessing the **robustness** of an ML model allows us to evaluate the stability of the model when faced with noisy inputs, adversarial attacks, or distribution shifts, and can be used to understand its readiness for real-world deployment. From this perspective, robustness evaluation should consider the *operating conditions* where the target model will be used. This is especially true in contexts, such as railway, where the operating conditions can be affected by the interactions between the nodes composing the system. Therefore, robustness should be assessed in multi-node scenarios that explicitly allow heterogeneous operating conditions, instead of assuming a single uniform data regime. Moreover, operational threat models frequently align with black-box conditions, where an attacker cannot access model parameters or gradients and only observes hard labels. In such settings, robustness failures concentrate near decision boundaries, where small input shifts can flip the predicted class [8]. This makes *boundary-based robustness evaluation* essential: average accuracy can remain high while boundary regions become fragile, leading to overly optimistic robustness claims [9].

In the **railway** domain, evaluating robustness poses another challenge: available datasets are often unsuitable for this task. Trains generate large volumes of telemetry, but the operational semantics of many signals are not always publicly documented or directly available to external evaluators. Status variables might reflect internal logic, vendor-specific settings, or undocumented diagnostics. This means that labels that appear to indicate system states are often weak or not physically meaningful [10]. In these cases, classifiers might separate data well by using quirks from data collection, not real physical patterns [11]. Boundaries based on these quirks have little engineering value. Testing them does not reveal how a detector would perform under real railway conditions, thereby weakening any robustness analysis based on such data.

With this in mind, we present SereBench, an open-source security and reliability benchmarking framework for testing the adversarial robustness and federated resilience of AI-based attack and anomaly classification in networked railway CPS. SereBench uses a train simulator to obtain controllable class semantics and verifiable ground truth. The simulator enables telemetry generation that preserves explicit cause–effect relations among signals and operating modes, so that perturbations can be interpreted as changes in system behavior rather than as arbitrary feature edits. This is a prerequisite for boundary-based robustness studies, where stressing classifiers must not push samples into physically meaningless regions.

We designed the framework to support two complementary evaluation modes: Online Evaluation and Adversarial Benchmarking. *Online evaluation* measures how the classifier performs during operation by using normal, anomalous, and attack-related data to test detection in realistic data streams. In contrast, *Adversarial Benchmarking* is done offline at regular intervals. It focuses on boundary-based changes that gradually make classes harder to separate. This approach allows for a controlled test of robustness in hard-label and black-box scenarios. Both evaluation modes can be executed in single-node or multi-vehicle configurations, leveraging federated learning, which allows the study of robustness effects across heterogeneous nodes rather than as an isolated property of a single classifier.

Although SereBench is designed as a modular railway-oriented benchmarking framework, the experimental campaign presented in this paper focuses on the pneumatic braking subsystem as a physically critical and tightly constrained component, where perturbations can be injected and interpreted unambiguously. Therefore, the reported robustness results concern braking-pressure dynamics and should not be assumed to generalize to traction, signaling, onboard diagnostics, or other railway subsystems without additional subsystem-specific validation. The experimental variability that defines *anomaly*, *attack*, and *adversarial perturbations* is injected in the hydraulics/braking module, where pressure dynamics provide a physically interpretable basis for boundary-based robustness evaluation. In this setup, normal operations, anomalies, and attacks are generated in a structured manner, yielding telemetry with clear, interpretable ground truth. Concretely, the benchmarked events are expressed as perturbations of brake-pressure signals, so class semantics remain anchored to mechanically meaningful dynamics. The resulting data can then be stressed through controlled Gaussian boundary perturbations and evaluated through HSJA-based hard-label black-box adversarial testing, allowing classifiers to be challenged near their decision boundaries.

Building on this framework, our experimental study explores various configurations that combine federated learning, noise-augmented training, and robustness evaluation under both controlled Gaussian boundary stress and HSJA-based adversarial evaluation. This approach allows us to decouple and isolate the specific factors influencing model robustness. Specifically, we examine the individual and collective effects of noise-augmented training, cross-vehicle collaboration, classifier architecture, federated aggregation strategy, and heterogeneous local stress conditions, providing a granular view of how robustness is maintained in complex, distributed environments.

The main contributions of this work can be summarized as follows:

- We introduce SereBench, an open-source security and reliability benchmarking framework for evaluating adversarial robustness and federated resilience of classification models in railway CPS, and instantiate it in this work through a pneumatic braking subsystem case study.
- We design a generative railway simulator that produces physically coherent telemetry with explicit ground truth for normal operation, anomalies, and cyber-attacks.
- We define a boundary-based robustness evaluation methodology tailored to hard-label, black-box settings and streaming constraints typical of IIoT deployments.
- We discuss a pneumatic braking subsystem case study encompassing multiple classifier architectures and federated aggregation strategies, showing how SereBench supports robustness evaluation under controlled braking-pressure perturbations, FL configurations, heterogeneous local stress conditions, and controlled communication impairments affecting the FL model-exchange flow.

The rest of this paper is organized as follows. Section 2 provides background and explains the limits of real railway telemetry for robustness testing. Section 3 reviews current railway CPS reliability and security platforms, adversarial robustness benchmarks, and federated learning methods. Section 4 introduces the proposed benchmarking framework. Section 5 presents our experimental setup and the obtained results. Finally, Section 6 discusses the limitations of the proposed approach, while Section 7 concludes the paper and outlines future research directions.

2. Background

Robustness evaluation in railway Cyber–Physical Systems cannot be treated as a purely algorithmic problem. It is tightly constrained by the

nature of operational telemetry, the semantics of observable signals, and the coupling between physical processes and control logic. This section provides the background needed to motivate the design choices underlying SereBench, focusing on the characteristics of railway telemetry, the limitations of real-world datasets for robustness benchmarking, and the requirements that a reproducible and physically meaningful evaluation framework must satisfy.

2.1. Railway CPS telemetry

Telemetry from Railway CPS is a by-product of operating a safety–critical asset. A single telemetry stream is typically consumed by multiple functions at once, including control loops, onboard diagnostics, event logging, and maintenance support. For this reason, railway telemetry is inherently heterogeneous: continuous physical measurements, such as pressures, currents, voltages, and speeds, coexist with discrete indicators, mode flags, and diagnostic codes generated by different subsystems and sampled at different rates.

Although continuous variables are associated with physical units, their interpretation is strongly context-dependent since a pressure level that is perfectly acceptable during cruising may indicate a fault during braking; at the same time, a traction current considered normal under load may be suspicious during coasting. Discrete variables present an even more problematic scenario. In many cases, these systems encode internal thresholds, diagnostic routines, or vendor-specific state machines rather than externally observable physical conditions. As repeatedly highlighted in the Industrial Control Systems (ICS) literature, operational datasets often conflate physical dynamics with logging and control logic, making both label definition and performance interpretation inherently ambiguous [12,13].

Another defining feature of railway telemetry is the strong coupling among subsystems. Physical processes and control logic evolve together; for example, traction phases influence braking behavior, braking commands affect pneumatic responses and speed profiles, and safety logic, such as ERTMS-related diagnostics, reacts to operating modes and fault conditions. If these dependencies are ignored during evaluation, an ML-based detector may appear stable while actually exploiting correlations incidental to the data-collection process. If such coupling is not preserved during robustness evaluation, classifiers may learn spurious invariants that remain stable under perturbation, yielding artificially robust decision boundaries. This is a known issue in streaming environments, where regime changes and recurrent variations in data generation can destabilize learned decision rules unless explicitly accounted for during evaluation [14].

2.2. Limitations of real-world railway datasets for robustness benchmarking

Public datasets in the railway domain are scarce, and when available, they often lack the metadata required to interpret codes, missing values, and data acquisition logic [10]. As a result, the operational meaning of many recorded signals remains only partially observable, and class semantics are rarely verifiable in engineering terms. In addition to the large amount of data, robustness benchmarking requires clear, defensible class semantics, temporal coherence that works for streaming analysis, and perturbation spaces that remain understandable as classifiers are pushed to their limits near their decision boundaries.

When the semantics of discrete indicators, diagnostic codes, or operating modes cannot be reconstructed with confidence, it may not be feasible to determine whether learned decision boundaries reflect physical behavior or merely internal logging conventions. To overcome this obstacle and augment the semantic relevance of robustness evaluations, our study focuses on simulating a networked railway CPS that includes telemetry from multiple functional areas, including traction and electrical units, pneumatic components, ERTMS diagnostics, and positional subsystems. However, in the experimental campaign presented in this paper, class differentiation and boundary-oriented perturbations are defined on the pneumatic braking subsystem, where pressure dynamics provide explicit and physically interpretable class semantics. The simulator design is informed by empirical properties observed in real datasets, while enabling explicit control over state definitions and transitions.

Temporal inconsistency and streaming ambiguity. Boundary-based robustness analysis involves temporally significant sequences where regimes, transitions, and anomaly precursors must be identifiable. Real-world railway traces often contain irregular sampling rates, repeated data, and data gaps. Such issues are well documented in ICS data collection pipelines [13,15]. Under these conditions, it becomes difficult to distinguish genuine changes in plant dynamics from artifacts introduced by instrumentation or recording. These issues do not diminish the value of real-world datasets for descriptive analysis or exploratory modeling. However, they explain why trace-driven paradigms may often be unsuitable for systematic robustness benchmarking, where interpretability, controllability, and reproducibility are essential. These requirements motivated the adoption of a simulation-based framework, which enables explicit state modeling, physically consistent signal generation, and controlled stress testing of classifier decision boundaries.

2.3. Robustness benchmarking requirements in railway CPS

Robustness benchmarking of classifiers in the context of railway CPS must satisfy the following three core requirements.

Explicit and verifiable state definitions. Robustness can only be meaningfully assessed when classes correspond to operational states that can be interpreted in engineering terms. Therefore, the benchmark must make explicit what is meant by *normal*, *anomalous*, and *attack-related* behavior, and how these states manifest in observable signals.

Physically consistent inter-subsystem correlations. Perturbations that arbitrarily modify individual features can easily produce samples that would never arise in real operation conditions, particularly when subsystems are tightly coupled. Evaluating classifiers on such samples reveals properties of the model's geometry rather than its operational reliability, e.g., braking cylinder pressures inconsistent with brake pipe dynamics or traction currents incompatible with operating mode.

Controlled mechanisms for stressing decision boundaries. The boundary-based evaluation requires gradual decreases in class separability while preserving the physical plausibility of observations. Earlier adversarial robustness work has clearly demonstrated that unprincipled evaluation protocols may strongly overestimate a model's robustness [9]. In the railway context, this translates into the need for repeatable stress mechanisms that expose where and how classifiers become uncertain.

To meet the requirements discussed above, this work adopts a simulation-based benchmarking approach. The proposed framework is built around a railway-oriented simulator that provides explicit and verifiable ground truth, preserves physically meaningful correlations among subsystems, and allows classifier decision boundaries to be stressed in a controlled manner. The simulator is specifically designed to support reproducible robustness evaluations under selected railway-oriented operating assumptions, including streaming data, hard-label feedback, and black-box access to the learning models.

3. Related work

This section places SereBench in the context of three main research areas relevant to our work: (i) open-source platforms and testbeds for CPS/IIoT security and reliability, especially in the railway sector; (ii) methods for assessing adversarial robustness, with a focus on decision-boundary analysis, hard-label black-box evaluation and recent time-series and industrial CPS adversarial robustness studies; and (iii) federated learning as a way to study distributed monitoring in networked CPS.

3.1. Security and reliability testbeds and benchmarking platforms for CPS/IIoT

Early research on CPS and IIoT security and reliability mostly used large single-purpose simulators to test control logic or measure detection accuracy in defined fault or attack scenarios. As machine learning-based detection became more common, these methods showed clear limits: results were hard to reproduce, experiments depended on specific setups, and comparing different methods was often not practical [16].

To address these issues, the community has moved toward modular, reproducible testbeds made for benchmarking. These platforms focus on controlled experiments, repeatability, and keeping tests separate from real systems, which are often off-limits for security testing because of safety and cost. ICSSIM [17] is an example, offering a flexible framework for industrial control system security experiments and allowing systematic study of attacks and system behavior in controlled settings. Several studies in both the CPS and IIoT domains have highlighted the need for reproducible and isolated testbeds for security and reliability experimentation [18], due to the impracticality of performing adversarial testing on operational systems [15]. Existing surveys emphasize that evaluation results in industrial monitoring strongly depend on the experimental setup, data semantics, and deployment assumptions [19], making fair comparison across approaches difficult without shared benchmarking infrastructures [20]. Similar concerns have also been raised in the networking community, where recent systematic investigations emphasize how experimental design, dataset semantics, and deployment assumptions strongly influence intrusion detection results, calling for more robust and standardized evaluation practices [21]. Further, recent work on adversarial robustness has shown that evaluation protocols focusing exclusively on clearly separated classes and nominal operating conditions can lead to a systematic overestimation of model reliability [22]. In such settings, global performance metrics may hide fragile decision regions that emerge only when inputs approach class boundaries or reflect ambiguous operational states [23].

In the railway field, Open-FARI [3] is an important step toward open and repeatable experiments. It offers a container-based framework for federated learning using railway-like data, making it possible to study distributed training across several vehicles. Another solution is provided by SereWay [24], which builds on Open-FARI by adding cyberattacks as a specific observation class, moving from simple anomaly detection to multi-class intrusion classification. Still, both platforms mostly test classifiers in clearly separated conditions. They do not examine how classifiers behave near decision boundaries or provide ways to test models with purposely unclear inputs. As a result, these platforms primarily evaluate detection accuracy under nominal conditions, leaving the most fragile regions of the classifier, where operational ambiguity and uncertainty arise, largely unexplored.

It is important to note that SereBench is meant to complement these platforms. Rather than aiming to cover all possible adversarial attack families, the current implementation focuses on how a classifier behaves when the line between normal, anomalous, and attack data becomes less clear under controlled and physically interpretable perturbation conditions.

