Future Generation Computer Systems 176 (2026) 108162

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

On effectiveness of AI-based misbehavior detection in medical IoT

Hamid Al-Hamadi [a,*], Ing-Ray Chen [b], Ding-Chau Wang [c], Abdullah Almutairi [d]

[a] Computer Science Department, Kuwait University, Kuwait
[b] Computer Science Department, Virginia Tech, United States
[c] Information Management Department, Southern Taiwan University of Science and Technology, Taiwan
[d] Information Science Department, Kuwait University, Kuwait

ARTICLE INFO

Keywords:
Misbehavior detection
Learning
AI
IoT
Hypoglycemia
CSII

ABSTRACT

Artificial Intelligence (AI) classification techniques are pivotal for misbehavior detection in the Internet of Things (IoT), but their potential for severe failure poses a risk in safety-critical applications. This work introduces a novel statistical methodology to evaluate the operational readiness of these AI systems by quantitatively forecasting their effectiveness throughout the learning process. The significance of our methodology lies in its ability to provide predictive insights into an AI detector's performance, enabling a system architect to make data-driven decisions about deployment. We use two lightweight statistical analysis methods: one to model device compliance and forecast the detector's false negative probability (p_{fn}) of missing a malicious device and its false positive probability (p_{fp}) of misidentifying a benign one, and another to model the learning curve and predict the future misclassification rate. This framework allows a designer to determine precisely when a system has been trained sufficiently to meet predefined safety and reliability targets. We demonstrate the feasibility of our approach on an artificial pancreas system with a smart Continuous Subcutaneous Insulin Infusion (CSII) device, confirming the effective and predictable detection of sophisticated attacks.

1. Introduction

AI classification techniques with learning capabilities play a pivotal role in modern misbehavior detection systems, particularly in identifying malicious IoT devices within cyber-physical systems (CPS). This paper proposes a methodology for evaluating the effectiveness of AI-based misbehavior detection systems equipped with learning capabilities. Our design is based on a self-monitoring module residing within a medical IoT device embedded in a medical CPS [1], or within an unmanned aerial vehicle in a drone CPS [2]. The responsibility of such a self-monitoring module is to monitor an embedded IoT device and detect if the embedded IoT device is malicious. It executes misbehavior detection code and monitors the embedded device. To prevent the case in which the self-monitor module itself is compromised, a secure computation space (e.g., [3,4]) is used for the monitor code. This ensures the execution of the misbehavior detection code by the self-monitoring module in a secure computation space, even in the case of the operating kernel of the embedded IoT device being compromised. The monitor code executed by the self-monitoring module is based on AI classification methods with learning capability deriving from machine learning such as a random forest (RF) [5] for classifying if the embedded device is malicious. After monitoring and collecting data, the self-monitoring module performs data analysis to arrive at the conclusion that the device (being monitored on) is malicious or not. A common problem with AI-based systems is that, despite producing mostly correct outputs, sometimes they can severely fail. When a bad node is classified as a good node, it incurs a false negative. On the other hand, when it classifies a good node as a bad node, it incurs a false positive. The typical design goal is to minimize the false negative probability while satisfying the maximum false positive probability requirement.

In this work, we focus on understanding and evaluating the impact of learning capabilities in AI-based misbehavior detection systems within CPSs. Specifically, we address the key question of how the learning capability of AI-based classification methods can be quantified and how this learning affects the effectiveness of detecting misbehavior in CPSs. To answer this, we propose a methodology to assess and quantify the learning effect while analyzing its influence on the performance of AI-based misbehavior detection systems in CPSs. Traditional AI metrics (e.g. accuracy, precision, recall, etc) also show how well the model learned by quantifying its performance. However, the novelty of our method is that it is a probabilistic model that models the entire spectrum of an AI-based misbehavior detection system behavior. It predicts how the model

* Corresponding author.
E-mail addresses: hamid@cs.ku.edu.kw (H. Al-Hamadi), irchen@vt.edu (I.-R. Chen), dcwang@stust.edu.tw (D.-C. Wang), abdullah.almutairi@ku.edu.kw (A. Almutairi).

https://doi.org/10.1016/j.future.2025.108162
Received 7 May 2025; Received in revised form 14 August 2025; Accepted 21 September 2025
Available online 23 September 2025

performance will improve over time and when learning will level off. This allows the system designer to make a data-driven decision about when to stop the training and deploy the device. We propose two lightweight statistical analysis methods to analyze (1) behavior data collected from monitoring the embedded IoT device and (2) outputs generated from the module itself (e.g., yes/no outputs for misbehavior classification). The first statistical analysis method is for quantifying the compliance degree of an IoT device to assess the system's effectiveness, specifically by computing its false negative probability (p_{fn}), the risk of failing to detect a malicious device, and its false positive probability (p_{fp}), the risk of incorrectly flagging a compliant one. The second statistical analysis method is for quantifying the learning capability of the AI detection method, with the objective of estimating the frequency at which a false positive or a false negative will happen. The end result is a methodology allowing us to quantify the learning capability and analyze its effect on the effectiveness of AI-based misbehavior detection systems. We demonstrate the effectiveness of misbehavior detection code with a self-monitoring module residing in a Continuous Subcutaneous Insulin Infusion (CSII) device embedded within an artificial pancreas system [6] for Type 1 Diabetes (T1D) patients where a constant dosage of insulin infusion is required to maintain a healthy blood glucose level [7]. We test our methodology with a popular AI-based classification method, namely, RF [5] which is known to possess great learning capabilities to improve classification accuracy. The contributions of the paper are as follows:

- We propose a probabilistic forecasting framework to evaluate the effect of learning on the operational readiness of AI-based misbehavior detectors. This allows a system designer to quantify risk and make a data-driven decision about when to deploy a system based on its predicted ability to meet effectiveness requirements (p_{fn} and p_{fp}).
- We develop two lightweight statistical methods that enable a system designer not only to predict the current p_{fn} and p_{fp} but also to forecast the future trajectory of the misclassification rate during training. This transforms evaluation from a static snapshot into a dynamic, predictive process.
- We demonstrate, through comparative analysis, the superiority of our predictive methodology over traditional, reactive techniques like mean-based thresholding and standard early stopping. Our framework provides actionable, forward-looking insights where other methods fall short.
- We validate our methodology by applying it to a critical medical use case–a smart CSII device for T1D patients–and show it can effectively learn to detect a range of sophisticated attacks, including stealthy nocturnal patterns.

The remainder of this paper is organized as follows. Section 2 provides background on CSII devices for T1D patients. Section 3 reviews existing literature and contextualizes our contributions. Section 4 details our system architecture, the behavior model for the smart CSII device, and the various attacker scenarios considered. Section 5 presents our core statistical methodology for forecasting effectiveness and quantifying the learning process. Section 6 describes the preparation of the simulated datasets used for evaluation. In Section 7, we present a comprehensive performance evaluation, including a validation of our model and a comparative analysis against baseline techniques. Section 8 discusses the implications of our findings and acknowledges the study's limitations. Finally, Section 9 summarizes the paper and outlines future research directions.

2. Background of CSII devices for T1D patients in a medical CPS

In this section we provide a general background of Continuous Subcutaneous Insulin Infusion (CSII) devices typically embedded within an artificial pancreas system [6]. Table 1 lists the notations used in our proposed work.

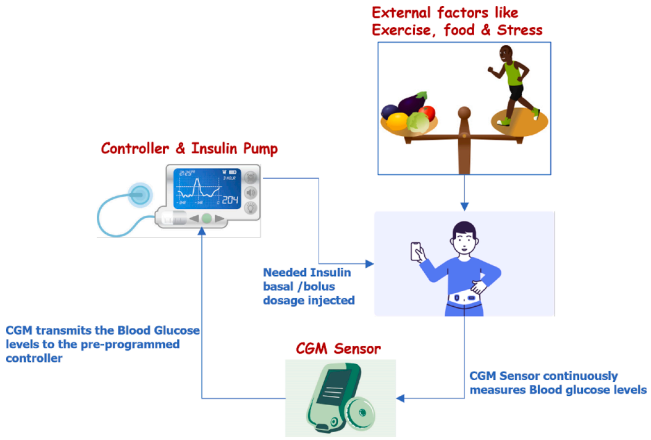


Fig. 1. Artificial pancreas system.

The Artificial Pancreas (AP) is an open-loop insulin delivery CPS which combines an insulin pump CSII device with a Continuous Glucose Monitoring (CGM) device to sense the blood glucose levels subcutaneously. The blood glucose measurements from CGM are transmitted to the insulin pump using radio transmitter followed by infusion of required insulin by the CSII pump. As illustrated in Fig. 1, an AP typically consists of a CSII device for insulin infusion, a CGM device for uninterrupted glucose monitoring, and a control algorithm to auto-adjust insulin dosages [8]. As we will see in the Related Work section, many AI-based control algorithms have been proposed to identify correct insulin dosages for regulating blood glucose levels according to an individual patient's situations.

Most T1D patients rely on CSII devices to help regulate glucose levels between 4.0 and 10.0 mmol/L. If the glucose level falls below 4.0 mmol/L, patients are generally required to consume a carbohydrate-rich meal/drink to quickly elevate their blood glucose and stop it from falling further. While this is a plausible solution most of the time, it is not always feasible, particularly when user interactions are limited, such as at nighttime. A more realistic solution would be to develop smart features in CSII devices that can detect continuously when hypoglycemia events might occur. Smart algorithms could be deployed to readjust insulin infusion rates and to develop personalized glycemic models aimed at simultaneously increasing time-in-target and eliminating hypoglycemia events. Fig. 2 illustrates a scenario, generated by Ulna [9] which is a development platform for an artificial pancreas, where a T1D patient is monitored using a CSII device pre-programed with the algorithm of behavior rules. According to the graph, the algorithm monitors the blood glucose level of the patient for a 24 h simulation period and the CSII pump dispenses the required insulin basal rates. The simulation graph displays meal carbs (red), insulin doses (blue), and basal rates (yellow dashed line) recorded at 10 m intervals. Insulin is infused to stabilize blood glucose levels and prevent hypoglycemia. If glucose is too high, additional insulin is automatically delivered via the CSII pump. Misbehaviors, such as incorrect dosage or failure to adjust insulin based on rapid glucose changes, can endanger the patient. AI algorithms, while mostly accurate, can sometimes produce incorrect outputs, and attacks on the CSII device may also cause misbehaviors. This paper introduces a specification-based misbehavior detection method applicable to medical CPSs with IoT devices, demonstrated using CSII devices. Detecting misbehaviors and eliminating them could prove to be valuable to the patient who uses the CSII device. In this work, we propose a specification-based misbehavior detection method that can be generally applied in a medical CPS with embedded IoT devices to identify a malicious IoT device. We exemplify the proposed specification-based misbehavior method with CSII devices.

Table 1
Notations.

