Hybrid Knowledge and Data Driven Synthesis of Runtime Monitors for Cyber-Physical Systems

Xugui Zhou, *Student Member, IEEE,* Bulbul Ahmed, *Student Member, IEEE,* James H. Aylor, *Fellow, IEEE,* Philip Asare, *Member, IEEE,* and Homa Alemzadeh, *Member, IEEE*

Abstract—Recent advances in sensing and computing technology have led to the proliferation of Cyber-Physical Systems (CPS) in safety-critical domains. However, the increasing device complexity, shrinking technology sizes, and shorter time to market have resulted in significant challenges in ensuring the reliability, safety, and security of CPS. This paper presents a hybrid knowledge and data-driven approach for designing run-time context-aware safety monitors that can detect early signs of hazards and mitigate them in CPS. We propose a framework for formal specification of unsafe system context using Signal Temporal Logic (STL) combined with two optimization approaches for scenario-specific refinement and integration of STL specifications using data collected from closed-loop CPS simulations. We demonstrate the effectiveness of our approach in simulation using an autonomous driving system (ADS) and two closed-loop artificial pancreas systems (APS) as well as a publicly-available clinical trial dataset. The results show that a safety monitor developed with the proposed approaches demonstrates up to 4.7 times increase in average prediction accuracy (F1 score) over several well-designed baseline monitors while reducing both false-positive and false-negative rates in most scenarios.

Index Terms—Safety, Resilience, Run-time Verification, Anomaly Detection, Hazard Analysis, Cyber-Physical Systems.

1 Introduction

Rapid advances in sensing and computing technology have led to the proliferation of Cyber-Physical Systems (CPS) in various safety-critical application domains like intelligent health care and autonomous driving. However, the increasing device complexity, shrinking technology sizes, and shorter time to market have resulted in significant challenges in ensuring the reliability, safety, and security. This is evident from the increasing number of reports on accidental faults and malicious attacks targeting CPS sensors, actuators, or control software that jeopardize the system operation at run-time and lead to catastrophic consequences [1]–[4].

Significant progress has been made in improving CPS resilience using correct-by-construction techniques like quantitative risk assessment [5], control-theoretic hazard analysis [6], and model-based design [7], verification [8], and control synthesis [9] using formal and mathematical models. However, CPS are still vulnerable to residual faults and security vulnerabilities that might evade even the most rigorous design and verification methods and appear at run time [10]. Thus, run-time monitoring and anomaly detection are essential for complementing such offline analysis and assurance methods.

Model-based run-time monitoring approaches for CPS rely on developing dynamic models of the physical processes, environment, and operator behavior for combined cyber-physical monitoring and more accurate detection of anomalies [11], [12]. However, developing models that can fully capture the complex system dynamics and unpredictable human behavior is challenging, and the use of simple linear models [13] often leads to inaccuracy and false alarms [14]. Further, most existing methods focus on detecting safety violations *after* they occur, which is often too late for timely recovery and mitigation [15]. On the other hand, recent data-driven approaches to anomaly detection in CPS use machine learning (ML) for improved detection accuracy and latency [16]–[19]. Nevertheless, they suffer from the common problems of ML-based systems, including limited availability of labeled datasets, lack of model transparency, and suboptimal performance for corner cases [20]–[23].

In this paper, we propose a *hybrid* model/knowledge and data-driven approach to the design and synthesis of context-aware safety monitors that can be integrated with CPS control software at design time to continuously detect and mitigate the execution of unsafe controller commands caused by accidental faults or malicious attacks at run-time (Fig. 1). Our approach combines the formal specification of safety context and unsafe control commands based on hazard analysis and domain knowledge with data-driven optimization techniques to generate safety properties to be checked by the run-time monitor. We use collected data from the closed-loop CPS simulation or run-time operation to refine the safety properties with relevant parameters or to train ML models under the guidance of the generated safety specifications. The monitor is synthesized as a wrapper around the control software with only access to the input-output interface (sensor and actuator values) to perform real-time context inference and execution of safety specification formulas for *preemptive* detection of unsafe control commands and prediction of hazards.

The main contributions of the paper are as follows:

- Designing a framework to generate formal safety context specifications that can *predict* hazards for a reasonable time window in CPS. This framework closes the gap be-

- *X. Zhou, J. Aylor and H. Alemzadeh are with the Department of Electrical and Computer Engineering, University of Virginia, Charlottesville, VA 22904.E-mail: {xugui, jha, alemzadeh}@virginia.edu*
- *B. Ahmed is with University of Florida, Gainesville, FL 32611.*
- *P. Asare is with University of Toronto, Toronto, ON, Canada.*

Manuscript received Month Day, 2021.