3.2. Adversarial robustness and decision-boundary analysis

Adversarial robustness has been widely studied in supervised learning, first under white-box settings where attackers know all model details. While this is useful for analysis, it may often not match real-world CPS or IIoT monitoring systems. In practice, attackers interacting with deployed detectors typically observe only the final decision label. This constraint motivates decision-based, or black-box, adversarial attacks, in which the attacker probes the classifier through queries and infers the structure of its decision boundary indirectly. Boundary Attack [25] was among the first methods to formalize this approach, showing that decision boundaries can be approximated without gradient access. Subsequent work, including HopSkipJumpAttack (HSJA) [8], significantly improved query efficiency, making boundary exploration feasible even under restrictive interaction budgets. Other studies have explored query-efficient sampling and local boundary estimation strategies in similar black-box settings [26].

A central lesson from this literature is that robustness claims depend critically on the evaluation protocol and on whether fragile decision-boundary regions are explicitly probed. Overall accuracy can hide serious weaknesses if models are tested on clearly separated data only. This has led to tools like AutoAttack [9], which uses several attack methods to avoid false claims of robustness. Other work has stressed adaptive testing, showing that defenses tested only in fixed ways may fail against attacks designed for their weaknesses [27].

A related line of work has studied adversarial robustness for time-series models and industrial CPS from perspectives that are complementary to SereBench. For time-series classification, Belkhouja and Doppa proposed TSA-STAT, which generates adversarial examples by constraining statistical properties of the signal, rather than relying only on standard norm-bounded perturbations [28]. In a complementary direction, Belkhouja et al. used dynamic time warping to define adversarial perturbations that better reflect temporal misalignment and similarity in sequential data [29]. These works are relevant to the positioning of SereBench because they show that adversarial testing for time-series data should account for temporal and statistical structure, beyond generic point-wise perturbation models. Industrial CPS and control-oriented studies have addressed similar concerns from a system-security perspective. Chen et al. showed that adversarial examples can evade machine-learning-based intrusion detectors in industrial control systems, including through optimization-based and generative strategies [30]. Pozdnyakov et al. provided a systematic benchmark of adversarial attacks and defenses on the Tennessee Eastman Process, while Zhuo et al. studied attack and defense strategies for data-driven fault detection and classification systems [31,32]. Other works move closer to process-aware robustness: Maiti et al. considered adversarial attacks on data-driven invariant checkers for CPS, and Tian et al. introduced a multi-label false-data injection attack for smart-grid locational detection [33,34].

Taken together, these studies show that adversarial robustness evaluation for time-series and industrial CPS involves several attack models and assumptions, including statistical constraints, temporal alignment, optimization-based attacks, generative attacks, process-control benchmarks, invariant-based checks, and false-data injection settings. At the same time, these works are mostly evaluated on generic time-series datasets, industrial process-control benchmarks, or power-system scenarios, and they do not target railway-specific telemetry with explicit braking-oriented physical semantics. This distinction clarifies the scope of SereBench. The framework is not intended to provide exhaustive coverage of all time-series adversarial threat models. Rather, in this paper, SereBench adopts a deliberately scoped and physically interpretable design: Gaussian boundary stress is used to reduce class separability over mechanically meaningful braking-pressure channels, while HSJA is used to probe decision-boundary fragility under a hard-label black-box threat model. Extending SereBench with temporal, adaptive, universal, surrogate-based, and process-aware attacks is therefore a natural direction for future work.

3.3. Federated learning in CPS security and reliability

Federated Learning (FL) has emerged as a practical solution for training shared models in environments where raw data cannot be centralized due to privacy, bandwidth, or resilience constraints [35]. These constraints are particularly relevant in networked CPS, where data is generated by physically distributed assets, such as trains or substations, and may be subject to proprietary or regulatory restrictions.

In security and reliability monitoring, FL brings both benefits and challenges. Training on different nodes can help models generalize by exposing them to varied conditions. However, differences in data, training, and noise can slow learning or weaken robustness. Studies [36–39] have shown that FL systems can be vulnerable to noisy or adversarial clients, and that it is hard to transfer robustness learned on one node to the whole system.

SereBench uses FL as an experimental mechanism to study how robustness-related behavior changes across distributed vehicle nodes operating under heterogeneous local perturbation regimes. In particular, the framework allows evaluating whether models trained under more challenging local conditions can influence the behavior of other nodes through model aggregation. Most existing FL security studies [39–41] focus on average performance, privacy, robustness, or convergence under homogeneous or mildly heterogeneous conditions, without explicitly analyzing how boundary-related robustness effects emerge across nodes in physically interpretable railway CPS settings.

Recent federated learning research has also investigated contribution and trust evaluation mechanisms. For example, lightweight contribution evaluation approaches aim to quantify the usefulness of client updates during federated training, while trust-oriented schemes assess the reliability of participants in distributed or digital-twin-enabled environments [42,43]. These works are complementary to SereBench: they focus on evaluating client contribution or trustworthiness during federated learning, whereas SereBench focuses on benchmarking robustness-related behavior under physically meaningful railway CPS perturbations and multi-vehicle aggregation.

3.4. Our contribution

To clarify the overall positioning of SereBench, Table 1 compares the proposed framework with the main research directions discussed in the Related Work section. The comparison is not limited to federated learning, but also considers CPS/IIoT benchmarking platforms, adversarial robustness and decision-boundary analysis approaches, and prior railway-oriented frameworks such as Open-FARI and SereWay.

The comparison follows the main dimensions addressed by this work, namely open-source benchmarking for railway CPS, physically coherent telemetry generation with explicit ground truth, security and reliability event classification, boundary-oriented robustness evaluation under hard-label black-box and streaming assumptions, and federated multi-vehicle evaluation through the case study considered in this paper.

The comparison highlights that existing CPS/IIoT testbeds and evaluation studies provide useful support for controlled experimentation, but they are not specifically designed around railway CPS semantics and do not jointly address physically coherent telemetry, explicit ground truth, and boundary-based robustness evaluation. Decision-boundary and adversarial robustness methods focus on fragile classifier regions and hard-label black-box settings, but they are generally domain-agnostic and do not enforce physically meaningful railway telemetry constraints. Time-series and industrial CPS adversarial robustness studies broaden the evaluation landscape by considering statistical constraints, temporal alignment, process-control benchmarks, invariant-based checks, and false-data injection settings. However, these works are mostly evaluated on generic time-series datasets, industrial process-control benchmarks, or power-system scenarios, and do not target railway-specific telemetry with explicit braking-oriented physical semantics. Federated learning studies address distributed training and robustness-related issues across multiple nodes, but they do not combine railway-specific class semantics with physically constrained boundary-oriented evaluation. Finally, Open-FARI and SereWay provide railway-oriented and reproducible experimentation layers, but they mainly evaluate classifiers under clearly separated operational classes. SereBench complements these directions by combining open-source railway CPS benchmarking, physically coherent telemetry generation, explicit ground truth for security and reliability events, boundary-based robustness evaluation, online monitoring, and federated multi-vehicle experimentation within a single framework. In this paper, these capabilities are demonstrated through a pneumatic braking-subsystem case study and should not be interpreted as exhaustive coverage of all railway subsystems or all time-series adversarial threat models.

Existing security and reliability benchmarks for CPS and railway domains do not jointly address physically coherent railway telemetry generation, explicit ground truth for security and reliability events, boundary-based robustness evaluation, and federated multi-vehicle experimentation within a single reproducible framework. To fill this gap, SereBench combines three elements: (i) a railway-inspired simulator with explicit and interpretable class semantics; (ii) a domain-constrained boundary-oriented perturbation mechanism designed to stress classifier fragility while preserving the physical plausibility of telemetry observations; and (iii) federated, multi-vehicle experimentation to observe how robustness-related behavior changes across nodes under heterogeneous local perturbation regimes. In this way, SereBench acts as a benchmarking layer focused on the most failure-prone regions of the classifier, rather than as a platform limited to clearly separated nominal, anomalous, and attack conditions.

4. Proposed benchmarking approach: SereBench

We introduce SereBench, an open-source framework[1] for evaluating adversarial robustness and federated resilience of AI-based anomaly and attack classification in networked railway CPS [44]. To provide explicit, verifiable ground truth and to maximize physically meaningful correlations among subsystems, SereBench uses a railway CPS simulator that generates related telemetry data. This makes it possible to overcome the limitations of current real-world railway datasets and to meet the robustness benchmarking requirements discussed in Section 2.

The simulated system is organized into functional areas that reflect the conceptual structure of a railway CPS, including electrical and traction behavior, pneumatic braking dynamics, and ERTMS-related diagnostics. The simulator is able to generate telemetry observations from each area, emulating both normal and abnormal, either malicious or non-malicious, system behavior. Observations are analyzed by a monitoring subsystem that uses ML models to classify security- and reliability-related events.

This controlled environment enables the **adversarial robustness benchmarking** of ML anomaly detection models, where their decision boundaries are stressed in a controlled manner. This is done through boundary-based changes that gradually make classes harder to separate. This approach allows for a controlled test of robustness in hard-label and black-box scenarios.[2] By progressively reducing class separability via targeted modifications, the benchmark explores classifier behavior near decision boundaries without entering physically meaningless regions. The emphasis on boundary behavior under realistic hard-label assumptions, black-box with respect to the classifier internals but domain-informed with respect to the targeted sensing channels, is motivated by decision-based adversarial methodologies [8,25]. As a result, robustness scores obtained in this setting can be interpreted as properties of the classifier under realistic operational constraints and controlled degradations.

[1] SereBench has been made publicly available at https://github.com/DIETI-DISTA-IoT/SereBench.

[2] In this work, boundary-based robustness is evaluated through two complementary mechanisms: controlled, domain-constrained Gaussian perturbations applied to selected braking channels, and decision-based hard-label black-box adversarial evaluation based on HopSkipJumpAttack. The former progressively reduces class separability under physically interpretable stress conditions, while the latter queries the classifier to identify adversarial examples near the decision boundary.

Table 1
Comparison between SereBench and representative related works discussed in the Related Work section.

Related work	Open	Rail.	Phys. GT	Sec./ Rel.	Bound.	Stream.	FL/ MN
ICSSIM [17]	✓	×	~	~	×	×	×
CPS/IIoT testbeds [15,18–21]	~	×	~	~	×	~	×
Decision-boundary and adversarial robustness methods [8,9,25–27]	~	×	×	×	✓	×	×
Robustness-centric evaluation studies [22,23]	~	×	~	~	✓	~	×
Time-series and industrial CPS adversarial robustness studies [28–34]	~	×	~	~	✓	~	×
FL for distributed monitoring [35–43]	~	×	×	~	~	~	✓
Open-fari [3]	✓	✓	~	~	×	~	✓
SereWay [24]	✓	✓	~	✓	×	~	~
SereBench	✓	✓	✓	✓	✓	✓	✓

Legend: ✓ = addressed; ~ = partially addressed; × = not explicitly addressed.
Open = open-source or reproducible benchmark support.
Rail. = railway CPS specificity.
Phys. GT = physically coherent telemetry with explicit ground truth.
Sec./Rel. = support for security and reliability event classification.
Bound. = boundary-based robustness evaluation.
Stream. = streaming or online monitoring/evaluation.
FL/MN = federated learning or multi-node evaluation.
For SereBench, the reported dimensions refer to the framework capabilities and to the pneumatic braking subsystem case study evaluated in this work. They should not be interpreted as evidence of exhaustive cross-subsystem validation, complete coverage of all railway subsystems, or complete coverage of all time-series adversarial threat models.

Adversarial robustness benchmarking can be executed in either single-node or multi-vehicle configurations. In a multi-vehicle configuration, SereBench allows enabling federated learning to study federated resilience effects across distributed vehicle nodes. This makes it possible to observe how robustness-related behavior changes under model aggregation when vehicles operate under heterogeneous local stress conditions.

We now describe the railway scenario, the adversarial perturbation and robustness evaluation strategy, and the evaluation workflow.

4.1. Railway scenario

We consider a networked railway CPS for onboard monitoring, where several vehicles act as independent nodes.

In the context of SereBench, the term *networked* refers to a distributed monitoring architecture composed of multiple vehicle nodes that operate locally while participating in a shared evaluation workflow. Each vehicle generates its own telemetry stream, maintains and updates a local model, and performs local inference on the observations produced by its onboard simulator. When federated learning is enabled, vehicles do not exchange raw telemetry. Instead, they periodically share model parameters with a federated aggregation component, which produces an updated global model that is redistributed to the participating nodes. Therefore, the networked interaction modeled in SereBench captures the logical coordination among distributed railway assets for collaborative robustness evaluation, rather than the packet-level behavior of an operational railway communication network.

Each node runs two main components: a *railway simulator* and an *online monitoring* system.

The **railway simulator** generates time-stamped telemetry from five railway areas, emulating the behavior of a railway CPS system:

- *Operating context and dynamics* - describes the overall operating conditions of the vehicle, including the current driving mode (traction, coasting, braking, stopping), the vehicle status, and contextual variables that regulate the behavior of other subsystems.
- *Control logic and cab* - produces driver commands and control signals that regulate braking and traction requests, and coordinate the interaction between on-board components.
- *Traction and electrical units* - generates traction-related states and electrical operating conditions that affect vehicle movement and energy consumption.
- *Pneumatic braking dynamics* - models the pressures in the main reservoir, brake pipe, and brake cylinders on all cars, including event-dependent disturbances and sensor noise.
- *ERTMS/HMI diagnostics* - emulates signaling information, supervision data, and human-machine interface indicators used for on-board monitoring.

Although SereBench generates telemetry for the five functional areas described above, the present experimental campaign uses the pneumatic braking subsystem as the target subsystem for class differentiation and boundary-oriented perturbation. This design choice is methodological rather than incidental: the pneumatic braking subsystem provides safety–critical and mechanically interpretable pressure dynamics, enabling controlled perturbation while avoiding arbitrary feature manipulation. This choice allows changes in the generated signals to be related to operating modes, anomalous behaviors, and attack-related conditions. The results should therefore be read as a braking-oriented case study within the broader SereBench framework, rather than as evidence that the same robustness behavior necessarily extends to traction, signaling, onboard diagnostics, or other railway subsystems. Cross-subsystem robustness validation remains outside the scope of the present experimental campaign and requires subsystem-specific perturbation models and validation procedures.