Notation	Description
p_{fn}	False Negative Probability (probability of failing to detect a malicious device)
p_{fp}	False Positive Probability (probability of incorrectly flagging a compliant device)
c_i	The target device's compliance degree in response to the ith event
C_T	Minimum compliance degree below which the target device is considered malicious when an event occurs
$\hat{a}$	Maximum likelihood estimate of a model parameter a
T_{IDS}	Inter-arrival time for event arrivals (i.e., monitoring interval)
t_i	Time at which the ith misclassification output is observed in a testing cycle
n	Number of events (monitoring intervals) to be tested in a testing cycle
θ	Improvement parameter for an AI-based IDS with learning capability in a testing cycle
λ_0	Misclassification output arrival rate at the beginning of a testing cycle
t_t	Total time taken for n event arrivals in a testing cycle
λ	Misclassification output arrival rate estimated at the end of a testing cycle

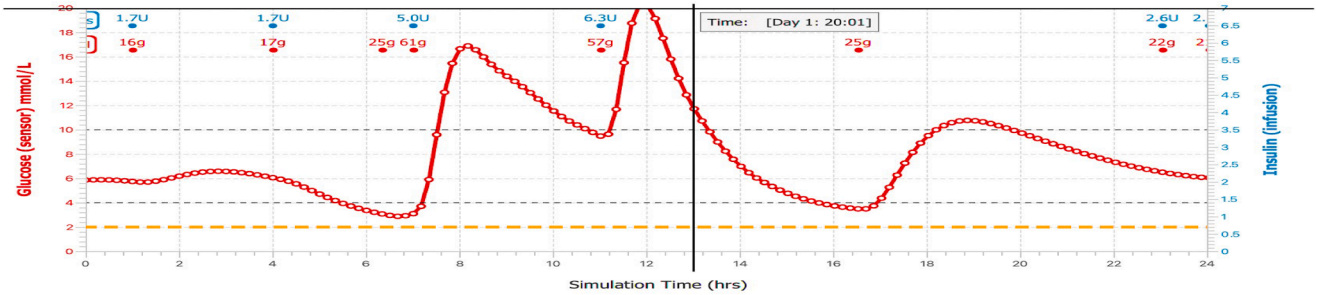


Fig. 2. Continuous glucose monitoring in a CSII device.

3. Related work

Recent advances in intrusion detection for medical cyber-physical systems (MCPS), particularly closed-loop insulin delivery systems, have emphasized adaptive intelligence and explainability. However, many proposed systems lack mechanisms to quantify learning readiness, convergence, and deployment safety–especially in constrained embedded environments.

Recent efforts in MCPS security have focused on detecting anomalies in smart medical environments, particularly those involving insulin pumps and healthcare IoT platforms. A subclass-aware Intrusion Detection System (IDS) was proposed that leverages Deep Subclass Dispersion (SDOSVM) to enhance discrimination of abnormal behaviors in medical IoT traffic. The model, which targets healthcare settings, combines clustering with one-class classification and achieves strong results on the TON_IoT dataset [10]. While effective at class separation, it lacks any modeling of time-based classifier convergence or deployment assurance. An adaptive intrusion detection framework using Bayesian possibilistic clustering and fuzzy classifier ensembles was introduced to address concept drift and evolving IoT conditions [11]. The system shows strong adaptability in dynamic environments but does not offer mechanisms to evaluate classifier readiness or performance stability over time. Reducing false alarms in healthcare environments has also received attention. Maiga et al. developed a deep learning-based IDS that integrates probabilistic clustering with human expert input, aiming to boost interpretability while maintaining alert precision [12]. However, their model does not include trend-aware readiness estimation or predictive statistical tools. In contrast to these systems, our approach provides a time-aware, behavior-grounded statistical framework for operational readiness monitoring in insulin pumps. Using Beta and Poisson-based distributions, we model classifier confidence and improvement dynamics, enabling predictive alerts and safe deployment thresholds.

As IoT deployments increasingly move toward edge and embedded devices, intrusion detection systems are being reimagined for constrained platforms. Statistical modeling and cooperative game theory were used by Rocca to improve IDS interpretability via Shapley value decomposition, adding insight into feature contributions and robustness under adversarial conditions [13]. While the method enhances explainability, it does not quantify classifier convergence or monitor training sufficiency. Benchmarking studies have also explored resource-constrained deployments. Rizvi et al. evaluated various AI-based IDS approaches across different devices, profiling each model's computational footprint and latency under constrained environments [14]. Their results provide deployment insight, but do not examine learning trajectories or readiness indicators. Real-time inference and explainability are key goals in recent IDS design. A modular, explainable IDS designed for edge IoT environments was introduced with the goal of low-latency operation and modular detection [15]. Although the system supports explainable detection, it does not estimate when a model has trained sufficiently or how its trustworthiness evolves. More recently, Rahman et al. introduced AI2DS, a deep autoencoder-based IDS that flags anomalies through reconstruction errors and supports real-time adaptation using only benign traffic during training [16]. Their adaptive thresholding mechanism enables lightweight and semi-supervised intrusion detection, but does not provide pre-deployment safety signals or quantify learning confidence. Farrukh et al. proposed AIS-NIDS, a self-sustaining packet-level IDS system that leverages incremental learning to detect both known and zero-day threats without full retraining [17]. Although it adapts effectively, the system lacks time-aware readiness metrics or convergence analysis. Unlike the works above, our system not only supports lightweight deployment but integrates predictive readiness analytics. It enables proactive risk awareness by tracking classifier behavior over time, which is essential for secure operation in clinical IoT environments.

Only a few works have attempted to model classifier trust or readiness using formal statistical tools. A high-throughput, Field-Programmable Gate Array (FPGA) IDS was developed by Wu and Kondo that supports continuous learning and adaptive feature compression for on-device deployment [18]. Their system enables efficient updates but does not assess whether the model is sufficiently trained or stable for deployment. Efforts to improve generalization and reduce training complexity include Zhao et al.'s adaptive-loss and dimensionally compressed IDS, which shows promising performance across heterogeneous IoT scenarios [19]. While learning-efficient, it does not capture

behavioral changes or readiness convergence over time. Efficiency-focused designs like those from Kaushik et al. utilize statistical feature selection and ensemble learning to enhance robustness and reduce training overhead [20]. Although resource-optimized, the approach lacks monitoring of classifier stability or learning progression. Tewari et al. proposed a quantized and pruned Convolutional Neural Network - Bidirectional Long Short-Term Memory (CNN-BiLSTM) architecture optimized for embedded systems, capable of preserving accuracy under memory and compute constraints [21]. Their method ensures lightweight deployment but does not quantify readiness for operational safety. Our system fills this gap by integrating Beta-based classifier confidence analysis and Poisson-modeled learning speed estimation. These statistical tools provide deployment-time insights that complement traditional accuracy-based evaluation, enabling trustworthy AI deployment in medical IoT devices.

Post-deployment model confidence and behavior calibration have received increasing attention in resource-constrained systems. Simeone and Park propose a conformal calibration framework to provide uncertainty quantification and error detection in wireless environments, enabling lightweight, model-agnostic trust monitoring [22]. While their technique provides offline calibration, it lacks continuous learning assessment or domain-specific behavior analysis. Kebir and Tabia propose a unified framework combining incremental learning, calibration, and explainability to handle both resource constraints and concept drift [23]. While promising, it does not provide operational indicators tied to statistical change detection. Other efforts leverage generative modeling or uncertainty sampling to detect drift in unsupervised contexts. For instance, Hossain and Waqas use Variational Autoencoders (VAEs) for detecting sensor drift in IoT devices via latent distribution shifts [24], while Winter et al. benchmark uncertainty estimation techniques for drift detection without labeled data [25]. Our method contributes a statistical readiness model grounded in behavior tracking and deployment-time evaluation–integrated into a continuous monitoring loop suitable for closed-loop medical CPS.

The threat of adversarial manipulation remains a central challenge for deep learning-based IDS. Adversarial training techniques such as Adversarial Intrusion Detection Training Framework (AIDTF) [26] and Projected Gradient Descent (PGD)-based defenses [27] improve resilience but often lack applicability in constrained clinical settings. Nguyen et al. introduced a Generative Adversarial Network (GAN)-based adversarial IDS training system augmented by explainability modules to improve interpretability during attacks [28]. Rashid et al. use semi-supervised decision trees enhanced with adversarial defense layers to improve detection under blackbox attacks [29]. Güngör et al. develop an adversarial sample selection mechanism for hyperdimensional computing in constrained IDS, showing improved robustness under low-power constraints [30]. In contrast, our work incorporates statistical tracking of classifier behavior and attack-induced deviations–enabling the detection of smart attack patterns on insulin dosing. These indicators serve as early warning signals and strengthen the robustness of medical IDS under stealthy adversarial scenarios.

Threshold-based anomaly detection, which relies on comparing statistical metrics against a predefined or dynamic baseline, is a widely established practice in cybersecurity. For example, recent work in industrial control systems has utilized the mean of network features within a time window, triggering an alert if a feature deviates beyond a set threshold [31]. Similarly, in the context of IoT security, researchers have proposed systems that classify a device as anomalous if the mean of its recent connection patterns crosses a statistically defined threshold when compared to its historical profile [32]. This same principle has been applied to real-time botnet detection, where a device is flagged as malicious if the average rate of its outgoing packets exceeds a specific threshold [33]. While these mean-based thresholding techniques are effective for detecting clear deviations from a norm, they provide a binary, reactive assessment based on past behavior. In contrast, our proposed methodology advances this approach by providing a probabilistic, forward-looking forecast. Instead of a simple binary decision, our model quantifies the evolving probability of future misbehavior (P_{fn}), enabling a more quantitative and predictive risk assessment.

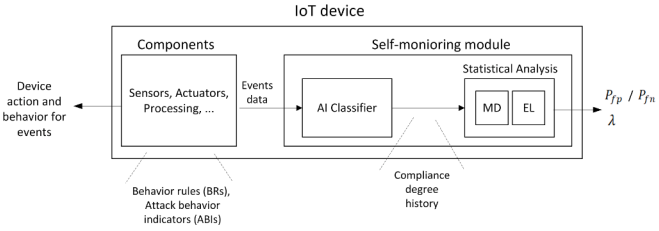


Fig. 3. Overview of the system model.

Recent efforts to enhance early stopping mechanisms focus primarily on improving training efficiency and avoiding overfitting, yet lack predictive capacity. For instance, Paguada et al. propose a generalized early stopping method based on kernel density estimation to minimize overtraining risk while adapting to nonstationary behavior during model training [34]. Similarly, Chen et al. introduce a Bayesian optimization technique that incorporates pseudo-labels to improve hyperparameter search under early stopping constraints [35]. While these approaches contribute to training optimization, they do not model future system behavior or provide probabilistic readiness indicators. In contrast, our approach estimates the arrival rate of misclassification events and forecasts the trajectory of model performance, enabling decisions such as when to halt training or deploy a system based on quantitative risk assessments.

4. System model

4.1. Overview

Fig. 3 presents the general architecture of the proposed system model. While our proposed model is generically applicable to IoT devices, we instantiate it in the context of the CSII device. At its core is the self-monitoring module, which continuously observes the device's behavior by processing events data from its components. Within this module, an AI-based classifier evaluates whether the behavior complies with predefined behavior rules (BRs) and derived attack behavior indicators (ABIs), introduced in Section 4.2. These behaviors may correspond to advanced or stealthy attacks under various scenarios, detailed in Section 4.3. The classifier generates a compliance score per observation, which accumulates into a compliance degree history and is then analyzed statistically, as described in Section 4.4. As detailed in Section 5, this statistical analysis–via the misbehavior detection (MD) and effect of learning (EL) components–enables the system to compute the probabilities of misclassifying bad devices as good (PFN) and good devices as bad (PFP), as well as to estimate the dynamic misclassification arrival rate, $\lambda(t)$. The proposed method offers predictive insights into device behavior, enables informed early stopping of classifier training, and provides a readiness metric that enhances both the reliability and intelligence of IoT device monitoring frameworks.