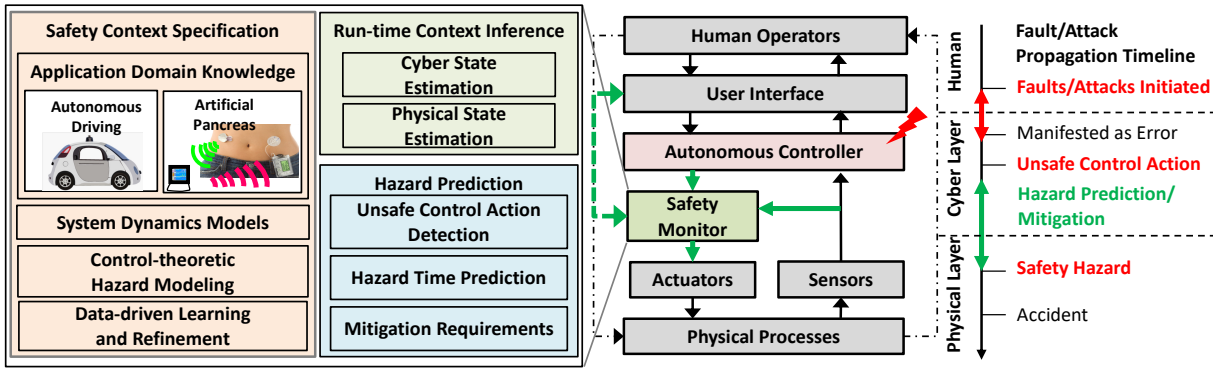


Fig. 1: CPS Control System with the Context-aware Safety Monitor and Fault Propagation Timeline.

tween design-time control-theoretic hazard analysis and run-time safety monitoring (Section 3.2). The generated signal temporal logic (STL) formulas can be used for run-time identification of unsafe control actions that potentially lead to hazards in different CPS with the same functional specification.

- Proposing two approaches of combining domain knowledge with data-driven optimization techniques for safety context specification and learning (Section 3.4), including (i) weakly supervised learning to optimize the scenario-specific STL formulas for safety monitoring using both fault-free and faulty data collected from the closed-loop cyber-physical system; (ii) neural network models trained using fault-free and faulty data that are regularized using a custom loss function to encourage satisfying the scenario-specific safety specifications and improve the transparency of the ML-based monitors.
- Developing a reaction time estimator that can predict the maximum time budget within which the control software should issue a mitigation action to prevent potential hazards. The proposed reaction time estimator could work together with the safety monitors to identify the recovery action requirements and restrain false alarms (Section 3.6).
- Demonstrating the generalizability of our proposed approach by applying it to two closed-loop artificial pancreas systems (APS) and an autonomous driving system (ADS) (Section 4). We evaluate the proposed context-aware monitor in comparison to several baselines that represent state-of-the-art solutions in anomaly detection and safety monitoring literature, including general guidelines, model predictive control, and machine learning (ML) (Section 5.3). Experimental results show that the safety monitors developed using our approach achieve up to 4.7 times increase in average prediction accuracy (F1 score) while maintaining low false-positive and false-negative rates (Section 5.4). The effectiveness of the proposed method has also been attested to using a publicly-available dataset from a clinical trial of 168 diabetes patients (Section 5.4.6). Further, the developed safety monitor has the ability to protect the target system from stealthy attacks (e.g., surge attacks or bias attacks [14]) (Section 5.4.5).

2 Preliminaries

In CPS, interconnected software and hardware components are deeply intertwined with the physical world. The core of CPS are controllers that keep the system robust to the unexpected environment by estimating the physical system state based on sensor readings and changing the state through sending control commands to actuators according to desired control targets and strategies. Safety hazards and accidents might occur as a result of unsafe commands issued by the controller due to accidental faults or malicious attacks on the controller (algorithm, software, or hardware), sensor data, or actuators.

Vulnerable Controllers: Most previous research on CPS safety and security have focused on detecting safety-critical faults or attacks on sensor data before they reach the controller by implementing redundant hardware [24] or software [18] components, quickest change detection techniques [14], invariant monitoring [25], [26], or ML-based anomaly detection [16]. However, less attention has been paid to accidental faults that directly compromise the control software or hardware functionality or attacks that exhibit the malicious behavior *after* the controller has received the sensor data. These attacks or faults could exploit the vulnerabilities in the communication channels [1], [27], mobile and app-based controllers [28], and software development processes [29], bypass the defense mechanisms mentioned above, and expose the system and its users to potential safety hazards.