The generated telemetry observations represent the state of the simulated railway CPS at a specific time, and are stored as fixed-size feature vectors. Observations cover all the areas mentioned above, each governed by deterministic rules with added stochastic variation to emulate realistic changes while preserving cause-and-effect links. Specifically, the simulator generates signals according to an explicit operational profile defined at the scenario level. This profile specifies the sequence and duration of operating phases for each vehicle, such as traction, cruising, coasting, service braking, and emergency braking. Based on this profile, the simulator activates control actions, pressure variations, and mode transitions at predefined times, and drives the temporal evolution of subsystem variables. As a result, telemetry reflects coherent operating regimes and physically consistent dynamics, rather than independent or artificially correlated signal fluctuations.

Notably, telemetry generation is not based on independent random sampling, but on structured, state-dependent models that encode physically grounded relationships among operating mode, event semantics, and braking pressures. This design preserves causal links between control state (e.g., braking vs. coasting), physical responses

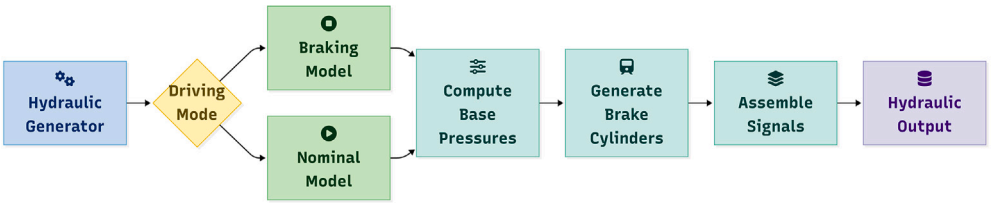


Fig. 1. Layered data generation process of the braking subsystem.

(e.g., cylinder pressures), and observed signals, ensuring that class differences and perturbations arise from interpretable system dynamics rather than statistical artifacts. As an example, Fig. 1 illustrates the braking hydraulic data generation process. Pressure signals are not sampled independently; rather, they are derived from physically motivated operating states and control conditions. At each simulation step, the *hydraulic generator* simulates the primary pressure source, then baseline pressure values are computed according to the current driving mode (i.e., nominal vs. braking). During nominal operation, the system maintains pressures close to their reference levels, while braking phases induce systematic reductions in reservoir and pipe pressures and corresponding increases in brake cylinder activations. This state-driven behavior reflects the underlying pneumatic dynamics of the braking system. All generated signals are finally normalized and assembled into structured telemetry vectors that include pressures from all brake cylinders together with the main braking channels (*Hydraulic Output*). This representation preserves both spatial and functional relationships among braking components.

The simulator generates three classes of observations: (i) *Normal*, telemetry consistent with nominal subsystem behavior and expected cross-subsystem dependencies; (ii) *Anomaly*, non-malicious degradations that perturb specific subsystems while preserving structural constraints; (iii) *Attack*, deliberate manipulation of telemetry observations preserving their interpretability from an engineering standpoint. These classes are generated by manipulating data from the braking subsystem, which we consider as a controlled, physically grounded part of the CPS. Therefore, boundary-oriented stress perturbations, discussed later, can be injected and interpreted unambiguously within this braking-oriented setting.

The braking subsystem is modeled as a pneumatic system composed of: (i) a main air reservoir, (ii) a brake pipe, (iii) two brake chains spanning all carriages, (iv) a main brake cylinder per carriage per chain, and (v) a trailer brake cylinder per carriage per chain. With four carriages, this results in 18 pressure sensors in total. In nominal conditions, braking commands result in decreases in main reservoir and brake pipe pressures and a simultaneous increase in brake cylinder pressures. When braking is not active, the same signals reverse direction. These dynamics define the normal operation, which is used by the simulator to generate the three classes of observation, as shown in the left portion of Fig. 2:

- **Normal**: follow the standard pneumatic dynamics described above, including changes caused by control actions and normal operational variation.
- **Anomaly**: generated by simulating emergency braking. Pressure fluctuations are amplified in the main reservoir and brake pipe, while all other braking components follow the nominal braking dynamics.
- **Attack**: models cyber-induced effects on the braking subsystem or its sensors. We assume an attacker capable of spoofing brake-related pressure readings or altering the reported state of the braking subsystem. As with anomalies, attack-induced deviations are injected exclusively into the main reservoir and brake pipe signals.

The **online monitoring** makes use of a *local model* to continuously perform the classification of the telemetry vectors received as input. The telemetry vectors are also used to *periodically train* the model. This emulates the usual behavior of onboard monitoring systems with continuous detection and model updates performed as maintenance routine. Therefore, at scheduled intervals, the node trains (or retrains) the classifier on the telemetry accumulated during operation. When federated learning is active, the local model of each vehicle is also combined into a shared global model; the resulting model is then used to update the parameters of each local model. This model-exchange mechanism represents the networked coordination layer implemented in SereBench. In addition to ideal communication conditions, the FL model-exchange flow can be subjected to configurable message drops and delivery delay with jitter, in both the uplink and downlink directions, as detailed in Section 4.3. These mechanisms provide a controlled message-level abstraction of missing or stale model updates. They do not reproduce the complete packet-level behavior of an operational railway communication network. Bandwidth limitations, network topology, protocol-specific scheduling and recovery mechanisms, and fully asynchronous FL remain outside the scope of the current implementation and are discussed in Section 6.

4.2. Adversarial perturbation and robustness evaluation strategy

Boundary-oriented robustness evaluation is used to reduce class separability and stress the classifier near its decision boundaries. These perturbations are not modeled as cyberattacks, but rather as abstract, domain-constrained manipulations of the classifier's inputs. SereBench supports two complementary perturbation and evaluation mechanisms. The first one is based on **domain-constrained Gaussian noise** injected into selected braking-pressure channels, and is used to reduce class separability in a controlled and physically interpretable way. The second one is based on **HopSkipJumpAttack** (**HSJA**) [8], and is used as a decision-based hard-label black-box adversarial evaluation of the current classifier. The former provides a configurable stress mechanism over the simulated railway telemetry, while the latter evaluates how close adversarial examples can be found to the original observations through classifier queries only.

In SereBench, the Gaussian boundary stress perturbations are restricted to brake-related pressure signals to preserve mechanical plausibility. In this context, we generate perturbation events through a narrow injection surface that affects only the two global braking channels, i.e., the reservoir and the brake pipe. Restricting injection to these channels prevents artifacts from spreading across multiple elements and maintains a controlled setting in which small perturbations on global channels can induce measurable changes in system behavior. Extending anomaly injection/attacks to other simulator blocks is compatible with the modular design, but it is not necessary to study boundary fragility within the current benchmark.

The Gaussian boundary stress component is implemented by injecting zero-mean Gaussian noise into the main reservoir and brake pipe pressure readings. The amount of noise can be set differently for each vehicle, thereby varying the degree of boundary stress induced by the Gaussian perturbation mechanism. These changes can be applied to both anomaly and attack observations, making them harder to differentiate from other classes. This method makes it possible to

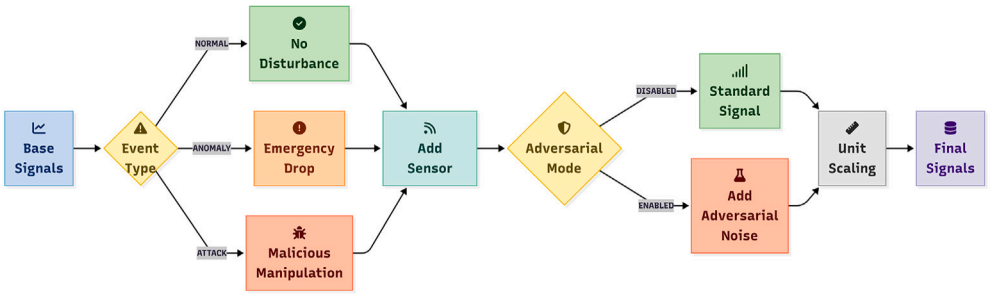


Fig. 2. Per-vehicle Gaussian boundary stress workflow applied to the braking subsystem in SereBench.

assess boundary fragility while keeping the data close to real operating conditions.

This perturbation mechanism is not intended to model the full spectrum of adaptive, temporal, or process-aware adversarial attacks. Rather, it is used as a controlled, domain-constrained boundary-stress mechanism to progressively reduce class separability over mechanically meaningful braking-pressure channels while preserving the physical interpretability of the generated telemetry.

From an architectural perspective, the Gaussian perturbation mechanism is integrated within the hydraulic data generation process shown in Fig. 1, which is further detailed in Fig. 2.

This figure summarizes how the three observation classes are generated and how Gaussian boundary stress perturbations are applied within each vehicle node. The railway simulator produces an unperturbed telemetry stream (*Base Signals*) used for normal operation (i.e., *normal* class). On top of this baseline, semantic event effects are applied in the form of structured pressure shifts. Non-malicious degradations (i.e., *anomaly* class) introduce moderate deviations consistent with emergency braking scenarios, whereas malicious events (i.e., *attack* class) induce larger, coordinated variations that emulate intentional manipulation of brake-related signals. These transformations define class-specific operational regimes that remain mechanically interpretable. Subsequently, stochastic components are introduced in two stages. A low-amplitude noise term is always added to model intrinsic sensor uncertainty and measurement variability. Instead, when the Gaussian stress mode is enabled, an additional noise component is injected on the global braking channels, with an intensity that can be configured independently for each vehicle. As a result, vehicles can be exposed to different stress levels, and class separability can be progressively reduced in a reproducible manner. All generated signals are finally normalized and assembled into structured telemetry vectors.

By superimposing Gaussian stress perturbations on top of state-driven and event-driven dynamics, the simulator ensures that boundary-adjacent samples originate from realistic operating conditions rather than arbitrary signal distortions. As a result, robustness evaluation reflects classifier behavior under physically plausible degradations and controlled ambiguity, instead of under synthetic noise patterns detached from system semantics.

It is important to note that SereBench also allows using the Gaussian perturbation mechanism during training. In this configuration, the model is trained on noisier and more ambiguous observations generated by the simulator. Therefore, this setting is better interpreted as noise-augmented training rather than as optimized adversarial training, since the perturbations are not generated by directly attacking the current model.

The HSJA-based component is executed during robustness evaluation rather than during telemetry generation. Unlike the Gaussian stress mechanism, HSJA does not inject a predefined noise distribution into selected channels. Instead, it interacts with the current classifier as a hard-label black-box adversary: it observes only the predicted class, queries the model, and searches for adversarial examples that cross the decision boundary while remaining close to the original observations. This makes HSJA complementary to the Gaussian stress mechanism. The Gaussian branch provides a controlled and domain-constrained way to reduce class separability, whereas HSJA evaluates how close adversarial examples can be found to the original observations through a decision-based black-box attack.

In this work, HSJA is used to evaluate decision-boundary fragility under a hard-label black-box threat model, where the attacker can query the classifier and observe only the predicted label, without access to model parameters, gradients, or confidence scores. This choice complements the controlled Gaussian boundary stress, but it does not exhaustively cover temporal, adaptive, universal, surrogate-based, or process-aware adversarial attacks, which are left as future works.

4.3. Evaluation workflow

Fig. 3 summarizes the end-to-end workflow implemented in SereBench, which is controlled by a **Simulation Manager** that instantiates and configures the vehicles composing the scenario. The manager controls all relevant behaviors through configuration parameters, and through graphical and command-line-based interfaces to dynamically change simulator settings, class generation frequency, per-vehicle perturbation levels on the braking channels, training and evaluation schedules, and, when enabled, the timing and method of model aggregation.

The scenario to simulate is described by means of a *Scenario definition* file provided to the Manager, which describes the details of the simulation, such as number of vehicles, subsystems to emulate, operation profile of each vehicle, simulation duration, how to generate the attack and anomaly classes, class generation frequency, the perturbation rules (i.e., the description of how to generate perturbations on the telemetry), training and evaluation schedules, and, when enabled, the timing and method of FL model aggregation. In each vehicle, the workflow is characterized by five main stages, i.e., *railway simulation*, *online monitoring*, *model update*, *robustness benchmarking*, and the (optional) *federated learning*.

Railway Simulation makes use of the *simulator* described in Section 4.1 to perform the generation of the telemetry observations from the considered five railway areas and the three observation classes, i.e., *normal*, *anomaly*, and *attack*. This stage considers the configuration provided by the manager and generates a *Telemetry stream* as output, which encompasses the generated observations, provided as a sequence of fixed-size feature vectors.

Model Update is a periodic stage where the *Local Model*, i.e., the model under test, is periodically updated simulating a maintenance routine. To this aim the observations from the simulator are accumulated in a *Telemetry Buffer* and used for a *Periodic Training* of the model. Note that the buffer does not represent a pre-collected dataset, but models the delayed update behavior typical of onboard monitoring systems. It is integral to the design and serves to maintain a clear distinction between what the model sees when making predictions and what is subsequently used for training updates. At the end of each model update, the *current model version* is provided as output. Notably,

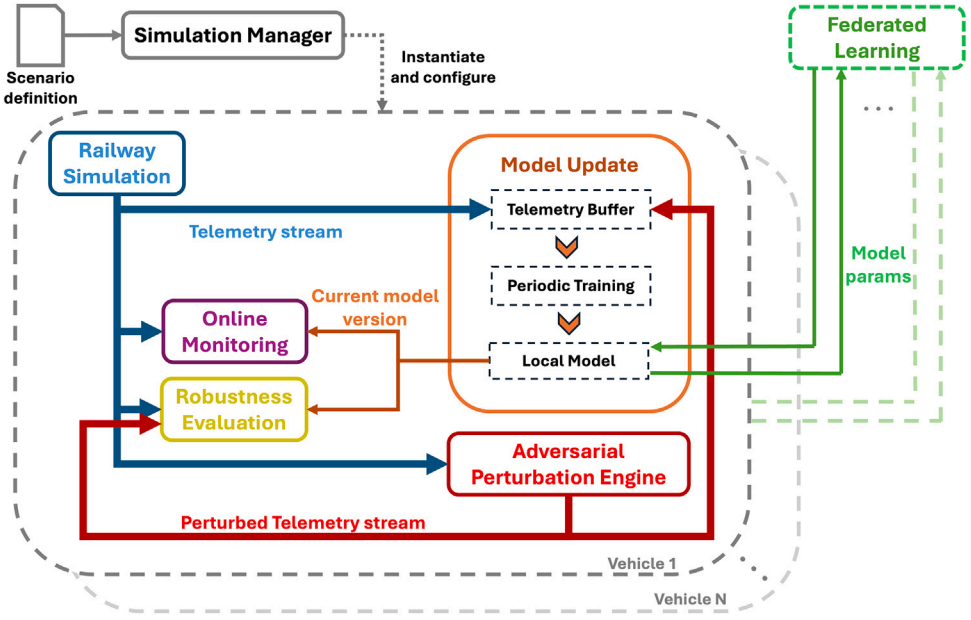


Fig. 3. Execution flow implemented at each SereBench vehicle node.

the model update can also be performed under a noise-augmented training configuration. In this case, the local model is trained by using the Gaussian-stressed version of the telemetry stream generated by the simulator. This exposes the classifier to more ambiguous observations during training, but does not rely on optimized adversarial examples generated against the current model. Further, the model update also takes place when federated learning is activated. In this case, the model is updated with the parameters provided by the FL aggregator (described later). It is important to note that SereBench is not tailored to a specific AI model.