4.2. Design of smart CSII with misbehavior detection

In this section, we provide the details of misbehavior detection code in the self-monitoring module that monitors the programmable CSII device. The CSII device consists of a CGM sensor inserted under the patient's skin, that continuously measures the interstitial glucose levels, which is the glucose found in the fluid between the cells, as shown in Fig. 4. Subsequently, the CGM sensor transmits the measured glycemic levels to the receiver end of the CSII device which initiates the necessary insulin infusions through the subcutaneous infusion set. Therefore, the glycemic levels of the T1D patients are monitored continuously and

Table 2
CSII device behavior rules.

ID	Behavior Rule	Glycemic Aspect
BR-1	**if** GL is very-high and $GLSlope > 1$ then inc-insulin	hyperglycemia
BR-2	**if** GL is very-high and $GLSlope < -1$ then const-insulin	normal glycemia
BR-3	**if** GL is very-high and $-1 \leq GLSlope \leq 1$ then inc-insulin	hyperglycemia
BR-4	**if** GL is high and $GLSlope > 1$ then inc-insulin	hyperglycemia
BR-5	**if** GL is high and $GLSlope < -1$ then const-insulin	normal glycemia
BR-6	**if** GL is high and $-1 \leq GLSlope \leq 1$ then const-insulin	normal glycemia
BR-7	**if** GL is medium and $GLSlope > 1$ then const-insulin	normal glycemia
BR-8	**if** GL is medium and $GLSlope < -1$ then dec-insulin	mild hypoglycemia
BR-9	**if** GL is medium and $-1 \leq GLSlope \leq 1$ then dec-insulin	mild hypoglycemia
BR-10	**if** GL is low and $GLSlope > 1$ then shutoff-insulin	hypoglycemia
BR-11	**if** GL is low and $GLSlope < -1$ then shutoff-insulin	hypoglycemia
BR-12	**if** GL is low and $-1 \leq GLSlope \leq 1$ then shutoff-insulin	hypoglycemia
BR-13	**if** GL is very-low and $GLSlope > 1$ then shutoff-insulin and invoke-alert	severe hypoglycemia
BR-14	**if** GL is very-low and $GLSlope < -1$ then shutoff-insulin and invoke-alert	severe hypoglycemia
BR-15	**if** GL is very-low and $-1 \leq GLSlope \leq 1$ then shutoff-insulin and invoke-alert	severe hypoglycemia

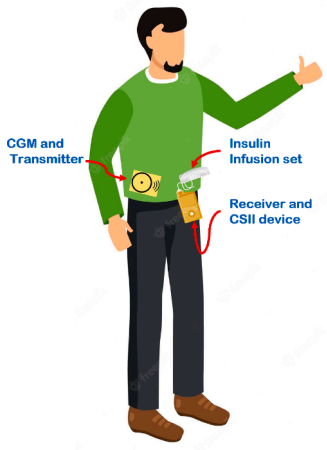


Fig. 4. CSII device receiving CGM measurements through wireless sensor.

Table 3
CSII device attack behavior indicators.

ID	Observation	Attack Indicators
ABI-1	**if** GL is very-high and $GLSlope > 1$	dec-insulin
ABI-2	**if** GL is very-high and $GLSlope < -1$	inc-insulin or dec-insulin
ABI-3	**if** GL is very-high and $-1 \leq GLSlope \leq 1$	dec-insulin
ABI-4	**if** GL is high and $GLSlope > 1$	dec-insulin
ABI-5	**if** GL is high and $GLSlope < -1$	inc-insulin or dec-insulin
ABI-6	**if** GL is high and $-1 \leq GLSlope \leq 1$	inc-insulin or dec-insulin
ABI-7	**if** GL is medium and $GLSlope > 1$	inc-insulin or dec-insulin
ABI-8	**if** GL is medium and $GLSlope < -1$	inc-insulin
ABI-9	**if** GL is medium and $-1 \leq GLSlope \leq 1$	inc-insulin
ABI-10	**if** GL is low and $GLSlope > 1$	inc-insulin
ABI-11	**if** GL is low and $GLSlope < -1$	inc-insulin
ABI-12	**if** GL is low and $-1 \leq GLSlope \leq 1$	inc-insulin
ABI-13	**if** GL is very-low and $GLSlope > 1$	inc-insulin and no-alert
ABI-14	**if** GL is very-low and $GLSlope < -1$	inc-insulin and no-alert
ABI-15	**if** GL is very-low and $-1 \leq GLSlope \leq 1$	inc-insulin and no-alert

insulin infusion is adjusted based on the requirement, thus preventing undesirable and dangerous blood sugar variations. This is particularly helpful in handling the hypoglycemic problems in T1D patients.

Based on the above, we propose a self-monitoring module that performs misbehavior detection at runtime based on a set of pre-defined rules for characterizing misbehavior of the smart CSII device that monitors the glycemic behavior of the T1D patient. Here, the glycemic behavior over a 30-days rolling window is recorded by a CGM sensor. Our behavior rule model characterizes glucose levels into different states: Very High [14-30 mmol/L], High [9-14 mmol/L], Medium [4-9 mmol/L], Low [2-4 mmol/L], and Very Low [0-2 mmol/L]. The model also takes into consideration the gradient of the glucose readings in each state to decide how to alter insulin infusion. To decide if the basal rate needs to be changed, the model applies 15 behavior rules (BR-1 to BR-15) as shown in Table 2 to tweak the rate. If the infusions from the CSII device deviate from the behavior specifications outlined in the behavior rules of Table 2, the misbehavior detection system through its self-monitoring module will detect such misbehaviors irrespective of whether it is caused by inherent faults associated with AI-based algorithms (i.e., AI fails miserably at times) or malicious attacks. This is done by first converting the behavior rules to attack indicator actions as shown in Table 3. The behavior rules set can be converted into the set of attack indicators as shown in Table 3. Every Attack Behavior Indicator (ABI) is represented as a conjunctive normal form (CNF) of the (Observation vs Attack Indication Action) predicate, which is assessed to determine if it yields a value of (1) true or (0) false. This indicates whether the associated behavior rule has been violated or not. In the event of all ABI conditions taking the value of 0, that indicates that not any of the behavior rules are violated, thus the system is in safe operating mode. However, in the event of any of the ABI condition taking the value of 1, that indicates that the matching behavior rule is violated [1]. The self-monitoring module identifies all actions that violate behavior rules, ensuring stability of insulin infusions when the CSII device malfunctions and preventing dangerous glycemic levels. Misbehaving actions, like incorrect insulin infusions, can critically alter glycemic levels. For example, normal levels may be disrupted by incorrect dosages, hyperglycemia may worsen with extra insulin, hypoglycemia can become life-threatening if insulin is not stopped, and alerts may fail to summon medical help.

The attack behavior indicators in Table 3 are explained as follows: In the situations of ABI-1, ABI-3 and ABI-4 when the glycemic aspect is hyperglycemia, the normal action is to increase insulin dosage whereas the misbehavior action is to decrease insulin further causing hyperglycemia. In ABI-2, ABI-5, ABI-6 and ABI-7 when the glycemic aspect is normal glycemia, there should not be any change in insulin infusion whereas the misbehavior action is to either increase or decrease the insulin dosage. In ABI-8, ABI-9, ABI-10, ABI-11 and ABI-12 when the glycemic aspect is hypoglycemia, the normal action is to decrease or shutoff insulin dosage whereas the misbehavior action is to increase the insulin dosage, thus further worsening the hypoglycemic level. In ABI-13, ABI-14 and ABI-15 when the glycemic aspect is severe hypoglycemia, the potentially dangerous hypoglycemic level is aggravated if the misbehavior action is triggered to further increase insulin infusion and prevent alerts from being invoked.

4.3. Attack scenarios

The environment considered is noisy, resulting in imperfect monitoring and occurrence of mis-monitoring probability. Our approach in-

volves representing the likelihood of mis-monitoring, P_e, as a variable that follows a continuous uniform distribution between 'a' and 'b'. Here, both 'a' and 'b' are real numbers within the [0, 1] range, with 'a' being less than or equal to 'b'. Typically, we test three cases: $[a, b] = [0, 10\%]$, [0, 20%], and [0, 30%], corresponding to low, medium, and high noise levels, respectively. To demonstrate the resilience of our proposed AI-based misbehavior detection self-monitoring module against malicious attacks, we consider a wide variety of attack scenarios. Unlike signature-based attacks that can be easily detected by IDSs through comparison against known signature databases, attacks with smart behaviors (e.g. on-off attacks or opportunistic attacks) are far more difficult to detect by an IDS [36], and thus in our work we consider a variety of smart attack behaviors. Specifically, we consider four attack scenarios, namely, Reckless, Random, Insidious and Insidious Nocturnal. The attack behavior indicators of Table 3 are explained for these 4 attack scenarios, as follows:

1. Reckless
   Reckless attacks happen whenever there is an opportunity for the malicious CSII device to attack, i.e., when any ABI event condition among the 15 listed in Table 3 is true.
2. Random
   Random attacks happen with an attack probability of P_a, when the malicious CSII device attacks randomly or inconstantly in order to avoid detection. i.e., when a ABI event condition among the 15 listed in Table 3 is true, the device chooses to attack with probability P_a. We simulate this attack scenario by randomly generating a number in [0,1] and the malicious CSII device attacks when this number is below P_a.
3. Insidious
   The malicious CSII device remains concealed and launches attacks only under specific ABI event conditions where it aims to maximize damage. Among of the 15 ABIs listed in Table 3, ABI-1, ABI-2, and ABI-3 (associated with very high glycemic levels) as well as ABI-13, ABI-14, and ABI-15 (linked to dangerously low glycemic levels) are the most damaging. Consequently, the malicious CSII device, operating in an insidious mode, will attack only when these conditions are met.
4. Insidious Nocturnal
   The malicious CSII device attacks in insidious mode only during nighttime (i.e., between 8pm and 4am) when the patient is asleep. This attack is more dangerous than insidious as the patient will not be aware of the attack

4.4. Assessing compliance degree

Examples of embedded IoT devices in a CPS include sensor, controller or actuators [1,2]. Each IoT device runs on secure trusted hardware [3,4] enabling the associated self-monitoring module using a secure computation space for executing the misbehavior detection code, even in the case of the operating kernel for the IoT getting compromised. The self-monitoring module assesses the compliance degree of the IoT device, c, such that c is 1 if the IoT device is perfectly well-behaved and 0 if the IoT device is misbehaved. An AI classification method such as RF[5] is used for classifying if the embedded IoT device is malicious. Since an RF classifier produces only a binary classification output, c is 1 if it classifies the IoT device as compliant and c is 0 if it classifies the IoT device as non-compliant. Since the self-monitoring module wakes up when an event occurs (including a timer event), the monitoring module collects the value of c on an event by event basis. When an event occurs, the self-monitoring module collects event data in the form of (y_1, y_2, …, y_p, b) as input to the classifier and generates an output $c = 1$ if the IoT device is compliant or $c = 0$ if the IoT device is non-compliant. Here y_1, y_2, …, y_p are p feature variables as identified by the system designer in the design phase and observed by the monitoring module in the testing phase which altogether defines the status of the device and b is the behavior/action exhibited by the device when the particular event occurs.

Let c_i be the value of c collected in the ith event interval. As a result, the self-monitoring module collects a compliance degree history $(c_1, c_2, c_3, ...c_n)$ of the IoT device (being monitoring on) over n events, after which the monitor module runs a statistical analysis (to be described) to identify if the IoT device is malicious or not. Here n is a model parameter that will be determined by the system designer as the size of the testing dataset for assessing the effectiveness of the self-monitoring module (to be discussed in Section 7).