Both the current model version and the telemetry stream are provided to the **Online Monitoring** stage, which continuously executes an inference pipeline with the aim to classify the vectors composing the stream leveraging the trained model. During execution, inference metrics, i.e., accuracy, precision, recall, and F1-score, are collected in order to evaluate the detection performance of the model under test. In this way time-based effects, such as warming, drift, and convergence, can be observed rather than averaged.

Robustness Evaluation is a periodic stage used to assess the robustness of the current model version during system operation. Similar to online monitoring, this stage takes as input both the telemetry stream and a snapshot of the current model version. In SereBench, robustness evaluation combines two complementary components: a Gaussian boundary stress evaluation and an HSJA-based adversarial evaluation. The Gaussian boundary stress evaluation follows the path represented in Fig. 3 by the Adversarial Perturbation Engine. Before reaching the robustness evaluation stage, the telemetry stream generated by the simulator is processed by this engine, which applies the configured Gaussian perturbation rules to the selected braking-pressure channels. The resulting perturbed telemetry stream is then used to evaluate the current model on Gaussian-stressed observations, producing robustness metrics such as accuracy, precision, recall, and F1-score. This branch quantifies how classifier performance changes when class separability is reduced through controlled, domain-constrained perturbations. The HSJA-based adversarial evaluation is instead executed within the Robustness Evaluation block itself. Unlike the Gaussian stress branch, it does not rely on a predefined perturbation distribution applied to the telemetry stream. The attack interacts with the current classifier under hard-label black-box access, queries the model, and searches for adversarial examples that cross the decision boundary while remaining close to the original observations. In addition to the same classification metrics, the HSJA branch reports the average L2 perturbation between original and adversarial samples and the average number of model queries required by the attack. Both components are executed at predefined intervals and do not update the model parameters. They are used only for robustness assessment, allowing SereBench to compare controlled Gaussian boundary stress with decision-based black-box adversarial evaluation.

When the scenario definition describes a multi-vehicle deployment, the **Federated Learning** stage can be enabled. This stage collects the parameters of the models used in each vehicle and performs an FL aggregation, where locally trained models are periodically aggregated to produce a shared model. The aggregated model is then redistributed to all vehicles and used as the starting point for subsequent local model updates. This procedure allows us to observe how robustness-related behavior changes across nodes after aggregation when vehicles are exposed to different local perturbation levels. The aggregation rule is configured at the scenario level, so the same multi-vehicle workflow can be executed with different federated aggregation strategies without changing the local vehicle pipeline.

The FL communication flow can also be configured to include network impairments. In the current implementation, packet loss and delay can be applied to the exchange of model parameters between the vehicle nodes and the FL Manager. These impairments affect both the uplink direction, where vehicles send their local model weights to the FL Manager, and the downlink direction, where the FL Manager redistributes the aggregated global weights to the vehicles. Packet loss is modeled before message submission to the communication layer, by dropping a message according to a configurable probability. Delay is modeled by postponing message delivery according to configurable latency and jitter values. In this way, the federated workflow can be executed not only under ideal communication, but also under controlled conditions where model updates may be delayed or missing. This mechanism is not used as a full packet-level simulator of an operational railway communication network. Its purpose is narrower: it provides configurable communication impairments for the FL workflow, allowing SereBench to evaluate how federated aggregation strategies behave when model updates are delayed or lost.

Table 2
Default Gaussian perturbation intensity per vehicle configured in SereBench.

Vehicle	Noise Std.	Role in evaluation
Angela	0.0	Clean reference node
Bob	0.7	Mildly stressed node
Claude	1.4	Moderately stressed node
Daniel	2.1	Strongly stressed node

5. Experimental evaluation

We conduct a set of controlled experiments in a fully simulated railway CPS environment by using the proposed SereBench platform. The aim of the following experiments is to demonstrate the ability of SereBench to enable adversarial robustness benchmarking in a controlled braking-oriented railway CPS case study and the type of insights that can be obtained using the framework. In the following, we describe the case study considered for the experiments, the evaluation goals, as well as the obtained results.

5.1. Case study

We model a networked railway CPS composed of four simulated vehicles, namely `Angela`, `Bob`, `Claude`, and `Daniel`. Each vehicle operates as an independent node and runs the workflow described in Section 4.3 and shown in Fig. 3. Moreover, our deployment also makes use of a cloud-hosted *Weights and Biases* (W&B)[3] dashboard, where all the metrics collected during the experiments are sent.[4]

As a reminder, SereBench evaluates model robustness through the complementary mechanisms described in Section 4.2. The Gaussian boundary stress evaluation injects controlled Gaussian noise into selected braking-pressure channels, whereas the HSJA-based evaluation applies a decision-based hard-label black-box adversarial procedure to the current classifier. In the Gaussian boundary stress setting, the default noise intensity is configured on a per-vehicle basis to induce heterogeneous local stress conditions across nodes, as reported in Table 2. Unless otherwise specified by experiment-specific overrides, `Angela` receives no injected Gaussian stress, whereas `Bob`, `Claude`, and `Daniel` experience progressively larger perturbation levels.

Each vehicle hosts a local neural classifier, which represents the *model under test* for the evaluation. The classifier architecture is selected through the `anomaly_detection.model_type` configuration parameter. In the experimental campaign, SereBench considers three interchangeable architectures: a **Multi-Layer Perceptron** (MLP), a one-dimensional **Convolutional Neural Network** (1D-CNN), and a **ResNet**-style feed-forward model for tabular data. The MLP is retained as the default baseline, while 1D-CNN and ResNet are included to show that the same benchmarking workflow can be applied to different classifier backbones.

At the beginning of each simulation run, the selected classifier is initialized with random weights and no pretraining, so that the observed behavior depends on the interaction between the learning process, the simulated system dynamics, and the configured robustness evaluation mechanisms. All considered architectures share the same input and output dimensionality and include a final two-dimensional linear representation immediately before the output layer. This common design keeps the diagnostic visualizations comparable across models and avoids tailoring the benchmark to a single classifier architecture.

Unless otherwise specified, each experimental configuration is executed over five independent runs with different random seeds, namely 42, 123, 456, 789, and 1234. Within a single run, all vehicles use the same seed, while different runs use different seeds. This setup supports reproducibility and reduces the dependence of the observed trends on a single random initialization. The communication-stress analysis included in ET3 is executed with two independent seeds, namely 42 and 123, and the MLP architecture only, in order to keep the analysis focused on the effect of FL-flow communication impairments rather than on architectural variability.

According to the workflow described in Section 4.3, SereBench can be used to evaluate Federated Learning settings. When FL is enabled, vehicles exchange model updates at predefined synchronization points while retaining their local telemetry. An aggregation step produces a shared model that is redistributed to all nodes and used as the initialization for subsequent local updates. In this evaluation, FL can be instantiated with three aggregation strategies: **Federated Averaging** (FedAvg) [45], **FedProx** [46], and **FedYogi** [47]. This allows SereBench to compare robustness-related behavior across the complete FL configurations associated with FedAvg, FedProx, and FedYogi when vehicles are exposed to heterogeneous local stress conditions.

5.2. Evaluation goals

With the aim of demonstrating the ability of SereBench to enable adversarial robustness benchmarking in the considered railway CPS case study, we conduct a set of experiments targeting the following types of evaluation:

- **ET1:** Understand if and how the robustness of a classifier degrades as adversarial perturbations progressively push samples toward decision boundaries under hard-label black-box assumptions;
- **ET2:** Evaluate if noise-augmented training improves the robustness of a classifier in railway CPS settings;
- **ET3:** Analyze how federated aggregation affects robustness-related behavior across vehicles, both under heterogeneous local stress conditions and under controlled communication impairments affecting the FL model-exchange flow;

To this aim, we executed the experiments summarized in Table 3.

First, we performed robustness benchmarking experiments, where the model under test is evaluated through the Gaussian boundary stress and HSJA-based adversarial evaluation mechanisms described in Section 4.2. We evaluate six configurations: two non-federated settings for ET1 and ET2, and four partial noise-augmented training settings for ET3, including a non-federated reference and three federated aggregation strategies. These configurations allow us to isolate the effects of noise-augmented training, model aggregation, and their interaction under heterogeneous local stress conditions, addressing evaluation types **ET1**, **ET2** and **ET3**. In addition, within **ET3**, we include a focused communication-stress analysis in which packet loss and delay are applied to the FL model-exchange flow. This additional analysis keeps the local telemetry and training pipeline clean and stresses only the exchange of model weights between vehicles and the FL Manager, in order to evaluate how aggregation strategies react to delayed or missing model updates. We assess robustness through periodic robustness benchmarking phases executed during system operation. In each phase, we evaluate the current classifier version on Gaussian-stressed observations and HSJA-based adversarial examples (depending on the vehicle). We collect the resulting metrics, which include Accuracy, Precision, Recall, and F1-score. In addition to scalar indicators, each benchmarking phase produces visualization artifacts for qualitative analysis. For each vehicle, we log a two-dimensional PCA projection of the input space, a two-dimensional projection of the classifier's penultimate-layer representation, and the same representation colored by predicted labels. These visualizations support inspection of representation stability and boundary distortion under boundary stress and adversarial evaluation.

[3] https://wandb.ai/

[4] The dashboard can be accessed at https://wandb.ai/jfcevallos/SEREBench.

Table 3
Main robustness-benchmarking configurations used for ET1–ET3. Each configuration is executed over five independent runs. The additional ET3 communication-stress campaign is described separately in Table 4.

Evaluation Type	Main Experiment (5 runs each)	Noise-aug. training	Federated learning
ET1, ET2	#1	×	×
	#2	✓	×
ET3	#3	Partial	×
	#4	Partial	✓ (FedAvg)
	#5	Partial	✓ (FedProx)
	#6	Partial	✓ (FedYogi)

Table 4
FL-flow communication impairment configurations used within ET3.

Configuration	Packet loss	Delay	Jitter
clean	0%	0 ms	0 ms
loss40	40%	0 ms	0 ms
loss60	60%	0 ms	0 ms
delay4000	0%	4000 ms	1000 ms
delay8000	0%	8000 ms	2000 ms
combo-lo	20%	1500 ms	375 ms
combo-hi	60%	8000 ms	2000 ms

It is important to note that, in experiments #3–#6, the *Partial* keyword in the Noise-augmented Training column indicates that noise-augmented training is enabled only on the stressed vehicles, while `Angela` is kept as the reference node without noise-augmented training. This configuration is used to evaluate how model aggregation affects the robustness-related behavior of a less stressed reference node when the other vehicles are exposed to more challenging local stress conditions.

To make the selected delay values meaningful with respect to the FL exchange cadence, these runs use a weight push and pull interval of 15 s and an aggregation interval of 5 s. The 1.5 s, 4 s, and 8 s delay settings therefore correspond to approximately 10%, 27%, and 53% of the model-exchange interval, respectively. All impairment configurations are applied symmetrically to uplink and downlink model exchanges, while the local telemetry and training-data flows remain unaffected. For the communication-stress analysis, we use the MLP architecture and compare FedAvg, FedProx, and FedYogi under the configurations reported in Table 4. The goal is not to provide a complete packet-level simulation of an operational railway communication network, but to show that SereBench can introduce controlled packet-loss and delay conditions in the FL workflow and support sensitivity analyses of federated aggregation strategies under communication stress.

5.3. Results

Following the experimental plan in Table 3, we first assess the robustness of the model under test through scheduled robustness benchmarking phases. In each phase, the current model version is evaluated through the complementary mechanisms described in Section 4.2. The Gaussian boundary stress component injects zero-mean Gaussian noise into the two global braking channels of the braking system, namely the main reservoir and the brake pipe, and evaluates the model on Gaussian-stressed anomaly and attack observations. The HSJA-based component instead performs a decision-based hard-label black-box adversarial evaluation by querying the current classifier and searching for adversarial examples close to the original observations. Together, these mechanisms allow SereBench to inspect classifier behavior near decision boundaries under controlled stress and adversarial evaluation conditions.

Fig. 4 reports the main robustness-related metrics for each vehicle obtained under different perturbation regimes, with and without noise-augmented training. These results correspond to experiments #1 and #2, according to Table 3. As a reminder, `Angela` is configured as the clean reference node, while `Bob`, `Claude`, and `Daniel` are exposed to increasing Gaussian boundary stress levels, according to Table 2. The plots highlight heterogeneous, node-specific robustness behavior under different perturbation regimes and across classifier architectures. In the configurations without noise-augmented training, the metrics show how classifier behavior changes as the local stress level increases. This provides evidence that SereBench can expose differences in robustness-related behavior under controlled perturbation conditions, rather than only reporting nominal classification performance. The comparison with noise-augmented training further shows how the same benchmark workflow can be used to inspect whether training on more ambiguous observations changes the resulting robustness metrics across vehicles and classifier architectures.

Overall, these results illustrate how SereBench can be used to compare robustness-related behavior across vehicles, model architectures, and training configurations under a common evaluation workflow. The purpose of this analysis is not to identify a universally superior classifier, but to show that the benchmark supports systematic comparison under controlled Gaussian boundary stress and HSJA-based adversarial evaluation. This confirms the ability of SereBench to support the evaluations **ET1** and **ET2**.

To complement the insights suggested by the evaluation metrics, we analyze the diagnostic representation-space plots generated by the target models during benchmarking. These plots report the model's two-dimensional latent representation, corresponding to the penultimate layer, and the same representation colored by predicted labels, allowing us to qualitatively inspect how class organization changes across vehicles, training configurations, and classifier architectures.

Fig. 5 reports the manifold diagnostics under increasing perturbation and training configurations. Specifically, it shows the representation-space plots for `Angela` and `Daniel` when noise-augmented training is disabled, and for `Daniel` when noise-augmented training is enabled. For `Angela`, the latent representations show clearer class organization, which is coherent with the absence of Gaussian boundary stress in the default configuration. Conversely, for `Daniel` without noise-augmented training, the plots show stronger overlap and distortion in the latent space, consistently with the higher local stress level configured for this vehicle. This qualitative evidence suggests that stronger stress conditions make the separation among *Normal*, *Anomaly*, and *Attack* observations more difficult for the considered classifiers. When noise-augmented training is enabled for `Daniel`, the plots provide a qualitative view of how the learned representations change under the same evaluation conditions. In this case, the latent space appears more structured and more consistent with the predicted-label organization. These geometric patterns complement the scalar metrics by showing how representation stability and predicted-label regions vary across vehicles, training configurations, and classifier architectures.

To analyze how federated aggregation influences robustness-related behavior at a less stressed node, we focus on Angela as the reference vehicle. In the considered partial noise-augmented training setting, `Angela` does not use noise-augmented training locally, while the remaining vehicles are exposed to more challenging local stress conditions. This setup allows us to evaluate whether model aggregation influences the robustness-related behavior of a node that does not directly receive the same noise-augmented training exposure. Fig. 6 reports the ET3 FL-configuration comparison for `Angela`. The figure compares the non-federated configuration with three FL aggregation strategies, namely FedAvg, FedProx, and FedYogi, across the considered classifier architectures. The comparison includes standard classification metrics, such as Accuracy, Recall, Precision, and Macro-F1, together with HSJA-specific indicators, namely the average number of queries and the average L2 perturbation. The comparison reflects the complete strategy-specific configurations reported in Appendix A, including their associated learning-rate and model-exchange settings; it should

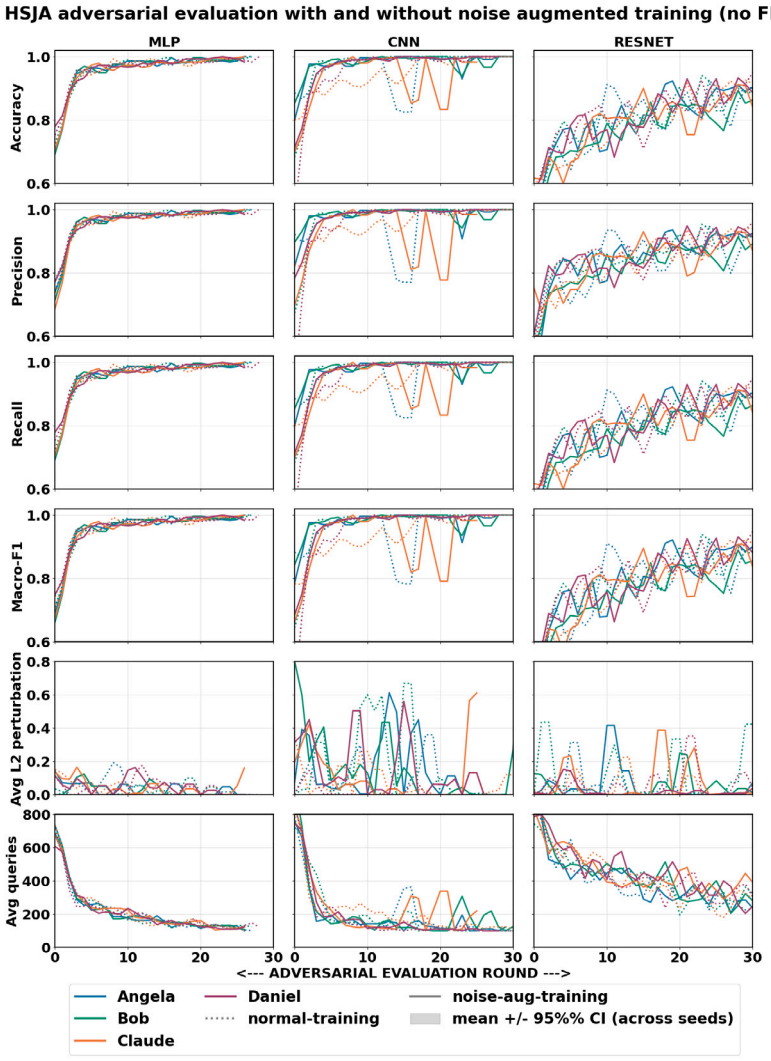


Fig. 4. HSJA-based adversarial evaluation metrics for each vehicle, classifier architecture, and training configuration, with and without noise-augmented training.

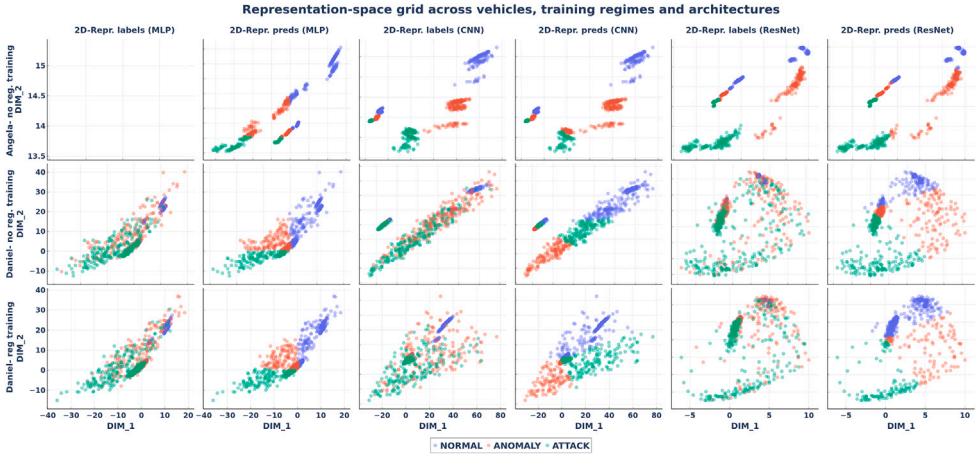


Fig. 5. Manifold diagnostic plots under increasing perturbation and training configurations. For each architecture, we report the 2D hidden representational space (penultimate layer) and the same representation colored by predicted labels.

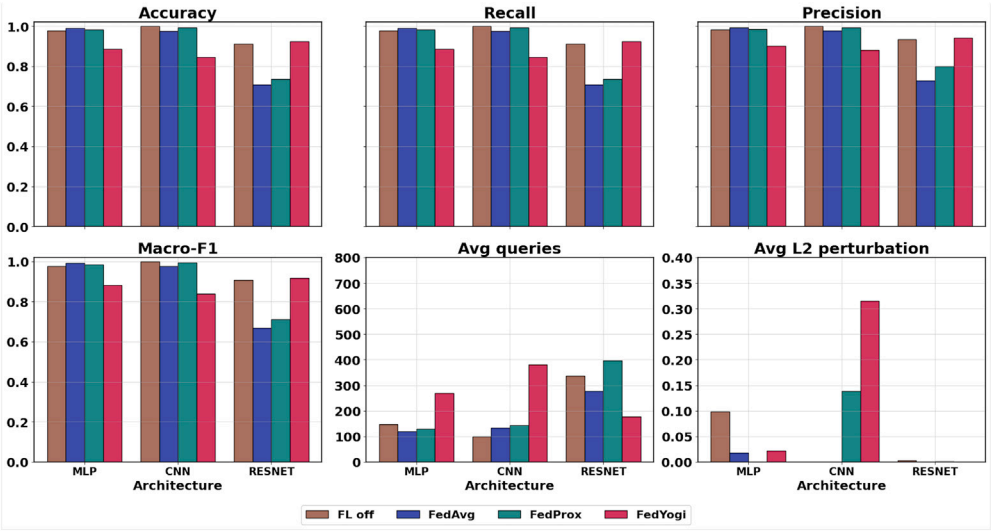


Fig. 6. ET3 comparison of the non-federated reference and the complete FedAvg, FedProx, and FedYogi configurations for Angela across the considered classifier architectures. The figure reports classification metrics and HSJA-specific indicators.

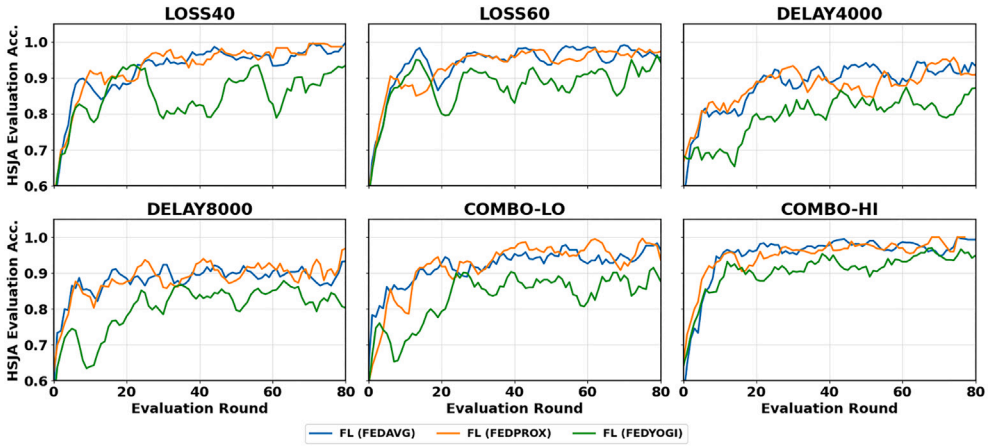


Fig. 7. Fleet-wide HSJA-based Accuracy dynamics under FL-flow communication impairments for the MLP architecture. The curves compare FedAvg, FedProx, and FedYogi under packet-loss, delay, and combined packet-loss/delay configurations. Curves report fleet-wide means over two independent seeds.

therefore not be interpreted as an isolated ablation of the aggregation rule alone. Overall, these results show how SereBench can be used to compare robustness-related behavior across the considered FL configurations under a common evaluation workflow.

Across the configurations considered in the preceding experiments, nominal Accuracy remains high in most cases. Within the simulated benchmark, noise-augmented training therefore does not produce a systematic reduction in nominal classification performance, while changing the robustness-related indicators observed under Gaussian and HSJA-based evaluation. This result should be interpreted within the controlled conditions of the present case study and does not by itself establish deployment readiness in an operational safety–critical system.

To further characterize **ET3**, we also evaluate the impact of communication impairments on the FL model-exchange flow. In this analysis, all vehicles are exposed to the same communication conditions, while the local telemetry generation and local training pipeline are kept unaffected by network impairments. Packet loss and delay are applied only to the exchange of model weights between vehicles and the FL Manager, both in the uplink direction, where vehicles transmit local model updates, and in the downlink direction, where the aggregated global model is redistributed. This setting isolates the effect of impaired FL communication from the effect of heterogeneous local Gaussian stress.

Fig. 7 reports the fleet-wide mean HSJA-based Accuracy averaged over two independent seeds. FedAvg and FedProx generally reach high values and exhibit relatively stable trajectories across the tested conditions, whereas FedYogi shows greater variability in several packet-loss and delay settings. The response is not monotonic with respect to impairment severity: for example, the combined high-impairment configuration maintains high HSJA-based Accuracy for all three strategies. The results therefore indicate that the temporal behavior depends on the interaction between the impairment profile and the aggregation strategy, rather than on the nominal impairment level alone.

Fig. 8 summarizes the fleet-wide mean metrics over two independent seeds. HSJA-based Accuracy and Macro-F1 remain high in most of the tested configurations, whereas the online metrics, and Online Macro-F1 in particular, show greater sensitivity to communication impairments. The most visible reduction occurs under the highest delay setting, although the effect is not uniform across FL configurations or impairment families. FedYogi records lower Online Macro-F1 in several stressed configurations while remaining competitive under others, and FedAvg and FedProx also exchange their relative positions depending on the selected metric. Consequently, the results do not support a universal ranking of the considered FL configurations. Rather, they show that SereBench can expose metric-dependent differences under controlled communication stress.

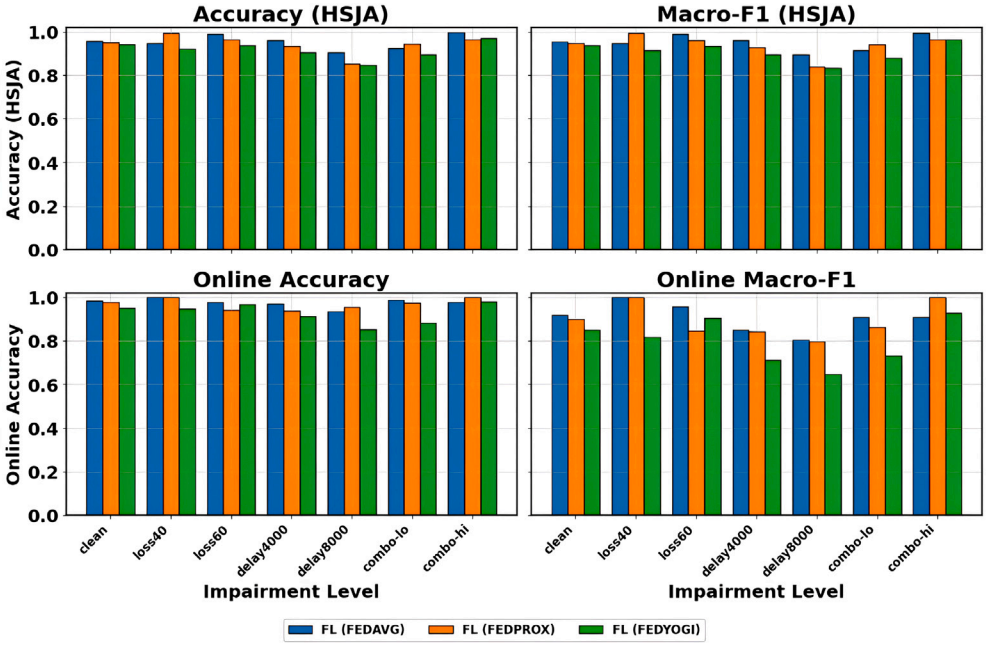


Fig. 8. Communication-stress analysis within ET3 for the MLP architecture. The figure compares FedAvg, FedProx, and FedYogi under clean communication, packet-loss, delay, and combined packet-loss/delay configurations. The reported metrics include HSJA-based Accuracy and Macro-F1, together with Online Accuracy and Online Macro-F1. Bars report fleet-wide means over two independent seeds.