5. Methodology

In this Section, we introduce two statistical analysis techniques for (1) estimating the false negative probability and false positive probability and (2) quantifying the learning capability of AI-based misbehavior detection code. Fig. 5 shows a high-level overview of how these techniques are applied within the self-monitoring module. These techniques are probabilistic models that help the designer predict how the model will improve during training and when the training levels off.

5.1. Misbehavior detection

The compliance degree of an IoT device, monitored by the self-monitoring module using AI-based detection code, is modeled by a random variable X with $G(\cdot) = Beta(\alpha, \beta)$ distribution. In this model, a value of 0 signifies complete non-compliance, while a value of 1 represents full compliance [37,38]. The distribution $G(a)$, for $0 \geq a \geq 1$, is defined as:

$$G(a) = \int_0^a \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}\, dx \tag{1}$$

and the expected value of X is given by

$$E[X] = \int_0^1 x\, \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}\, dx = \frac{\alpha}{\alpha+\beta} \tag{2}$$

Using the method of maximum likelihood, the α and β parameters are determined from the compliance degree history $(c_1, c_2, c_3, ...c_n)$ through numerical solutions of the following equations.

$$\frac{n\frac{\partial\Gamma(\hat{\alpha}+\hat{\beta})}{\partial\hat{\alpha}}}{\Gamma(\hat{\alpha}+\hat{\beta})} - \frac{n\frac{\partial\Gamma(\hat{\alpha})}{\partial\hat{\alpha}}}{\Gamma(\hat{\alpha})} + \sum_{i=1}^{n} \log c_i = 0$$

$$\frac{n\frac{\partial\Gamma(\hat{\alpha}+\hat{\beta})}{\partial\hat{\beta}}}{\Gamma(\hat{\alpha}+\hat{\beta})} - \frac{n\frac{\partial\Gamma(\hat{\beta})}{\partial\hat{\beta}}}{\Gamma(\hat{\alpha})} + \sum_{i=1}^{n} \log(1-c_i) = 0 \tag{3}$$

where

$$\frac{\partial\Gamma(\hat{\alpha}+\hat{\beta})}{\partial\hat{\alpha}} = \int_0^\infty (\log x) x^{\hat{\alpha}+\hat{\beta}-1} e^{-x} dx.$$

This scenario reveals that the run-time complexity is O(nlogn), which is acceptable for modern IoT devices. For highly resource-limited IoT devices, like sensors, we can model the compliance degree using a random variable X with a single-parameter $Beta(\beta)$ distribution, setting α to 1. The density function is $\beta(1-x)^{\beta-1}$ for $0 \leq x \leq 1$ and zero otherwise. The maximum likelihood estimate for β can be calculated as:

$$\hat{\beta} = \frac{n}{\sum_{i=1}^{n} \log(\frac{1}{1-c_i})} \tag{4}$$

In this case, the observed run-time complexity is O(n).

The method's effectiveness can be assessed through false negative and false positive probabilities, represented as p_{fn} and p_{fp}. We use a simple threshold-based method: if the compliance degree of a malfunctioning IoT device X_b (following a one-parameter $G(\cdot) = Beta(\beta)$ distribution) is above the system's minimum compliance threshold C_T, a false negative occurs:

$$p_{fn} = pr\{X_b > C_T\} = 1 - G(C_T) \tag{5}$$

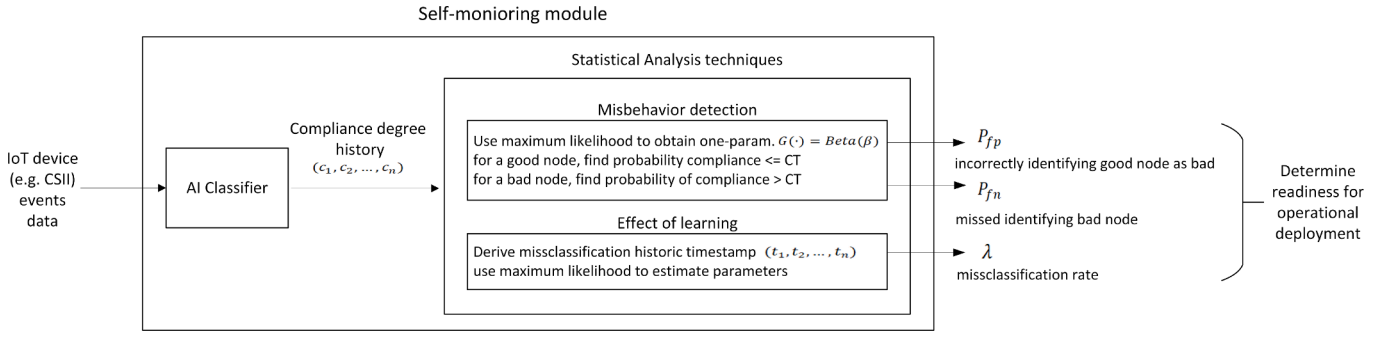


Fig. 5. The self-monitoring module inside the IoT device.

On the other hand, if the compliance degree of a correctly operating IoT device X_g (following a $G(\cdot) = Beta(\beta)$ distribution) is below or equal to C_T a false positive occurs:

$$p_{fp} = pr\{X_g \leq C_T\} = G(C_T) \tag{6}$$

During the testing phase for the purpose of estimating the self-monitoring module's p_{fn} and p_{fp}, we simulate a "bad" IoT device which exhibits bad behavior and a "good" IoT device which exhibits good behavior in response to events occurring in the system. We then allow the AI-based detection code to run through n monitoring intervals and collect the "bad" IoT device's compliance degree history $(c_1, c_2, c_3, \dots..c_n)$ and the "good" IoT device's compliance degree history $(c_1, c_2, c_3, \dots..c_n)$ with which we obtain the one-parameter $G(\cdot) = Beta(\beta)$ distributions for the "bad" IoT device and the "good" IoT device based on Eq. (4), and then compute p_{fn} and p_{fp} based on Eqs. (5) and (6), respectively.

Here we note that the statistical analysis technique discussed above is proposed to be run at the testing phase for the purpose of estimating the self-monitoring module's effectiveness in terms of p_{fn} and p_{fp}. Once p_{fn} and p_{fp} calculated satisfy the effectiveness requirement, the AI-based detection code is released to operational use and will be run at runtime for misbehavior detection. Specifically, given an IoT device (not knowing whether it is "bad" or "good" of course), the self-monitoring module running the AI-based detection code follows the same procedure as follows: it first collects the compliance degree history $(c_1, c_2, c_3, \dots..c_n)$; then it obtains the one-parameter $G(\cdot) = Beta(\beta)$ distribution for this IoT device; and finally it concludes that this IoT device is good with probability $1 - G(C_T)$ (since the IoT device's compliance degree is greater than C_T) and bad with probability $G(C_T)$ (since the compliance degree is less than C_T).

5.2. Effect of learning

In this work, we let the self-monitoring module run RF-based misbehavior detection code which is capable of learning so that the self-monitoring module can reduce the rate at which false positives or false negatives occur (typically due to misclassifications) as it learns from experiences (e.g., by supervised training during the testing phase). To quantify the impact of learning, let the misclassification rate, λ, be a function that decreases monotonically over time t, thereby illustrating that the number of misclassification outputs in the self-monitoring module decreases over time. In AI systems with learning capabilities, it is typically observed that the initial phase of learning tends to be more effective, with the rate of improvement slowing as the system continues to operate. A feasible solution to model λ with this feature is by using the logarithmic Poisson model [39,40] whereby it is assumed that the misclassification output arrival rate has a rapid decrease in the beginning and then smooths out with time. It can be obtained by

$$\lambda = \frac{\lambda_0}{\lambda_0 \theta t + 1} \tag{7}$$

where the model parameters are λ_0 and θ , referred to as the initial misclassification output arrival rate and the improvement parameter, respectively. In our case, these parameters can be estimated using the method of maximum likelihood by analyzing the number of misclassifications in p subintervals during the testing phase. The testing phase $(0, x_p]$ is partitioned into p disjoint subintervals where x is the event number, and let $y_l (l = 1, 2, ..., p)$ be the number of misclassifications in subinterval $(0, x_l]$. We find the number of misclassifications per interval based on the misclassification historic timestamp data $(t_1, t_2, ..., t_N)$ when the AI-based detection code generates a misclassification error. In this context, t_i represents the time at which the ith misclassification output is recorded, and N signifies the total number of misclassification outputs occurring within the p subintervals during the testing phase.

The maximum likelihood estimates of λ_0 (called $\hat{\lambda_0}$) and θ (called $\hat{\theta}$) can be obtained by finding the maximum likelihood estimate of the product $\phi = \lambda_0 \theta$ and solving the following equations [40]:

$$\frac{\partial L}{\partial \phi} = \sum_{i=1}^{p} y_i \, ' \left\{ \frac{\frac{x_i}{\phi x_i + 1} - \frac{x_{i-1}}{\phi x_i + 1}}{ln(\phi x_i + 1) - ln(\phi x_{i-1} + 1)} \right\} - \frac{y_p x_p}{(\phi x_p + 1) ln(\phi x_p + 1)} = 0 \tag{8}$$

$$\hat{\theta} = \frac{1}{y_p} ln(\hat{\phi} x_p + 1) \tag{9}$$

$$\hat{\lambda_0} = \frac{\hat{\phi}}{\hat{\theta}} \tag{10}$$

Here we again note that the self-monitoring module is to be run in the testing phase for misbehavior detection against the behaviors of the simulated "bad" and "good" IoT devices, so we know whether the self-monitoring module incurs a false negative or false positive on an event by event basis.

Therefore, according to Eq. (7), the misclassification output arrival rate λ is given by

$$\lambda = \frac{\hat{\lambda_0}}{\hat{\lambda_0} \hat{\theta} t_t + 1} \tag{11}$$

where t_t is the time for n event arrivals. As can be seen, when the system spends more time learning from examples in the testing phase, the rate of misclassification decreases over time but eventually the misclassification rate levels off, at which time the system is ready to be released for operational use. A lower misclassification rate leads to a lower false negative rate and a lower false positive rate.

6. Preparation of test datasets

This section describes the preparation of test datasets for testing the performance of the self-monitor misbehavior detection module. We have used Ulna [9] which is a development platform for artificial pancreas. The data is modeled from the real data of 15 patients collected by Haidar et al[41] and is clinically validated. Ordinary differential equations (ODE) were used to model the real data and generate the data used in our work. The model that generates our data was calibrated on real data and validated by the Ulna platform. We have used the data from

this system due to the unavailability of real data and the ability to simulate various types of attacks on the CSII device without jeopardizing the lives of the patients. We have used this system to generate the training data needed to create the misbehavior detection AI models and to create the attacks test data. A dataset contains 5-attribute records, as follows:

- Glucose level: The patient's glucose level.
- Insulin infusion: The total amount of insulin given to a patient.
- Time period: the time period of the day where the reading is from. The 24 h day was divided into six periods.
- Glucose slope: The gradient for the glucose level of the patient. The gradient is computed from the level of the previous hour.
- Compliance level: A binary target variable that indicates the compliance level of the CSII device in a monitoring interval.

We prepare a test dataset by post-processing data generated by Ulna which samples the data for the glucose level of type 1 diabetes patients based on user-specified protocols, including the trial duration (e.g., 60 days), mealtimes (e.g., three meals with carbs ranging between 30 – 65 grams and three snacks ranging between 15 - 25 grams), and onboard insulin levels. The platform generates CSII data for a set of actual patients and creates simulated data based on their glucose levels. A user devises a protocol specifying the behavior of a CSII device, i.e., the amount of insulin infused based on the glucose level (measured in mmol/L) and other factors. Eventually, we instruct Ulna to generate three simulated datasets, as follows:

1. Compliant CSII device dataset: This simulated dataset represents a compliant CSII device whose behavior is as specified in Table 2.
2. Reckless CSII device dataset: This simulated dataset represents a reckless-attack CSII device whose behavior is as specified in Table 3.
3. Insidious CSII device dataset: This simulated dataset represents an insidious-attack CSII device.

The sampled dataset generated by the Ulna platform contains a patient's glucose level and the insulin infusion of the device based on the behavior rules specified. The data records are in 10 m intervals starting at 7am for 60 days, resulting in a total of 8,908 records for each patient. We then post-process the simulated datasets generated by Ulna to extract the five-attribute records as described at the beginning of the section, as follows:

- Glucose level and insulin infusion: Taken directly from the Ulna dataset.
- Time period: Taken from the time of day.
- Glucose slope: Computed by the difference between the current glucose level and the glucose level from one hour ago. This cannot be computed for the first six records of the glucose level, hence they are removed from the dataset.
- Compliance degree: If the Ulna dataset is the compliant CSII device test dataset, it will be set to one. Otherwise, it will be set to zero.

Following the post-processing steps described above, the following five test datasets are generated:

1. Compliant CSII device dataset: created by post-processing Ulna-generated compliant dataset directly.
2. Reckless attack CSII device dataset: created by post-processing Ulna-generated reckless dataset directly.
3. Random attack CSII device dataset: Random attacks are simulated by combining the compliant dataset with the reckless dataset by a percentage depending on the probability of attack (P_a). For example, if P_a is 10%, the Random attack dataset will consist of 90% from the compliant dataset and 10% from the reckless dataset.
4. Insidious attack CSII device dataset: created by post-processing Ulna-generated insidious dataset directly.
5. Insidious nocturnal attack CSII device dataset: Attacks are simulated by combining the compliant dataset for the time periods 1-4 [4 am – 8 pm] with the insidious dataset for the periods 0 [12 am – 4 am] and 5 [8 pm – 12 pm].

At the end, each dataset contains 71,184 5-attribute records. We simulate noises (with respect to a data attribute) as a random variable with normal distribution with a mean (μ) of zero except for the time period attribute. To simulate multiple levels of noises, we increase the standard deviation (σ^2) values corresponding to the level of required noise. Specifically, low noises have a σ^2 of 5, medium noises have a σ^2 of 7, and high noises have a σ^2 of 9.

7. Evaluation results

7.1. Model analysis and validation

Without loss of generality, we first apply RF [5] as the underlying AI technique used by the self-monitoring module to classify if a target CSII device is misbehaved, as RF is known to have a great learning ability to improve its classification accuracy. We will show that other AI classification methods with learning capability (e.g., Artificial Neural Network (ANN), k-Nearest Neighbors (KNN), and Gradient Boosting (GB)) can also be similarly applied with results generating similar trends. Our experimental environment setup follows the behavior rules in Table 2 with 15 event types continuously occurring in the testing phase. We divide the first 2,000 data records from the dataset into 10 training sets with 200 data records each. The reason that we separate the training data into 10 training sets is to analyze the effect of classifier's learning capabilities on the performance of misbehavior detection. That is, we train the classifier 10 times, each with combining the training sets and then measure p_{fn} and p_{fp} to see the effect of learning on performance.

To measure p_{fn} and p_{fp} we consider two target CSII devices, one "good" and one "bad," whose actions toward an event follow Tables 2 and 3 accordingly. These are sampled using the datasets created in section 6. Essentially when an event occurs, the monitoring module collects event data along with the action taken by the target CSII as input with which the monitoring module classifies if the target CSII as "well-behaved" (in which case a compliance degree of 1 is assigned) or "misbehaved" (in which case a compliance degree of 0 is assigned). Of course, if the classifier is perfect, it will always classify the "bad" target CSII as "non-compliant" and the "good" target CSII as "compliant". However, like most AI programs, the classifier despite producing correct outputs mostly, sometimes they can severely fail. After the classifier is trained with each training set of 200 data records, we let it run through the testing dataset of 14,000 data records (so $n = 14{,}000$) and collect the "bad" IoT device's compliance degree history $(c_1, c_2, c_3, ...c_n)$ and the "good" IoT device's compliance degree history $(c_1, c_2, c_3, ...c_n)$ so as to obtain the one-parameter $G(\cdot) = Beta(\beta)$ distributions for the "bad" IoT device and the "good" IoT device based on Eq. (4), and to compute p_{fn} and p_{fp} based on Eqs. (5) and (6) respectively. As the classifier goes through the testing dataset of 14,000 data records, we also collect misclassification timestamp history $(t_1, t_2, ..., t_N)$ at which the classifier makes a misclassification error. The misclassification timestamp history is used to obtain the number of misclassifications y for all p subintervals. This allows us to compute the misclassification output arrival rate λ after the classifier is trained in the current learning testing cycle based on Eq. (11) (with t_t being the total time of $n = 14{,}000$ event arrivals). The procedure above is repeated 5 times each of which the classifier is trained with 200 events and tested with 14,000 events.

Table 4 shows the results. There are 5 rows each representing one cycle of the learning testing procedure. There are three columns showing λ (the arrival rate of misclassification outputs), p_{fn} (false negative probability) and p_{fp} (false positive probability) at the end of a learning testing cycle. We observe that λ reduces as time progresses as the classifier learns from examples. We note that p_{fn} and p_{fp} also decrease as time progresses because the classifier makes fewer misclassification errors as it sees more examples. In particular, the improvement param-

Table 4
Effectiveness of self-monitoring module running RF with no noise measured by λ, p_{fn}, and p_{fp}. Note: C_T is set at 0.75.

Learning testing Cycle	λ	p_{fn}	p_{fp}
1st	0.004	0.1772	0.1549
3rd	0.003	0.06	0.1497
5th	0.0024	0.0477	0.1474
7th	0.0017	0.0227	0.1465
9th	0.0014	0.0111	0.1455

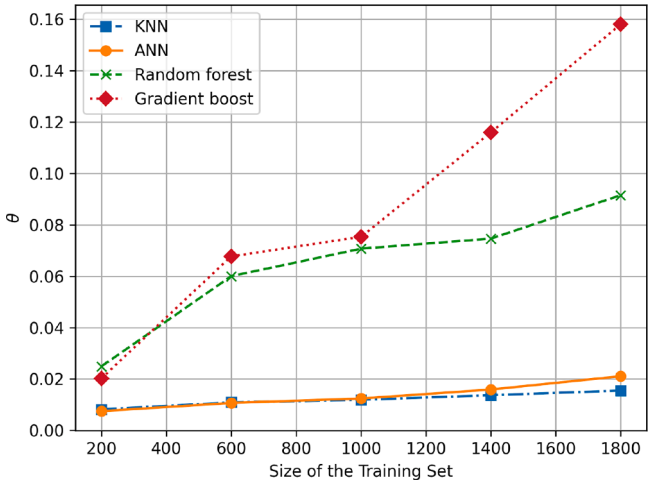


Fig. 6. Effect of training dataset size on θ.

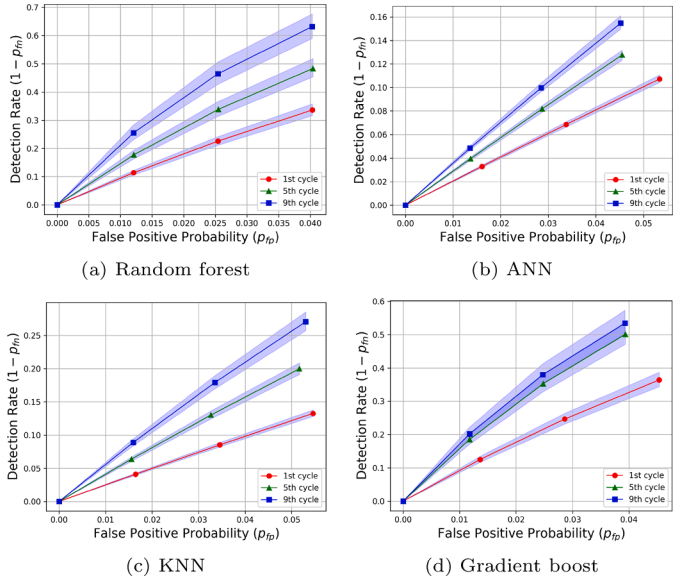


Fig. 7. AUROC curves over 3 learning testing cycles for AI-based misbehavior detection on the CSII Device with a 95% confidence interval.

eter θ impacts the extent to which learning enhances the system reliability. A more rapidly reducing λ at the beginning as a function of t is a result of a larger θ. This is shown in Fig. 6. The method discussed here allows one to estimate the value of θ for a system with learning capability.

Fig. 7 shows the *Area Under Receiver Operating Characteristic* (AUROC) curves for four AI-based misbehavior detection methods. The detection rate ($1 - p_{fn}$) on the Y coordinate and the false positive rate (p_{fp}) on the X coordinate. AUROC is commonly employed as a performance metric to evaluate misbehavior detection methods because it accurately portrays the tradeoff between p_{fn} and p_{fp} by adjusting the minimum compliance degree threshold C_T such that a higher C_T (i.e., a higher bar for good behavior) decreases p_{fn} but increases p_{fp}, and vise versa. There are three AUROC curves shown in the three subfigures in Fig. 7, each curve is obtained after the specified training-testing cycle. We can see that the AUROC improves as the training-testing cycles increase. The blue shaded area around each curve represents the 95 % confidence interval for each curve. This confidence interval was calculated by the bootstrapping resampling method. The compliance degree history was sampled with replacement 1,000 times, and for each resample, the AUROC is computed. The AUROC scores form a distribution that allows for the estimation of the 95 % confidence interval by determining the appropriate percentiles (2.5th and 97.5th). The confidence intervals for each curve estimate its variability and show that the AUROC curves differ and improve as the number of cycles increase. This is the case for the four AI-based methods in Fig 7(a)–(d).

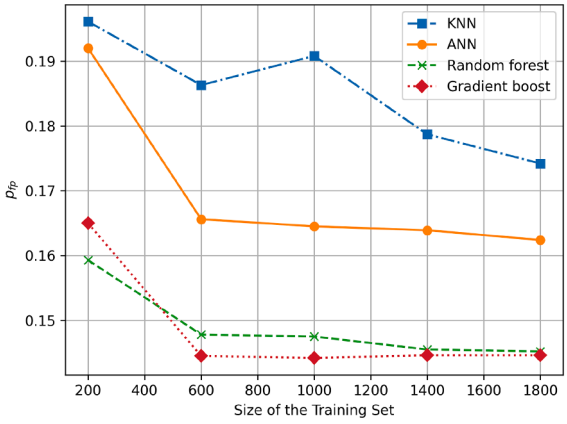
Fig. 8 shows the effect of the training dataset size on the improvement of p_{fp} and p_{fn} for the self-monitoring module running the RF classifier. The X-axis in the three figures represents the size of the training set and the Y-axis represents the value of p_{fp} or p_{fn}. Here we increase the number of training datasets to beyond 5 in order to show the effect (similarly for subsequent tests where applicable). In Fig. 8(a), we see the effect of the increase of the training set on p_{fp} when the AI-based misbehavior detection code is developed using different ML methods, including KNN, ANN, RF, and GB. We observe that for all schemes p_{fp} monotonically decreases as the size of the training set increases. This demonstrates that our methodology for building the AI-based detection code is generally applicable regardless of the AI method used as the classification scheme for misbehavior detection. Fig. 8(b) shows the effect of the size of the training set on p_{fn} under various attack scenarios. We observe that p_{fn} monotonically decreases as there is increase in the size of the training set only when the type of attack is a reckless attack. However, for other types of attacks increasing the training size does not improve in detecting nodes using these types of attacks. Also, from the figure we see the difference in p_{fn} among the various attack scenarios. We observe that insidious nocturnal attacks have the highest p_{fn} since they are the hardest to detect, followed by the insidious critical attacks, random attacks, and finally reckless attacks. Finally, Fig. 8(c) focuses only on the RF classifier from Fig. 8(a) which clearly shows the decrease in p_{fp} as the size of the training increases.