Overall, these results provide a preliminary sensitivity analysis of the FL model-exchange flow under controlled message-drop and delivery-delay conditions. They should not be interpreted as a complete packet-level simulation of an operational railway communication network or as a statistically conclusive ranking of the FL configurations. Instead, they show that SereBench can introduce delayed or missing model updates and evaluate their effects on model aggregation, online classification, and HSJA-based robustness indicators.

The experimental analysis reports multiple metrics collected during robustness evaluation and online monitoring, including Accuracy, Precision, Recall, F1-score, and Macro-F1 where relevant. In addition, HSJA-based evaluation reports attack-specific indicators such as the average L2 perturbation and the average number of model queries. The W&B dashboard is retained as supplementary material for reproducibility and inspection of the complete experiment logs, while the manuscript reports the main metrics needed to support the robustness benchmarking discussion. Overall, the presented results are intended to demonstrate how SereBench supports adversarial robustness benchmarking and federated robustness analysis within the pneumatic braking-subsystem case study considered in this work. ET1 and ET2 show how the framework evaluates classifier behavior under controlled Gaussian boundary stress and HSJA-based hard-label black-box adversarial evaluation. ET3 shows how federated aggregation can be compared under heterogeneous local stress conditions and how the same workflow can be stressed through controlled packet-loss and delay conditions affecting the FL model-exchange flow.

6. Limitations

Although SereBench enables controlled and reproducible robustness benchmarking in railway Cyber–Physical Systems, the scope of the current study entails several limitations that should be made explicit.

The experimental analysis concentrates on the braking subsystem. This choice is deliberate: braking dynamics are safety–critical, physically constrained, and well-suited for interpretable perturbations. At the same time, it restricts the evaluation to a single subsystem. Other components of railway CPS, such as traction control, onboard diagnostics, or signaling interfaces, exhibit different dynamics and coupling patterns. Robustness properties observed in the braking domain should therefore not be assumed to generalize without further investigation. Extending the evaluation beyond the pneumatic braking subsystem would require more than replacing the input variables used by the classifier. Each railway subsystem involves different physical or logical dynamics, failure modes, and admissible perturbation spaces. A traction-oriented evaluation, for example, would need to preserve the relationships among operating mode, voltage, current, and power demand. Signaling- and ERTMS-oriented evaluations would require explicit modeling of discrete supervision states and admissible state transitions, while onboard diagnostics would require interpretable mappings between diagnostic codes, subsystem states, and operational events. A cross-subsystem extension of SereBench would therefore require dedicated class semantics, perturbation constraints, and validation procedures for each target subsystem, supported by domain knowledge and, where available, metadata-rich operational traces.

A second limitation concerns the use of simulated telemetry. While the simulator preserves causal relationships among variables and enforces physically consistent correlations, it cannot capture all sources of variability present in operational data. Effects such as sensor aging, environmental influences, undocumented firmware behavior, or maintenance-induced drift are only partially represented. In this respect, the robustness scores obtained in this work should be interpreted as comparative indicators under controlled stress conditions, rather than as absolute measures of operational robustness.

The experimental evaluation considers multiple classifier architectures and federated aggregation strategies, namely MLP, 1D-CNN, and ResNet-style classifiers, together with FedAvg, FedProx, and FedYogi. Nevertheless, these configurations should be interpreted as representative benchmark instances rather than as an exhaustive exploration of all possible model backbones and FL algorithms. The purpose of the experimental campaign is not to identify the best-performing architecture or aggregation rule in absolute terms, but to demonstrate that SereBench can support systematic robustness comparisons across different model, training, and aggregation configurations. Future work

may extend the evaluation to additional architectures, personalization schemes, adaptive aggregation rules, and defenses against malicious or unreliable participants.

At the communication level, the current implementation provides configurable message-drop and delivery-delay mechanisms, including jitter, on the uplink and downlink exchange of FL model parameters. The communication-stress analysis included in ET3 uses these mechanisms to study how missing or stale model updates affect the considered aggregation strategies. This extension remains a controlled abstraction of the FL model-exchange flow rather than a complete packet-level model of an operational railway network. In particular, bandwidth limitations, network topology, transport- and link-layer scheduling, retransmission and recovery mechanisms, and fully asynchronous aggregation protocols are not explicitly represented. The reported results should therefore be interpreted as a preliminary sensitivity analysis under controlled FL-flow impairments, rather than as a complete validation of federated deployment under all railway communication conditions. Future work may combine the current mechanisms with explicit bandwidth models, network-aware scheduling, and asynchronous aggregation policies.

Finally, the adversarial evaluation considered in this work combines controlled Gaussian boundary stress with HSJA-based hard-label black-box evaluation. This design is appropriate for studying decision-boundary fragility under physically interpretable braking-pressure perturbations, but it is not intended to exhaustively cover the full spectrum of adversarial threats. Temporal, adaptive, universal, surrogate-based, and process-aware attacks are outside the scope of the present experimental campaign. Integrating these attack families into SereBench is an important direction for future work, especially to assess whether the robustness trends observed in the pneumatic braking-subsystem case study remain stable under more adaptive railway-specific threat models.

7. Conclusions

This paper presented SereBench, an open-source benchmarking framework for evaluating adversarial robustness and federated resilience of machine-learning-based classifiers in railway Cyber–Physical Systems. The framework addresses a critical gap in current evaluation practices by shifting the focus from aggregate Accuracy to classifier behavior near decision boundaries, under streaming conditions, and hard-label, black-box assumptions. In this work, SereBench was instantiated and evaluated through a pneumatic braking subsystem case study, where pressure dynamics provide a safety–critical and physically interpretable basis for controlled robustness evaluation. By adopting a generative simulation grounded in interpretable railway subsystems, SereBench enables robustness assessments with explicit class semantics and verifiable ground truth. The focus on the braking subsystem allows perturbations to be expressed in mechanically meaningful terms, ensuring that boundary stress tests remain confined to operationally plausible regions of the feature space. This design choice makes robustness scores interpretable as properties of the system under stress, rather than as artifacts of uncontrolled data generation. At the same time, the results should be interpreted as braking-oriented evidence and should not be assumed to generalize automatically to traction, signaling, onboard diagnostics, or other railway subsystems without additional subsystem-specific validation.

The experimental results show that, in the considered scenario, classifier behavior changes under controlled boundary stress and HSJA-based adversarial evaluation, even when nominal monitoring performance remains stable. Noise-augmented training affects the resulting robustness-related metrics by exposing the classifier to more ambiguous observations during training, while federated aggregation makes it possible to study how robustness-related behavior changes across vehicles operating under heterogeneous local stress conditions. A preliminary communication-stress analysis further showed that message loss and delivery delay affect online and HSJA-based metrics differently across the considered FL configurations, without identifying a single configuration that is uniformly superior under all tested conditions. These findings reinforce the need to treat robustness as a first-class evaluation objective in safety–critical CPS, rather than as a secondary outcome of nominal classification performance. Overall, SereBench provides a reproducible and extensible foundation for studying the resilience of learning-based detectors under controlled railway CPS conditions, as demonstrated in this work through a pneumatic braking subsystem case study. By making decision-boundary behavior observable and measurable under controlled and physically interpretable conditions, the framework contributes toward the systematic evaluation of dependable AI components for safety–critical railway systems.

Several directions for future work naturally emerge from this study. Extending the framework to additional railway subsystems, such as traction control or onboard diagnostics, would allow robustness properties to be evaluated across a wider range of operational behaviors and coupling patterns. Integrating selected real-world telemetry streams, when available and sufficiently metadata-rich, would further support validation of simulation-based findings and improve external validity. On the learning side, future experiments could explore personalization mechanisms, robustness-aware optimization techniques, and the impact of more complete communication models, including bandwidth constraints, network topology, protocol-level scheduling, and fully asynchronous aggregation. Finally, incorporating more complex temporal or representation-level adversarial strategies while preserving physical plausibility would allow SereBench to support a broader range of robustness evaluations.

CRediT authorship contribution statement

Alessandra Rizzardi: Writing – review & editing, Writing – original draft, Validation, Supervision, Funding acquisition, Methodology, Formal analysis, Conceptualization, Data curation. **Raffaele Della Corte:** Writing – review & editing, Writing – original draft, Validation, Supervision, Funding acquisition, Methodology, Formal analysis, Conceptualization, Data curation. **Jesús F. Cevallos-Moreno:** Writing – review & editing, Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization, Data curation. **Simona De Vivo:** Writing – review & editing, Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization, Data curation. **Sabrina Sicari:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Formal analysis, Conceptualization, Data curation. **Domenico Cotroneo:** Writing – review & editing, Writing – original draft, Validation, Supervision, Funding acquisition, Methodology, Formal analysis, Conceptualization, Data curation. **Alberto Coen-Porisini:** Writing – review & editing, Writing – original draft, Validation, Supervision, Funding acquisition, Methodology, Formal analysis, Conceptualization, Data curation.

Supplementary information

All code and configuration files required for experiment reproduction are available in our public project's repository at https://github.com/DIETI-DISTA-IoT/SereBench.

Declaration of competing interest

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

Acknowledgments

This work was supported by the SERENA-IIoT project, which has been funded by EU — NGEU, Mission 4 Component 1, CUP J53D23007090006, under the PRIN 2022 (MUR -*Ministero dell'Università e della Ricerca*) program (project code 2022CN4EBH).

Appendix. Experimental settings and hyperparameters

This appendix documents, in full, the configuration of every experiment in the campaign: the data-generation regime, the anomaly-detector architectures, the local training loop, the federated-learning aggregators, and the evaluation protocol. Our aim is reproducibility at the level of a single command line: all runs are produced by one driver script, `tests/experiments_all.py`, operating on a single, version-controlled configuration tree under `config/`. We first describe the configuration and override mechanism that makes the campaign self-consistent across architectures and aggregation strategies (Appendix A.1); we then give the base parameters shared by all runs (Appendix A.2), the evaluation protocol (Appendix A.3), the experimental matrix (Appendix A.4), and the exact parameter deltas introduced by each override profile (Appendix A.5).

A.1. Configuration and override mechanism

The `config/` directory is the single source of truth for the whole platform. The base file `config/default.yaml` contains a complete, runnable description of every subsystem (dashboard, producers, consumers, the Weights &Biases logger, the federated-learning manager, and the security manager), organized into top-level blocks: `default_vehicle_config` and a per-vehicle `vehicles` list (data generation and adversarial-noise parameters), `default_consumer_config` (the local training and evaluation loop), `anomaly_detection` (the classifier architecture), `federated_learning` (the aggregation node), and `wandb` (logging). Configuration never reaches a container through command-line arguments or environment variables (beyond immutable bootstrap values such as the Kafka broker address and the container name); instead, the dashboard serializes the merged configuration and delivers it to each container as an HTTP `POST` to `/configure`, or as the `params` payload of a `/command` request in the case of federated-learning start/stop.

Experiments are not separate config files but *deltas* layered on top of `default.yaml`. The driver applies them through the dashboard CLI (`tests/dash_cli.py`) in three primitive operations, executed once per run:

- `init-config` — (re)initializes the persisted CLI state from `default.yaml`, giving every run an identical, clean starting point.
- `apply-override <profile>` — deep-merges `config/overrides/<profile>.yaml` onto the current state. The merge is recursive for mappings but *replacing* for any scalar or list value, reproducing OmegaConf/Hydra merge semantics. One important consequence is that the `vehicles` list is a list and is therefore *replaced wholesale* by any override that mentions it: an override that re-tunes one vehicle must restate all four. Multiple profiles may be applied in sequence, each deep-merged on top of the previous, so the layering order is significant.
- `set <dot.path> <value>` — overrides a single leaf key. The driver uses this for the two per-run quantities that are not part of any profile: the random seed (written to both `default_consumer_config.seed` and `default_vehicle_config.seed`, so that classifier initialization and the data-generation simulator share one seed end to end) and the `wandb.run_name`/`wandb.group` labels.

A run is therefore fully specified by the tuple *(ordered list of override profiles, model architecture, seed)*. Because the experiment matrix itself is defined exactly once — as the `EXPERIMENTS` dictionary in `tests/experiments.py` — and *imported* by the per-architecture runners (`experiments_cnn.py`, `experiments_resnet.py`) and by the consolidated runner `experiments_all.py`, the three architectures cannot drift apart: they execute the same cells with only one additional architecture override interposed.

A.2. Base configuration shared by all runs

Every run inherits the parameters in Table 5 from `default.yaml`; experiment-specific overrides change only a small, explicitly enumerated subset of them (Appendix A.5). We group the base parameters into the four functional areas that matter for the campaign.

Data generation and the fleet. The simulated fleet consists of four vehicles — *Angela*, *Bob*, *Claude*, *Daniel* — each driven by the railway telemetry simulator `train_simulator.py` (from `of-core`). The simulator does not sample features independently; it follows physically grounded operational profiles (traction → cruising → coasting → service braking → emergency braking) and emits 40-dimensional observations, each labeled `NORMAL`, `ANOMALY`, or `ATTACK`. The inter-arrival rates (`mu_anomalies`= 0.157, `mu_normal`= 0.117) and the shape parameters (`alpha`= 0.05, `beta`= 1.9) are identical across vehicles and across all experiments; time is emulated at 1/100 of real time to accelerate the simulation. The per-vehicle Gaussian-stress standard deviations (`Mp_std`, `Bp_std`) form the canonical stress gradient 0.0/0.7/1.4/2.1 for Angela/Bob/Claude/Daniel in the base file; as explained below, the campaign manipulates these only on the *training* side, never on the evaluation side.