Fig. 9 shows the effect of the training size on the RF-based misbehavior detection method's various popular AI metrics such as accuracy, recall, precision, and the f_1 score. As the training dataset increases, the method's detection performance increases across all AI metrics.

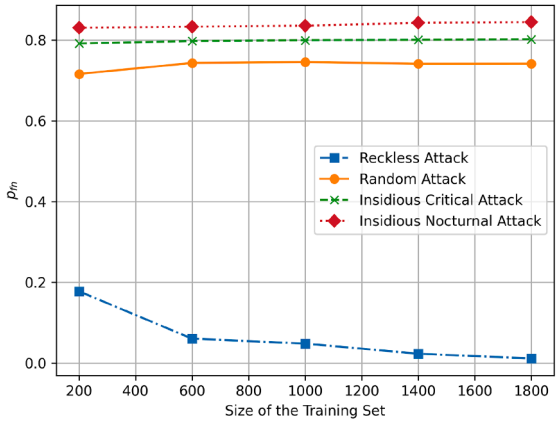
Fig. 10 shows the effect of training on λ under the various AI methods and various attack scenarios. Fig. 10(a) shows that the λ decreases for the different AI methods as the training increases. The arrival rate of the misclassification decreases as the AI methods is trained on more examples and becomes more accurate. Fig. 10(b) shows that the arrival rate of misclassifications among the various attacks is analogous to the order of values of p_{fn} of the attacks, as seen in Fig. 8(b), in the first training iterations. However, as the iterations increase the λ drops for all attacks to the point where they are very similar. This shows that increasing the training on the RF classifier will allow it to handle all types of misbehavior regardless of the type of attack.

Fig. 11 shows the AUROC curves under various attack scenarios. We observe that reckless attacks have the largest area under the curve while insidious nocturnal attacks have the smallest. This order correctly corresponds to the level of difficulty in detecting these attacks. Reckless attacks are the easiest to detect with the insidious nocturnal being the hardest to detect.

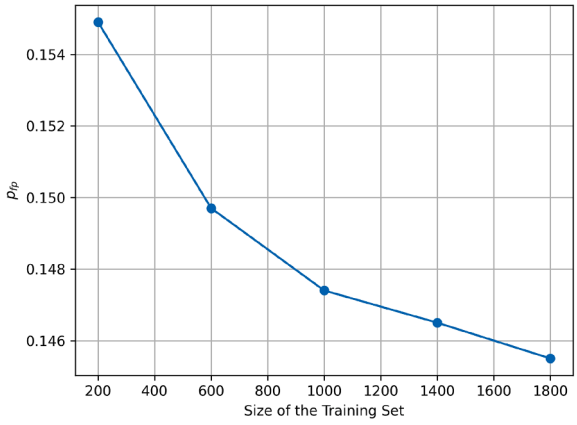
Fig. 12 shows the AUROC curves under random attacks with the attack probability in the range of [0.1, 0.3]. We observe that random attacks with the highest attack probability ($P_a = 0.5$) have the largest area under the curve while random attacks with the lowest probability of attack ($P_a = 0.2$) have the smallest area . It correctly reflects the trend



(a) Effect of training dataset size on p_{fn} under various AI classification methods.



(b) Effect of training dataset size on p_{fn} under various attack scenarios.



(c) Effect of training dataset size on p_{fp} on RF model.

Fig. 8. Effect of training dataset size on p_{fn} and p_{fp}.

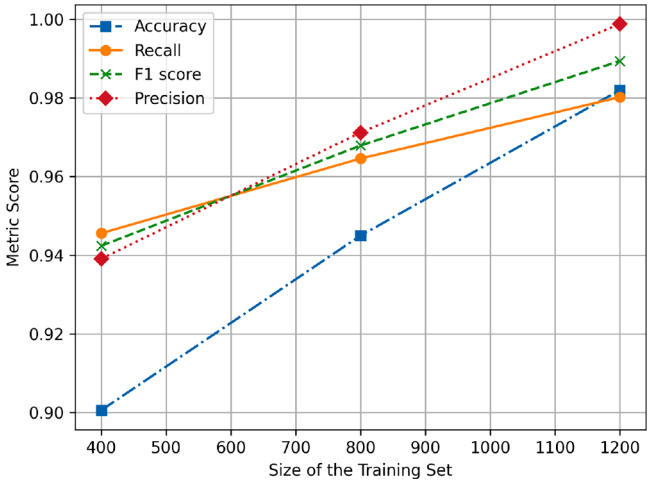
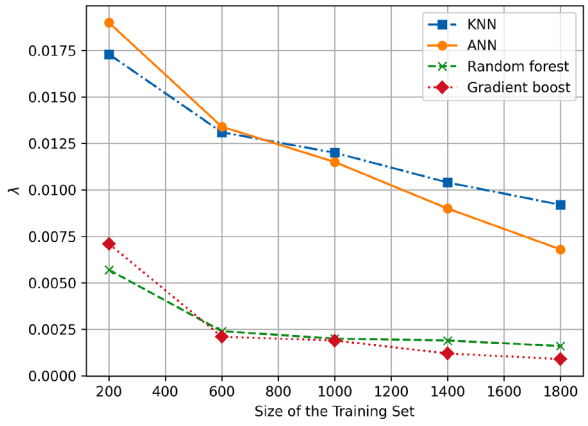
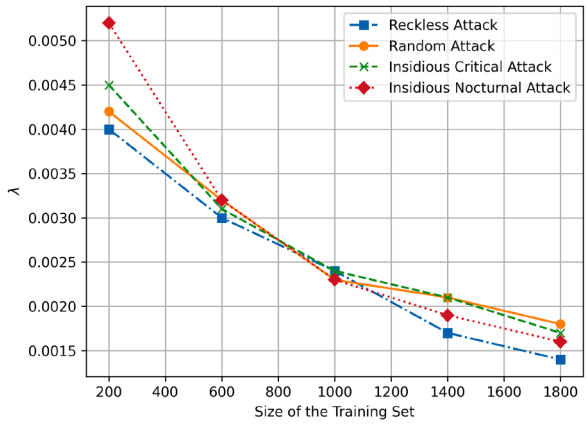


Fig. 9. Effect of training dataset size on AI metrics for a Random forest-based misbehavior detection method on insidious nocturnal dataset.



(a) Effect of training dataset size on λ under various AI classification methods.



(b) Effect of training dataset size on λ under various attack scenarios.

Fig. 10. Effect of training dataset size on λ.

that as the probability of random attacks increases, it is easier to detect random attacks.

Finally, Fig. 13 shows the AUROC curves under reckless attacks with varying noise levels. We observe that as the noise level increases, the probability of monitoring error also increases and consequently, the detection rate also decreases. This also correctly reflects the trend that in highly noisy environments, the performance of misbehavior detection will naturally degrade.

7.2. Comparative analysis

We compare our proposed methodology against baselines representative of techniques used in recent works. These include deterministic, mean-based thresholds for making a binary decision on a node's status (e.g., compliant or non-compliant), a common approach in the literature [31–33]. Furthermore, we compare against Early Stopping, a

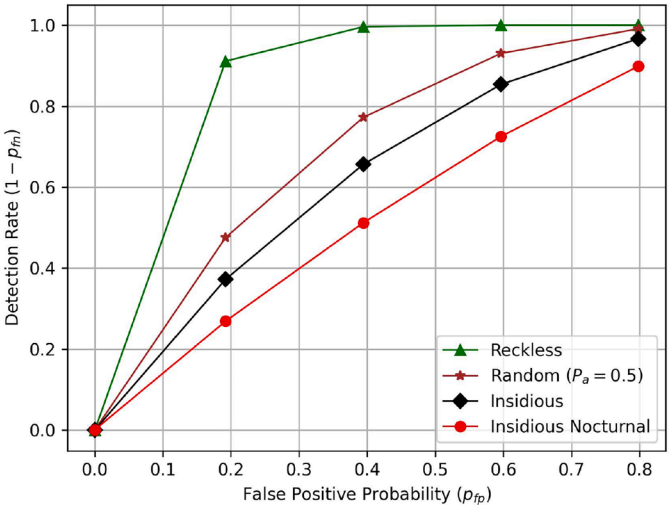


Fig. 11. AUROC curves under various attack scenarios.

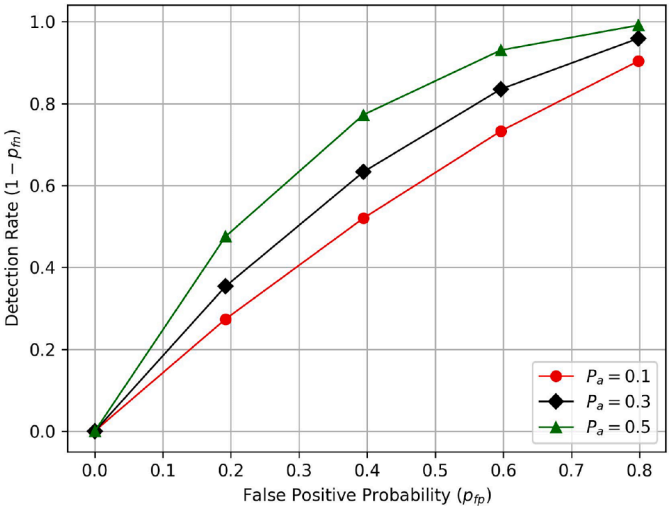


Fig. 12. AUROC curves under random attacks with the attack probability p_a in the range of [0.1, 0.5].

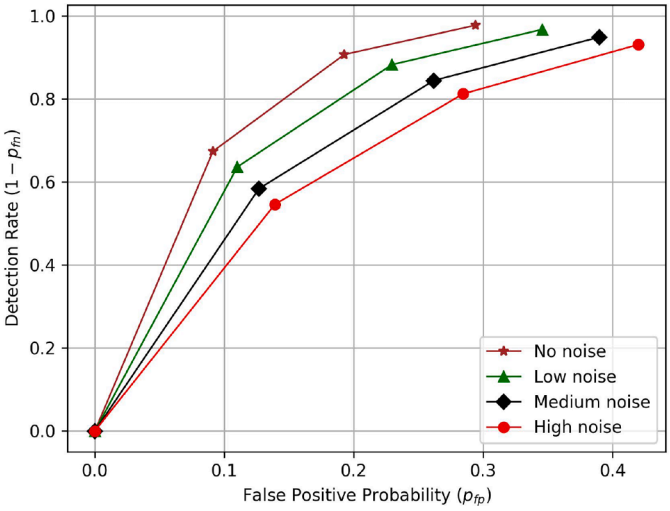


Fig. 13. AUROC curves under different noise levels.

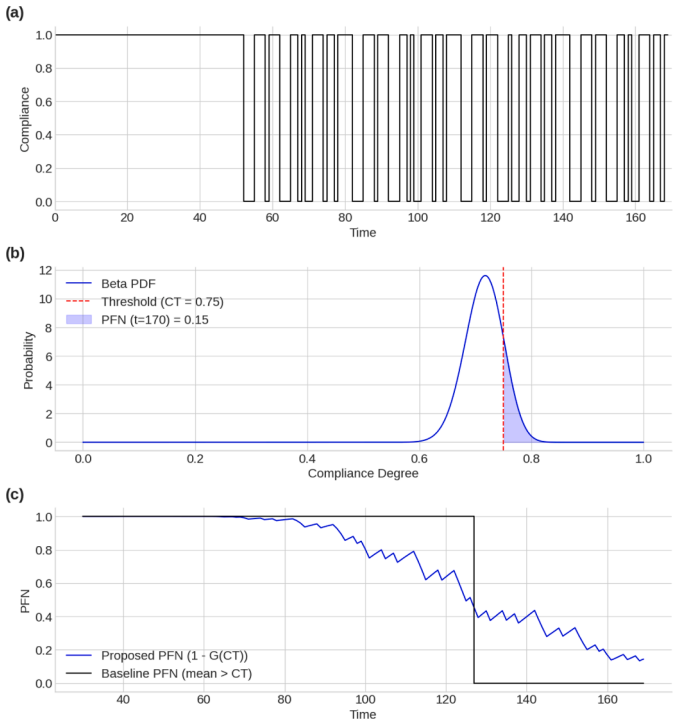


Fig. 14. Illustration of the proposed P_{fn} methodology showing (a) the observed compliance degree history over 170 timepoints; (b) the predicted likelihood of future compliance, calculated from the full history at t = 170; and (c) the evolution of the P_{fn} over time, comparing the proposed probabilistic method to a mean-based baseline.

standard method for determining when to halt model training [34,35]. The following analysis demonstrates how our predictive, probabilistic model provides more robust and actionable insights than these established techniques. We compare our proposed methodology for P_{fn} with that of a mean-based baseline. The distinction between the two methods is evident when analyzing the system's state at time $t = 100$, as shown in Fig. 14. At this point, the baseline approach concludes a P_{fn} of 1, as the cumulative mean compliance, $\bar{c}$, i.e., the False Negative Rate (FNR), remains greater than the threshold, C_T. This assessment, however, only indicates that a threshold has not yet been crossed and provides no insight into the magnitude of the risk. In contrast, the proposed methodology, by fitting the historical data to a Beta distribution, $G(\cdot)$, yields a quantitative probability. At $t = 100$, it calculates a $P_{fn} \approx 0.8$. This value represents the probability $Pr\{c > C_T\}$, quantifying the likelihood that the device's compliance degree, c, will exceed the threshold C_T. This probabilistic forecast, derived from modeling the system's behavior with the distribution $G(\cdot)$, allows making risk-informed decisions based on a quantified probability of failure rather than a simple determination of whether a threshold has been crossed.

The proposed model characterizes uncertain and dynamic device behavior, whether due to benign reasons or because of deceptive compliance patterns. It employs an adaptive probabilistic approach, using a Beta distribution to quantify the uncertainty of the true compliance probability. This model of compliance is dynamically shaped by an evolving history of discrete events. The resulting probabilistic forecast of the Probability of False Negative (P_{fn}) offers a significant advantage over deterministic, mean-based thresholds. It enables earlier detection of behavioral degradation and facilitates more robust, risk-informed deployment decisions.

Fig. 15 highlights the forecasting advantage of our methodology for a steadily improving system that has not yet met its performance target. This is contrasted with Early Stopping, a common technique that halts training if a validation metric fails to improve. For this experiment, we

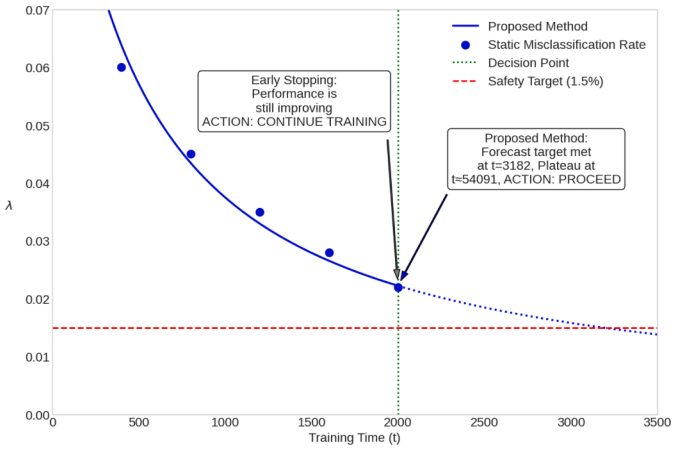


Fig. 15. Decision-making comparison with Early Stopping.

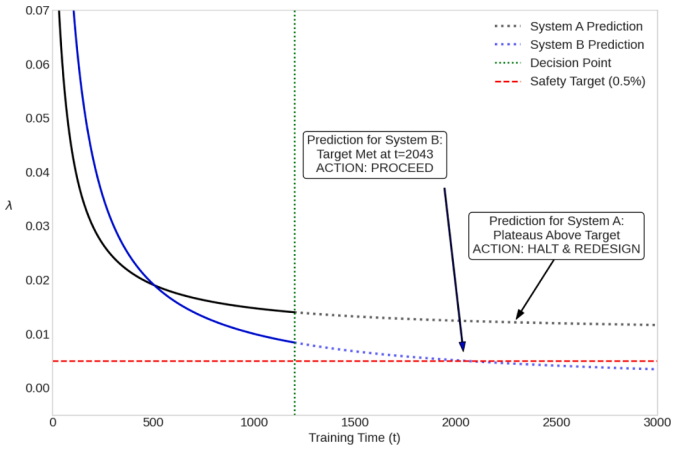


Fig. 16. Comparative analysis of two AI systems.

set a `patience` of $p = 1$ interval and a minimum improvement `delta` of $\delta = 0.001$. The stopping condition is therefore triggered if the current misclassification rate, $\lambda_{val}(t)$, is not less than the best-seen rate, λ_{best}, by at least δ (i.e., if $\lambda_{val}(t) \geq \lambda_{best} - \delta$). At the decision point of $t = 2000$, since performance is still consistently improving, this condition is not met and the rule provides no strategic guidance other than to "CONTINUE TRAINING." It offers no insight into when, or if, the goal will be reached. In contrast, our proposed methodology provides a clear, actionable forecast. It predicts the specific time at which the safety target will be met ($t = 3182$) and can also forecast the long-term practical plateau ($t \approx 54,091$), enabling proactive project planning and resource management.

Fig. 16 illustrates the strategic utility of our proposed methodology by analyzing two AI systems with different misclassification behaviors at a decision point of $t = 1200$. System A (black line) represents an underperforming model whose misclassification events cease after $t = 1060$, indicating that its learning has plateaued. System B (blue line) represents a successfully learning model whose misclassification events are initially frequent but become progressively sparser over time. Based on these distinct historical patterns, our methodology generates two different forecasts at the decision point. The forecast for System A shows its performance curve plateauing above the 1.5% safety target, providing a clear, early warning signal to HALT development and redesign the system. In contrast, the forecast for System B shows a clear trajectory to meeting the safety target, enabling a confident decision to PROCEED and allocate resources efficiently.

8. Discussion and limitations

Our experimental results demonstrate that the proposed statistical methodology provides a robust framework for evaluating the operational readiness of AI-based misbehavior detectors in critical IoT systems. The analysis in Section 7 confirms two primary benefits: first, the use of a Beta distribution to model the compliance history yields a continuous, probabilistic measure that is more informative than a discrete, deterministic threshold. Second, modeling the learning effect with a logarithmic Poisson process enables a predictive forecast of the misclassification rate, λ, offering actionable insights where traditional training techniques fall short.

When compared to related research, the advantages of our approach become evident. The mean-based thresholding techniques, representative of methods used by Rahman et al. [31], Lopez-Martin et al. [32], and Sohi et al. [33], provide a reactive assessment of a device's state. As shown in Fig. 14, such baselines can only determine if a cumulative average has crossed a threshold, whereas our probabilistic method quantifies the underlying risk ($P_{fn} \approx 0.8$ at $t = 100$) even when the mean behavior appears acceptable. This provides a crucial early warning of degrading performance. Similarly, our work complements advanced training optimization techniques like those proposed by Paguada et al. [34] and Chen et al. [35]. While their methods improve training efficiency, our work provides a framework for determining *operational readiness* based on a forecast of when performance will stabilize or meet a safety target, a crucial step for deploying AI in safety-critical systems like the CSII device. Unlike the majority of IDS research that focuses on final classification accuracy on benchmark datasets [10,12,14], our work shifts the focus from static performance metrics to a dynamic analysis of the learning process, which is essential for the assurance of trustworthy AI in medical systems.

However, our study has several limitations that open avenues for future research. First, the evaluation was conducted on a high-fidelity simulation of a CSII device. While the data is modeled from real patient behavior, validation on physical, resource-constrained IoT hardware in a clinical or near-clinical setting is a necessary next step to confirm the real-world efficacy of the framework. Second, the methodology's validity rests on the assumption that the stochastic nature of device compliance and learning dynamics can be effectively captured by parametric statistical distributions. While these are well-suited for the observed behaviors, future work could explore more complex time-series models [34] or other statistical distributions and uncertainty modeling techniques [11,22]. Finally, while our analysis included several common AI classifiers, the framework should be tested against a wider range of architectures, such as transformer-based models, to assess its generalizability. Future work will focus on deploying this framework on embedded hardware and extending the statistical models to support systems that utilize continuous online learning in operational environments.

The chosen AI methods may also introduce biases. RF can be biased by its feature splitting mechanism, which can lead to a higher p_{fn} if misbehavior data is sparse and underrepresented in feature subsets, as observed in our tests with insidious attacks. Similarly, KNN's distance-based nature makes it susceptible to noisy data, while ANN and GB can underfit and introduce bias if their architectures are not sufficiently complex. Such potential biases can be mitigated by various methods, including robust data preprocessing, relabeling, and applying regularization during the training process [42].

Furthermore, a high-noise data environment can affect model performance. While we explored this effect by simulating multiple noise levels, practical mitigation strategies would be important for real-world deployments. Proven solutions include data augmentation to enhance input quality [43], using prediction models to replace erroneous sensor values [44], or applying noise reduction methodologies such as input averaging [45] and clustering-based noise elimination [46].

9. Conclusions

This paper addressed the critical challenge of ensuring the reliability of learning-enabled AI misbehavior detectors in safety-critical IoT systems. We introduced a novel statistical framework that shifts the evaluation paradigm from a reactive, static assessment to a proactive, predictive forecast. Our methodology empowers system designers to quantify the learning process, predict future performance, and make informed, data-driven decisions about when a system is operationally ready for deployment.

Through a comprehensive evaluation on a smart CSII medical device, we demonstrated that our framework not only effectively tracks improvements in detecting sophisticated attacks but, more importantly, provides superior, actionable insights compared to traditional thresholding and early-stopping techniques. Future work will focus on several key directions: using our framework to develop and validate more data-efficient training strategies; applying it to other diverse IoT domains like industrial or vehicular systems; extending its support for online learning models; and performing validation on physical, resource-constrained hardware.