Classifier architectures. Three architectures are evaluated, selected through `anomaly_detection.model_type`: an `mlp` (the default), a 1D convolutional network (`cnn`) over the feature vector, and a tabular ResNet (`resnet`, pre-activation residual MLP blocks following Gorishniy et al. 2021). Crucially, all three share the same two-dimensional manifold (latent) layer immediately before the output head, so that decision boundaries are directly visualizable and — more importantly for this study — the HopSkipJump attack, the manifold projections, and the federated weight aggregation operate identically regardless of architecture. The shared hyperparameters (hidden width 128, one hidden block, layer normalization, dropout 0.1, 40-dim input, 3-way output, Xavier initialization, SGD at learning rate 0.05) are listed in Table 5. The CNN and ResNet profiles change *only* `model_type`; they inherit every other hyperparameter, which is what makes the cross-architecture comparison fair.

Local training loop (consumer). Each vehicle's consumer maintains a clean telemetry buffer (capacity 5000), retrains in mini-batches of 32 at a high cadence (`training_freq_seconds`= 0.01, `epoch_size`= 20), and runs the adversarial benchmarks every `run_benchmarks_freq_epochs`= 5 epochs, emitting heavier diagnostic plots once every fifth benchmark round. When noise-augmented training is enabled for a vehicle, the consumer augments its training batches with perturbed observations drawn from the producer's dedicated evaluation stream (whose noise level is set by `eval_Mp_std`/`eval_Bp_std`, falling back to the live-stream `Mp_std`/`Bp_std` when `null`); whether this augmentation is enabled is controlled by the `adversarial_training` flag, which is disabled by default. Despite its configuration name, this flag activates the noise-augmented training mechanism described in the manuscript rather than model-dependent adversarial training.

Federated-learning manager. When federated learning is active, the standalone aggregation node subscribes to the per-vehicle `{vehicle}_weights` topics, aggregates with a pluggable strategy, and republishes the global model on `global_weights`. The base file selects `fedavg` with an aggregation interval of 25 s and a weight buffer of 5; the server-side adaptive learning rate (0.01) and the Yogi moment terms (β_1=0.9, β_2=0.999, ϵ=10^{-3}) take effect only under the adaptive aggregators.

Table 5
Base configuration inherited by every run from `config/default.yaml`. Experiment-specific overrides (Appendix A.5) modify only the small subset of these values listed in Table 7; everything else is held fixed across the entire campaign.

Block/parameter	Value	Role
Data generation (`default_vehicle_config`, all vehicles)		
`mu_anomalies/mu_normal`	0.157/0.117	anomaly/normal inter-arrival rate
`alpha/beta`	0.05/1.9	profile shape parameters
`time_emulation`	true (1 : 100)	simulated-time acceleration
`anomaly/diagnostics_classes`	all	full class set
`adversarial_training`	false (default)	training-time noise-augmentation flag
`eval_Mp_std/eval_Bp_std`	null	augmentation-stream std (→ `Mp/Bp_std`)
Fleet noise gradient (`vehicles`, `Mp_std=Bp_std`)		
Angela/Bob/Claude/Daniel	0.0/0.7/1.4/2.1	per-vehicle Gaussian stress
Classifier (`anomaly_detection`)		
`model_type`	mlp | cnn | resnet	architecture (axis of study)
`input_dim/output_dim`	40/3	feature/class dimensions
`h_dim/num_layers`	128/1	hidden width/depth
`layer_norm/dropout`	true/0.1	regularization
`optimizer/learning_rate`	SGD/0.05	local optimizer
`initialization_strategy`	xavier	weight init
Local training loop (`default_consumer_config`)		
`buffer_size/batch_size`	5000/32	buffer/mini-batch
`training_freq_seconds/epoch_size`	0.01/20	retraining cadence
`run_benchmarks_freq_epochs`	5	benchmark period (epochs)
`plot_creation_freq_benchmarks`	5	heavy-plot period (benchmarks)
`weights_push/pull_freq_secs`	40/60	FL weight exchange cadence
`fedprox_mu`	0.01	proximal coeff. (FedProx only)
Evaluation protocol (`default_consumer_config`); see Appendix A.3		
`eval_sigmas`	[0.5, 1.0, 1.5, 2.0]	Gaussian σ-grid (fleet-wide)
`hsja_clean_anchors`	true	HSJA on clean anchors
`hsja_feature_indices`	[32, 33]	attacked features (`usBpPres/usMpPres`)
`hsja_n_per_class`	10	anchors/class/round
`hsja_n_steps`	30	boundary-walk iterations
`hsja_n_grad_samples`	30	gradient-estimation directions
`hsja_init_noise_scale`	3.0	init. adversarial-point std
Federated learning (`federated_learning`)		
`aggregation_strategy`	fedavg (default)	aggregator
`aggregation_interval_secs`	25	server aggregation period
`weights_buffer_size`	5	per-round weight buffer
`learning_rate` (server)	0.01	adaptive LR (FedYogi)
`yogi_beta1/2, epsilon`	0.9, 0.999, 10^{-3}	FedYogi moments
`initialization_strategy`	xavier	global-model init

A.3. Evaluation protocol

The defining methodological choice of this campaign is that the evaluation target is *decoupled* from the training-time noise and is therefore identical for every vehicle, every architecture, and every aggregation strategy. Two complementary robustness probes run synchronously inside the training loop every `run_benchmarks_freq_epochs` epochs.

Primary metric — HopSkipJump on clean anchors. HopSkipJump (HSJA; Chen et al. 2020) is a decision-based, query-only attack that searches for the minimal perturbation crossing the model's decision boundary. With `hsja_clean_anchors=true`, the attack starts from *clean* `ANOMALY/ATTACK` samples drawn from the consumer's own clean buffers rather than from the producer's noisy evaluation stream. As a result the reported `hsja_adv_eval/avg_perturbation` (how far the boundary is) and `avg_queries` (how expensive the attack is) measure the geometry of the *model's* decision boundary, not the incidental noise of any one vehicle's stream, and are thus directly comparable across the fleet. The attack is restricted to the two pressure features `usBpPres/usMpPres` (indices [32, 33]) via `hsja_feature_indices`, matching the Gaussian threat model. Each round attacks 10 anchors per class, walking the boundary for 30 iterations with 30 random directions for gradient estimation and an initial adversarial point found at noise scale 3.0. HSJA runs in a background thread and self-skips if a previous round is still active, so it never inflates the wall-clock of a run.

Secondary metric — fixed Gaussian σ-grid. The function `sigma_grid_evaluation` samples the clean buffers (∼100 observations per class) and, for each $\sigma \in \{0.5, 1.0, 1.5, 2.0\}$ (`eval_sigmas`), injects zero-mean Gaussian noise of standard deviation σ *only* on the two pressure features of `ANOMALY` and `ATTACK` observations, leaving `NORMAL` clean. It then logs accuracy, weighted precision/recall/F1, and macro-F1 under the W&B sub-group `adv_eval/sigma_<σ>/`. This yields a *robustness curve* (one operating point per σ) rather than a single confoundable scalar, and because the grid is a configuration constant it is identical for the whole fleet regardless of how each model was trained. The probe is guarded: it is skipped if any of the three clean buffers holds fewer than ten samples, and its cost (a handful of forward passes over ∼300 samples) is negligible. This decoupling makes the campaign comparisons interpretable: changes in a robustness curve can be related to the corresponding training and FL configuration, because the evaluation procedure remains unchanged across the campaign.

A.4. Experimental matrix

The campaign is organized into two blocks, summarized in Table 6. All six cells share the decoupled evaluation target of Appendix A.3; they differ only in the training-side intervention and, within Block B, in whether and how the local models are federated.

Block A — ET1/ET2: the effect of noise-augmented training (no FL). Two cells isolate the contribution of noise-augmented training on standalone models. The baseline `et2-noadv-nofl` applies no override: all four vehicles retain the default noise gradient but train on clean

Table 6
The experimental matrix, defined once in `tests/experiments.py` and shared by all architecture runners. The "Override(s)" column lists the profile(s) deep-merged onto `default.yaml`, in order; "–" denotes no override (pure defaults). Every cell uses the identical decoupled evaluation target (Appendix A.3).

#	W&B group	FL	Override(s)	Intervention
Block A — ET1/ET2 (noise-augmented-training effect, no FL)				
1	`et2-noadv-nofl`	no	–	clean training, default gradient
2	`et2-adv-nofl`	no	`exp_advtraining`	noise-augmented training, default gradient
Block B — ET3 (effect of FL aggregation on clean reference vehicle Angela)				
3	`et3-freerider-nofl`	no	`exp_et3_freerider`	non-federated reference
4	`et3-freerider-fedavg`	yes	`exp_et3_freerider`	FedAvg (default aggregator)
5	`et3-freerider-fedprox`	yes	`exp_et3_freerider` + `fedprox`	FedProx (μ=0.01)
6	`et3-freerider-fedyogi`	yes	`exp_et3_freerider` + `fedyogi`	FedYogi (adaptive server LR)

data (`adversarial_training` flag set to `false`). The `et2-adv-nofl` configuration override applies the `exp_advtraining` profile, which sets adversarial_training flag to `true` for the whole fleet while preserving the per-vehicle augmentation gradient (0.0/0.7/1.4/2.1). Because the evaluation procedure is fixed, the difference between the two cells' robustness curves can be attributed to noise-augmented training rather than to a change in evaluation difficulty.

Block B — ET3: effect of federated aggregation on a clean reference vehicle. The four Block B cells share the `exp_et3_freerider` configuration profile, which assigns a common training-augmentation noise level of 1.0 to the fleet while keeping Angela as the clean reference vehicle. Angela trains exclusively on clean data, whereas Bob, Claude, and Daniel use noise-augmented training with a noise level of 1.0. The profile also fixes the evaluation settings through `hsja_clean_anchors=true` and `eval_sigmas=[0.5, 1.0, 1.5, 2.0]`, so that the evaluation procedure remains unchanged across the four cells. Cell #3 provides the non-federated reference, whereas cells #4–#6 enable the complete FL configurations associated with FedAvg, FedProx, and FedYogi. Differences observed for Angela relative to the non-federated reference therefore reflect the introduction of federated model exchange under each complete configuration. Since the FL cells also differ in their learning-rate and model-exchange settings, the comparison should not be interpreted as an isolated ablation of the aggregation rule. The exact configuration differences are reported in A.5.

Architecture axis, seeds, and run accounting. Each of the six cells is executed on all three architectures. The consolidated runner `experiments_all.py` interposes one further override per architecture — none for `mlp` (the default), `cnn` for the 1D-CNN, `resnet` for the tabular ResNet — applied *after* the experiment's own overrides, and appends a `-cnn` / `-resnet` suffix to the W&B group and run name (no suffix for `mlp`, so MLP results land alongside the single-architecture runner's output). With five seeds $\{42, 123, 456, 789, 1234\}$ used for the full campaign, this yields 6 cells $\times$ 3 architectures $=$ 18 groups and $18 \times 5 = 90$ runs. The same seed value initializes both the classifier and the data-generation simulator within a run, so each run is reproducible end to end.

Per-run orchestration. For every run the driver executes a fixed, fully synchronous sequence against the dashboard: `init-config`; the experiment's `apply-override`(s); the architecture override (if any); `set` of seed, run name and group; `start-wandb` (generous 120 s timeout, as W&B can be slow to initialize); `produce-all` (then a short warm-up); `start-automatic-attacks` (so attack-class data is present before classification begins); `consume-all`; `start-federated-learning` (FL cells only); a run window of `--run-duration` seconds (default 600 s); and finally a single `shutdown` that tears the cluster down in order (attacks → FL → consumers → producers → W&B). Vehicle containers are created once (`dash_cli.py create-vehicles`) and reused across all runs; the driver only starts and stops the services inside them. Because the Gaussian σ-grid is cheap and HSJA runs in a self-skipping background thread, the minimum useful run length is governed by the number of benchmark rounds desired (one every five epochs over warm buffers), not by the cost of evaluation.

A.5. Override profiles: exact parameter deltas

Table 7 enumerates every parameter changed by each override profile relative to the base configuration; any parameter not listed for a profile retains its Table 5 value. Two subtleties are worth making explicit, as they affect how the federated cells should be read.

First, the `exp_et3_freerider` profile restates the *entire* `vehicles` list. This is required by the replacing merge semantics of Appendix A.1: a list is not deep-merged, so an override that touches one vehicle must enumerate all four or the others would be dropped.

Second, the FedAvg cell (#4) is configured differently from the other two federated cells. It carries *only* `exp_et3_freerider` and relies on the base file's default aggregation strategy (`fedavg`), thereby inheriting the base weight-exchange cadence (push 40 s/pull 60 s), aggregation interval (25 s), and local learning rate (0.05). The strategy-specific profiles `fedprox/fedyogi`, by contrast, additionally retune the federated cadence — lengthening weight push/pull to 120 s, shortening the aggregation interval to 20 s, and lowering the local learning rate to 0.002 — in addition to selecting their aggregator. We report this faithfully rather than silently homogenizing it; the practical effect is that the FedAvg cell exchanges weights more frequently and trains locally at a higher learning rate than the other aggregators, which should be borne in mind when comparing FedAvg against FedProx/FedYogi.

Reproducing a single cell. As a concrete example, cell #6 on the ResNet architecture with seed 123 is exactly the command sequence `init-config`; `apply-override exp_et3_freerider`; `apply-override fedyogi`; `apply-override resnet`; `set default_consumer_config.seed 123`; `set default_vehicle_config.seed 123`; `set wandb.group et3-freerider-fedyogi-resnet`; `set wandb.run_name et3-freerider-fedyogi-resnet_seed123_run2`, followed by the start/run/shutdown orchestration of Appendix A.4. Every other cell, architecture and seed in the campaign is obtained by the same template with the corresponding entries from Tables 6 and 7.

A.6. ET3 communication-stress campaign

The additional ET3 communication-stress campaign fixes the classifier architecture to the MLP and evaluates FedAvg, FedProx, and FedYogi under the seven impairment profiles reported in Table 4. Each FL-configuration–impairment-profile combination is executed with two independent seeds, yielding 21 configurations and 42 runs. The seed values used in this campaign are 42 and 123.