CRediT authorship contribution statement

Hamid Al-Hamadi: Writing – review & editing, Writing – original draft, Project administration, Funding acquisition, Conceptualization; **Ing-Ray Chen:** Writing – original draft, Supervision, Conceptualization; **Ding-Chau Wang:** Writing – review & editing, Supervision; **Abdullah Almutairi:** Writing – review & editing, Visualization, Investigation.

Data availability

The data used in this paper is available at https://www.kaggle.com/datasets/almutairiphd/infuser-rules-raw-data/data.

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.

Acknowledgment

This work was supported and funded by the Kuwait University Research Grant under Grant No. RQ01/19. The authors would like to thank Mohamed Smaoui, Computer Science Department, Kuwait University, for providing the simulation platform for Artificial Pancreas algorithms and related support. The authors would also like to thank the Research Associate, Anita Philips for her technical contribution, coding, and editing support in this work.

References

[1] G. Choudhary, P.V. Astillo, I. You, K. Yim, I.R. Chen, J.H. Cho, Lightweight misbehavior detection management of embedded IoT devices in medical cyber physical systems, IEEE Trans. Netw. Serv. Manag. 17 (4) (2020) 2496–2510.

[2] V. Sharma, I. You, K. Yim, I.R. Chen, J.H. Cho, BRIoT: behavior rule specification-based misbehavior detection for IoT-embedded cyber-physical systems, IEEE Access 7 (1) (2019) 118556–118580.

[3] Y. Chen, W. Sun, N. Zhang, Q. Zheng, W. Lou, Y.T. Hou, Towards efficient fine-grained access control and trustworthy data processing for remote monitoring services in IoT, IEEE Trans. Inf. Forensics Secur. 14 (7) (2019) 1830–1842.

[4] V. Costan, S. Devadas, Intel SGX Explained, 2016, (IACR Cryptology ePrint Archive).

[5] T.K. Ho, Random decision forests, in: Proceedings of 3rd International Conference on Document Analysis and Recognition, 1995, pp. 278–282.

[6] S. Kapil, R. Saini, S. Wangnoo, S. Dhir, Artificial pancreas system for Type 1 diabetes–challenges and advancements, Explor. Res. Hypothesis Med. 000 (000) (2020) 1–11. https://doi.org/10.14218/erhm.2020.00028

[7] National Institute of Diabetes and Digestive and Kidney Diseases, Type 1 Diabetes - NIDDK, 2023. Accessed: Aug. 13, 2023, https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes/type-1-diabetes.

[8] A. Cinar, K. Turksoy, Advances in Artificial Pancreas Systems, Components of an Artificial Pancreas System, 2018. https://doi.org/10.1007/978-3-319-72245-0-2

[9] M.R. Smaoui, R. Rabasa-Lhoret, A. Haidar, Development platform for artificial pancreas algorithms, PLoS ONE 15 (12) (2020) e0243139. https://doi.org/10.1371/journal.pone.0243139

[10] M. Fouda, R. Ksantini, W. Elmedany, A Novel Intrusion Detection System for Internet of Healthcare Things Based on Deep Subclasses Dispersion Information, IEEE Internet Things J. (2023) 8395–8407. https://doi.org/10.1109/JIOT.2022.3230694

[11] X. Li, Y. Wang, M. Zhou, W. Li, Bayesian model uncertainty estimation for misbehavior detection in IoT, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS), ACM, 2022, pp. 167–180. https://doi.org/10.1145/3548606.3560597

[12] A.-A. Maiga, E. Ataro, S. Githinji, Intrusion detection with deep learning classifiers: a synergistic approach of probabilistic clustering and human expertise to reduce false alarms, IEEE Access 12 (2024) 17836–17858. https://doi.org/10.1109/ACCESS.2024.3359595

[13] R. Rocca, Integrating statistical methods and game theory for enhanced IoT intrusion detection, in: 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), 2025, pp. 1–4. https://doi.org/10.1109/CCNC54725.2025.10976112

[14] S. Rizvi, M. Scanlon, J. McGibney, J.W. Sheppard, An evaluation of AI-based network intrusion detection in resource-constrained environments, in: 2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2023, pp. 275–282. https://doi.org/10.1109/UEMCON59035.2023.10315971

[15] M. Jin, et al., Poster: towards real-time intrusion detection with explainable AI-Based Detector, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024. https://doi.org/10.1145/3658644.3691410

[16] M.M. Rahman, M. Nijim, H. Jeong, AI2DS: advanced deep autoencoder-driven method for secure network intrusion detection, in: 2025 IEEE International Conference on Cyber Security and Resilience (CSR), 2025.

[17] F. Wali, O. Shahid, W. Mahmood, M. Usama, AIS-NIDS: an intelligent and self-sustaining network intrusion detection system, in: 2024 IEEE International Conference on Smart Computing (SMARTCOMP), 2024.

[18] S.-P. Wu, D. Kondo, A high-throughput network intrusion detection system with on-device learning using adaptive feature compression, IEEE Trans. Dependable Secure Comput. (2024). https://doi.org/10.1109/TDSC.2024.3381345

[19] G.-Y. Zhao, L. Zhang, W.-T. Liu, A novel intrusion detection method based on lightweight adaptive loss and dimensionality reduction, IEEE Access 10 (2022) 78512–78524. https://doi.org/10.1109/ACCESS.2022.3192385

[20] S. Kaushik, A. Bhardwaj, A. Almogren, S. Bharany, A. Ur Rehman, S. Hussen, H. Hamam, Robust machine learning based Intrusion detection system using simple statistical techniques in feature selection, Scientific Reports 15 (1) (2025) 3970.

[21] S. Tewari, D.S. Rawat, Lightweight intrusion detection system for IoT networks using quantized CNN-BiLSTM architecture, IEEE Trans. Inf. Forensics Secur. (2024). https://doi.org/10.1109/TIFS.2024.3387351

[22] O. Simeone, S. Park, M. Zecchin, Conformal calibration: Ensuring the reliability of black-box ai in wireless systems, (2025) arXiv preprint arXiv:2504.09310.

[23] S. Kebir, K. Tabia, On handling concept drift, calibration and explainability in non-stationary environments and resources limited contexts, in: ICAART (2), 2024 336–346.

[24] Md. K. Hossain, I. Ahmad, D. Habibi, M. Waqas, Enhancing IoT sensors precision through sensor drift calibration with variational autoencoder, IEEE Internet Things J. 12 7 8421–8437 (2025) https://doi.org/10.1109/TIFS.2024.3387351.

[25] J. Winter, H. Kim, S.-H. Lim, An empirical study of uncertainty estimation techniques for drift detection in deep learning-based IDS, in: 2023 ACM Conference on Computer and Communications Security, 2023.

[26] S. Xiong, Q. Li, Z. Zhang, AIDTF: adversarial training framework for network intrusion detection systems, in: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, 2023.

[27] K. Sauka, G.-Y. Shin, D.-W. Kim, M.-M. Han, Adversarial robust and explainable network intrusion detection systems based on deep learning, Appl. Sci. 12 (13) (2022) https://www.mdpi.com/2076-3417/12/13/6451.

[28] T.-T. Nguyen, M. Tran, Q.-A. Ha, A robust and trustworthy intrusion detection system using GAN-based adversarial training and explainability, in: 2024 ACM Symposium on Information, Computer and Communications Security, 2024.

[29] T. Rashid, N. Basha, E. Ahmed, A deep learning-based semi-supervised network intrusion detection system With adversarial defense, in: 2023 IEEE Symposium on Security and Privacy (SP), 2023.

[30] H. Güngör, B. Aksanli, A2HD: adaptive adversarial training for hyperdimensional computing in IoT security, in: 2024 IEEE/ACM Design Automation and Test in Europe Conference (DATE), 2024.

[31] A. Rahman, A.H.M. Sarwar Chowdhury, A.K.M. Bahalul, M. Haque, N. Islam, Attack detection in industrial control systems using machine learning, IEEE Access 10 (2022) 45563–45579. https://doi.org/10.1109/ACCESS.2022.3169872

[32] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, An unsupervised method for anomaly detection in IoT networks based on connection patterns, IEEE Internet Things J. 10 (10) (2023) 8948–8959. https://doi.org/10.1109/JIOT.2023.3235651

[33] N. Sohi, P. Kaur, J. Singh, Real-time detection of IoT botnet attacks using a hierarchical threshold-based approach, Comput. Secur. 121 (2022) 102839. https://doi.org/10.1016/j.cose.2022.102839

[34] I. Paguada, J. Stewart, A. Lyle, H. Medeiros, Robust Early Stopping for Time Series Forecasting Models Using Predictive Uncertainty, IEEE Trans. Comput. 70 (12) (2021) 2152–2163. https://doi.org/10.1109/TC.2021.3064767

[35] W. Chen, Z. Wu, J. Xu, Y. Wang, Bayesian optimization based on pseudo labels, in: 2022 Asia Conference on Algorithms, Computing and Machine Learning (CACML), IEEE, 2022, pp. 217–222. https://doi.org/10.1109/CACML55074.2022.00043

[36] H. Al-Hamadi, I.-R. Chen, D.-C. Wang, M. Almashan, Attack and Defense Strategies for Intrusion Detection in Autonomous Distributed IoT Systems, IEEE Access (2020). https://doi.org/10.1109/ACCESS.2020.3023616

[37] R. Mitchell, I.-R. Chen, Behavior-rule based intrusion detection systems for safety critical smart grid applications, IEEE Trans. Smart Grid 4 (3) (2013) 762–770. https://doi.org/10.1109/TSG.2013.2258948

[38] R. Mitchell, I.-R. Chen, Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems, IEEE Trans. Rel. 62 (1) (2013) 1–9. https://doi.org/10.1109/TR.2013.2240891

[39] F.B. Bastani, I.R. Chen, T. Tsao, Reliability of systems with fuzzy-failure criterion, in: Annu. Rel. Maintainab. Symp., 1994, pp. 442–448.

[40] J.D. Musa, K. Okumoto, A logarithmic poisson execution time model for software reliability measurement, in: Proc. 7th Int. Conf. Softw. Eng., Orlando, FL, 1984, pp. 230–237.

[41] A. Haidar, et al., Glucose-responsive insulin and glucagon delivery (dual-hormone artificial pancreas) in adults with type 1 diabetes: a randomized crossover controlled trial, Cmaj 185 (4) (2013) 297–305.

[42] M. Hort, Z. Chen, J.M. Zhang, M. Harman, F. Sarro, Bias mitigation for machine learning classifiers: a comprehensive survey, ACM J. Responsible Comput. 1 (2) (2024) 1–52. https://doi.org/10.1145/3631326

[43] M. Momeny, et al., Learning-to-augment strategy using noisy and denoised data: Improving generalizability of deep CNN for the detection of COVID-19 in X-ray images, Comput. Biol. Med. 136 (2021). https://doi.org/10.1016/j.compbiomed.2021.104704

[44] O. Oleghe, A predictive noise correction methodology for manufacturing process datasets, J. Big Data 7 (1) (2020). https://doi.org/10.1186/s40537-020-00367-w

[45] S.J. Min, Y.S. Jo, S.J. Kang, Super-resolving methodology for noisy unpaired datasets, Sensors 22 (20) (2022). https://doi.org/10.3390/s22208003

[46] S.Z. Liu, R.S. Sinha, S.H. Hwang, Clustering-based noise elimination scheme for data pre-processing for deep learning classifier in fingerprint indoor positioning system, Sensors 21 (13) (2021). https://doi.org/10.3390/s21134349