The local telemetry, training-data, monitoring, and logging flows remain unaffected. Communication impairments are applied only to the uplink transmission of local model weights and to the downlink transmission of aggregated global weights. During these runs, model weights are pushed and pulled every 15 s, while the FL Manager aggregates every 5 s. The implementation profiles retain the repository names `exp_et5_flow_*`, although the corresponding analysis is presented in the manuscript as an extension of ET3.

Table 7
Parameter deltas introduced by each override profile under `config/overrides/`, relative to Table 5. Unlisted parameters are unchanged. "all veh." means the value is applied to every vehicle in the restated `vehicles` list.

Profile	Parameter	Value
`exp_advtraining`	`adversarial_training` (all veh.) (noise gradient)	true unchanged (0.0/0.7/1.4/2.1)
`exp_et3_freerider`	`Mp_std=Bp_std` (all veh.) `adversarial_training`: angela `adversarial_training`: bob/claude/daniel `hsja_clean_anchors`; `eval_sigmas`	1.0 false (clean reference) true true; [0.5, 1.0, 1.5, 2.0]

`fedprox`	`aggregation_strategy`; interval
	`weights_push/pull_freq_secs`
	`learning_rate`; `fedprox_mu`
		fedprox; 20 s
		120/120
		0.002; 0.01

`fedyogi`	`aggregation_strategy`; interval server `learning_rate`; moments local `learning_rate`; push/pull	fedyogi; 20 s 0.01; 0.9/0.999/10^{-3} 0.002; 120/120
`cnn` `resnet`	`model_type` `model_type`	cnn resnet

Data availability

The data that has been used is confidential.

References

[1] Z. Wang, X. Liu, Cyber security of railway cyber-physical system (CPS) – A risk management methodology, Commun. Transp. Res. 2 (2022) 100078, http://dx.doi.org/10.1016/j.commtr.2022.100078.

[2] A. Blázquez-García, A. Conde, U. Mori, J.A. Lozano, A review on outlier/anomaly detection in time series data, ACM Comput. Surv. 54 (3) (2021) 1–33, http://dx.doi.org/10.1145/3444690.

[3] A. Rizzardi, R. Della Corte, J.F. Cevallos-Moreno, S. De Vivo, V. Orbinato, S. Sicari, D. Cotroneo, A. Coen-Porisini, Open-FARI: An open-source testbed for federated anomaly detection in the railway industrial internet of things, in: 2025 International Wireless Communications and Mobile Computing, IWCMC, 2025, pp. 764–769, http://dx.doi.org/10.1109/IWCMC65282.2025.11059481.

[4] J. Vitorino, I. Praça, E. Maia, Towards adversarial realism and robust learning for IoT intrusion detection and classification, Ann. Telecommun. 78 (7) (2023) 401–412, http://dx.doi.org/10.1007/s12243-023-00953-y.

[5] A. Belenguer, J.A. Pascual, J. Navaridas, A review of federated learning applications in intrusion detection systems, Comput. Netw. 258 (2025) 111023, http://dx.doi.org/10.1016/j.comnet.2024.111023.

[6] X. Zhou, M. Kouzel, H. Alemzadeh, Robustness testing of data and knowledge driven anomaly detection in cyber-physical systems, in: 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W, 2022, pp. 44–51, http://dx.doi.org/10.1109/DSN-W54100.2022.00017.

[7] W. Hao, T. Yang, Q. Yang, Hybrid statistical-machine learning for real-time anomaly detection in industrial cyber–physical systems, IEEE Trans. Autom. Sci. Eng. 20 (1) (2021) 32–46, http://dx.doi.org/10.1109/TASE.2021.3073396.

[8] J. Chen, M.I. Jordan, M.J. Wainwright, HopSkipJumpAttack: A query-efficient decision-based attack, in: 2020 IEEE Symposium on Security and Privacy, S&P, 2020, pp. 1277–1294, http://dx.doi.org/10.1109/SP40000.2020.00045.

[9] F. Croce, M. Hein, Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, in: Proceedings of the 37th International Conference on Machine Learning, ICML, 2020, pp. 2206–2216, http://dx.doi.org/10.5555/3524938.3525144.

[10] M.J. Pappaterra, F. Flammini, V. Vittorini, N. Bešinović, A systematic review of artificial intelligence public datasets for railway applications, Infrastructures 6 (10) (2021) 136, http://dx.doi.org/10.3390/infrastructures6100136.

[11] A.E. Labrador Rivas, T. Abrão, Faults in smart grid systems: Monitoring, detection and classification, Electr. Power Syst. Res. 189 (2020) 106602, http://dx.doi.org/10.1016/j.epsr.2020.106602.

[12] Y. Hu, A. Yang, H. Li, Y. Sun, L. Sun, A survey of intrusion detection on industrial control systems, Int. J. Distrib. Sens. Networks 14 (8) (2018) 1550147718794615, http://dx.doi.org/10.1177/1550147718794615.

[13] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, N. Meskin, Cybersecurity for industrial control systems: A survey, Comput. Secur. 89 (2020) 101677, http://dx.doi.org/10.1016/j.cose.2019.101677.

[14] A.L. Suárez-Cetrulo, D. Quintana, A. Cervantes, A survey on machine learning for recurring concept drifting data streams, Expert Syst. Appl. 213 (2023) 118934, http://dx.doi.org/10.1016/j.eswa.2022.118934.

[15] M. Conti, D. Donadel, F. Turrin, A survey on industrial control system testbeds and datasets for security research, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2248–2294, http://dx.doi.org/10.1109/COMST.2021.3094360.

[16] A.M. Koay, R.K.L. Ko, H. Hettema, K. Radke, Machine learning in industrial control system (ICS) security: Current landscape, opportunities and challenges, J. Intell. Inf. Syst. 60 (2) (2023) 377–405, http://dx.doi.org/10.1007/s10844-022-00753-1.

[17] A. Dehlaghi-Ghadim, A. Balador, M.H. Moghadam, H. Hansson, M. Conti, ICSSIM: A framework for building industrial control systems security testbeds, Comput. Ind. 148 (2023) 103906, http://dx.doi.org/10.1016/j.compind.2023.103906.

[18] M. Al-Hawawreh, E. Sitnikova, Developing a security testbed for industrial internet of things, IEEE Internet Things J. 8 (7) (2020) 5558–5573, http://dx.doi.org/10.1109/JIOT.2020.3032093.

[19] H.-K. Shin, W. Lee, J.-H. Yun, H. Kim, Implementation of programmable CPS testbed for anomaly detection, in: 12th USENIX Workshop on Cyber Security Experimentation and Test, CSET 19, 2019, http://dx.doi.org/10.5555/3359012.3359014.

[20] J. Gardiner, B. Craggs, B. Green, A. Rashid, Oops I did it again: Further adventures in the land of ICS security testbeds, in: Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy, 2019, pp. 75–86, http://dx.doi.org/10.1145/3338499.3357355.

[21] M. Amiri-Zarandi, R.A. Dara, X. Lin, SIDS: A federated learning approach for intrusion detection in IoT using social internet of things, Comput. Netw. 236 (2023) 110005, http://dx.doi.org/10.1016/j.comnet.2023.110005.

[22] C. Liu, Y. Dong, W. Xiang, X. Yang, H. Su, J. Zhu, Y. Chen, Y. He, H. Xue, S. Zheng, A comprehensive study on robustness of image classification models: Benchmarking and rethinking, Int. J. Comput. Vis. 133 (2) (2025) 567–589, http://dx.doi.org/10.1007/s11263-024-02196-3.

[23] P. Guo, C. Gong, X. Lin, Z. Yang, Q. Zhang, Exploring the adversarial frontier: Quantifying robustness via adversarial hypervolume, IEEE Trans. Emerg. Top. Comput. Intell. 9 (2) (2025) 1367–1378, http://dx.doi.org/10.1109/TETCI.2025.3535656.

[24] A. Rizzardi, R. Della Corte, J.F. Cevallos M, S. De Vivo, S. Sicari, D. Cotroneo, A. Coen-Porisini, SereWay: Toward security and reliability benchmarking for the railway iIoT, J. Netw. Syst. Manage. 34 (2) (2026) 52, http://dx.doi.org/10.1007/s10922-025-10026-1.

[25] W. Brendel, J. Rauber, M. Bethge, Decision-based adversarial attacks: Reliable attacks against black-box machine learning models, in: International Conference on Learning Representations, ICLR, 2018, http://dx.doi.org/10.48550/arXiv.1712.04248.

[26] D.J. Miller, Z. Xiang, G. Kesidis, Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks, Proc. IEEE 108 (3) (2020) 402–433, http://dx.doi.org/10.1109/JPROC.2020.2970615.

[27] F. Tramer, N. Carlini, W. Brendel, A. Madry, On adaptive attacks to adversarial example defenses, Adv. Neural Inf. Process. Syst. 33 (2020) 1633–1645, http://dx.doi.org/10.48550/arXiv.2002.08347.

[28] T. Belkhouja, J.R. Doppa, Adversarial framework with certified robustness for time-series domain via statistical features, J. Artificial Intelligence Res. 73 (2022) 1435–1471, http://dx.doi.org/10.1613/jair.1.13543.

[29] T. Belkhouja, Y. Yan, J.R. Doppa, Dynamic time warping based adversarial framework for time-series domain, IEEE Trans. Pattern Anal. Mach. Intell. 45 (6) (2022) 7353–7366, http://dx.doi.org/10.1109/TPAMI.2022.3224754.

[30] J. Chen, X. Gao, R. Deng, Y. He, C. Fang, P. Cheng, Generating adversarial examples against machine learning-based intrusion detector in industrial control systems, IEEE Trans. Dependable Secur. Comput. 19 (3) (2020) 1810–1825, http://dx.doi.org/10.1109/TDSC.2020.3037500.

[31] V. Pozdnyakov, A. Kovalenko, I. Makarov, M. Drobyshevskiy, K. Lukyanov, Adversarial attacks and defenses in fault detection and diagnosis: A comprehensive benchmark on the Tennessee eastman process, IEEE Open J. Ind. Electron. Soc. 5 (2024) 428–440, http://dx.doi.org/10.1109/OJIES.2024.3401396.

[32] Y. Zhuo, Z. Yin, Z. Ge, Attack and defense: Adversarial security of data-driven FDC systems, IEEE Trans. Ind. Inform. 19 (1) (2022) 5–19, http://dx.doi.org/10.1109/TII.2022.3197190.

[33] R.R. Maiti, C.H. Yoong, V.R. Palleti, A. Silva, C.M. Poskitt, Mitigating adversarial attacks on data-driven invariant checkers for cyber-physical systems, IEEE Trans. Dependable Secur. Comput. 20 (4) (2022) 3378–3391, http://dx.doi.org/10.1109/TDSC.2022.3194089.

[34] J. Tian, C. Shen, B. Wang, X. Xia, M. Zhang, C. Lin, Q. Li, LESSON: Multi-label adversarial false data injection attack for deep learning locational detection, IEEE Trans. Dependable Secur. Comput. 21 (5) (2024) 4418–4432, http://dx.doi.org/10.1109/TDSC.2024.3353302.

[35] P. Kairouz, H.B. McMahan, Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1–2) (2021) 1–210, http://dx.doi.org/10.1561/2200000083.

[36] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, P.S. Yu, Privacy and robustness in federated learning: Attacks and defenses, IEEE Trans. Neural Networks Learn. Syst. 35 (7) (2022) 8726–8746, http://dx.doi.org/10.1109/TNNLS.2022.3216981.

[37] K. Tam, L. Li, B. Han, C. Xu, H. Fu, Federated noisy client learning, IEEE Trans. Neural Networks Learn. Syst. 36 (1) (2025) 1799–1812, http://dx.doi.org/10.1109/TNNLS.2023.3336050.

[38] K.N. Kumar, C.K. Mohan, L.R. Cenkeramaddi, The impact of adversarial attacks on federated learning: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 46 (5) (2023) 2672–2691, http://dx.doi.org/10.1109/TPAMI.2023.3322785.

[39] X. Xie, C. Hu, H. Ren, J. Deng, A survey on vulnerability of federated learning: A learning algorithm perspective, Neurocomputing 573 (2024) 127225, http://dx.doi.org/10.1016/j.neucom.2023.127225.

[40] A.M. Abdelmoniem, C.-Y. Ho, P. Papageorgiou, M. Canini, A comprehensive empirical study of heterogeneity in federated learning, IEEE Internet Things J. 10 (16) (2023) 14071–14083, http://dx.doi.org/10.1109/JIOT.2023.3250275.

[41] Q. Han, S. Lu, W. Wang, H. Qu, J. Li, Y. Gao, Privacy preserving and secure robust federated learning: A survey, Concurr. Comput.: Pr. Exp. 36 (13) (2024) e8084, http://dx.doi.org/10.1002/cpe.8084.

[42] J. Guo, J. Li, Z. Liu, Y. Xiong, Y. Ma, A.V. Vasilakos, X. Li, J. Ma, LCEFL: A lightweight contribution evaluation approach for federated learning, IEEE Trans. Mob. Comput. 24 (7) (2025) 6643–6657, http://dx.doi.org/10.1109/TMC.2025.3545140.

[43] J. Guo, Z. Liu, S. Tian, F. Huang, J. Li, X. Li, K.I. Kostromitin, J. Ma, TFL-DT: A trust evaluation scheme for federated learning in digital twin for mobile networks, IEEE J. Sel. Areas Commun. 41 (2023) 3548–3560, http://dx.doi.org/10.1109/JSAC.2023.3310094.

[44] M.R. Asghar, Q. Hu, S. Zeadally, Cybersecurity in industrial control systems: Issues, technologies, and challenges, Comput. Netw. 165 (2019) 106946, http://dx.doi.org/10.1016/j.comnet.2019.106946.

[45] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, 2017, pp. 1273–1282, http://dx.doi.org/10.48550/arXiv.1602.05629.

[46] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, in: Proceedings of Machine Learning and Systems, vol. 2, 2020, pp. 429–450, http://dx.doi.org/10.48550/arXiv.1812.06127.

[47] S.J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, H.B. McMahan, Adaptive federated optimization, in: International Conference on Learning Representations, 2021, http://dx.doi.org/10.48550/arXiv.2003.00295.