We aim to address this problem by designing a run-time monitor that detects potentially unsafe control commands issued by the CPS controller, regardless of their originating causes, and stops their execution on the actuators to prevent hazards. We focus on the faults/attacks that target the controller itself [2], [27], [30], [31] or manifest on the controller output. The proposed monitor only requires access to the input-output interface for observing the sensor data and actuator commands, inferring system context, and making decisions (Fig. 1). So it can be integrated as a wrapper with the target CPS controller without any changes to the controller itself.

We assume the sensor data received by the monitor is protected using the aforementioned methods in the literature. Also, the safety monitor should be designed with more straightforward and transparent logic than the controller to be easily verified and made tamper-proof (e.g., using protective memories or hardware isolation [32] [33]). To minimize the chance of being compromised by attackers and maximize the protected area, the safety monitor should be implemented at the latest computation stage in the control system and as close as possible to the actuators (e.g., inside the insulin pump for APS).

Cyber-Physical System Context: Safety, as an emergent property of CPS, is context-dependent and should be en-

sured by applying a set of constraints on the system's behavior and control actions given the current system state [17], [34]–[37]. Our goal is to design a *context-aware* monitoring system that (i) evaluates whether the control commands issued by the controller given the current inferred context might lead to any future hazards and (ii) prevents the delivery of unsafe commands by issuing context-specific corrective commands without introducing new hazards [38].

However, designing such a context-aware monitoring system is challenging as it requires precise modeling and analysis of the multi-dimensional system context, including the physical processes, the environment, the cyber components that affect the physical processes, and their interactions in both temporal and spatial domains. Although recent ML-based approaches attempt to address the challenges in developing complex physical models by learning from data [12], [27], [39], [40], they still suffer from limited generalization and lack of transparency [21], which are essential for ensuring user trust and successful recovery and mitigation.

To address these challenges, we combine the modeling of domain knowledge and human expertise with data-driven techniques to improve the monitor's accuracy and transparency. Specifically, we adopt the control-theoretic notion of system context from the STAMP accident causality model [34] and propose a formal framework for specification and design of context-aware hazard detection and mitigation requirements, which are further optimized with data collected from closed-loop simulation or actual system operation. Our proposed framework bridges the gap between the high-level safety requirements identified from hazard analysis and the low-level formal specification of safety properties used for run-time monitoring.

Preemptive Hazard Detection: Previous works on anomaly detection have mainly focused on improving the accuracy in detecting faults/attacks, with less attention to hazard prediction or to how detection latency can impact recovery and hazard mitigation. Fig. 1 presents the fault propagation timeline in a typical CPS, from which we can observe that there is a time gap between the activation of faults/attacks and their propagation to cause erroneous cyber states, the generation of unsafe control commands in the cyber layer, and finally propagation of errors to the physical layer and occurrence of hazards. We define the time between activation of a fault to the occurrence of a hazard as *Time-to-Hazard (TTH)* and the time difference between the detection of anomaly and the occurrence of a hazard as *Reaction Time* (see Fig. 9 in Section 5.4). Reaction Time measures the timeliness of the monitor and whether the time left is enough to mitigate potential hazards and recover the control system by transitioning to a safe state. It indicates the maximum time budget for taking any mitigation actions before the hazard occurs, with positive values representing *early detection*.

Our goal is to ensure successful hazard mitigation by maximizing reaction time and minimizing the monitor detection latency. To achieve this, we focus on (i) detecting potential unsafe control commands that indicate early signs of impending hazards rather than detecting the hazardous states, (ii) minimizing the complexity of run-time monitor logic, and (iii) developing a reaction time estimator that predicts the maximum time left to the occurrence of hazards and helps with identifying potential false alarms.

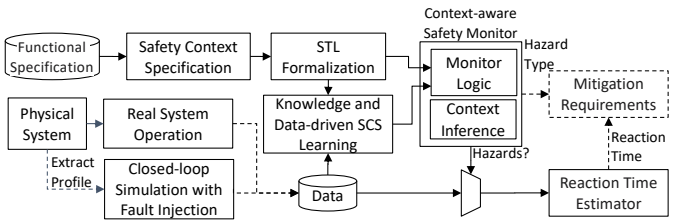


Fig. 2: Framework for Design of Context-aware Monitors.

3 Context-Aware Monitor Design

Fig. 2 shows the overall framework for designing context-aware safety monitors. Our approach starts with generating formal safety context specifications (SCS) from hazard analysis. Experimental data from the operation/simulation of the closed-loop system is used to refine the safety specifications with relevant parameters. The formalized safety specification is integrated into a monitor for run-time monitoring through either a weakly supervised approach or by training an ML model with a custom loss function. The monitor is then synthesized as a wrapper around the controller to perform real-time execution of safety specification formulas for preemptive detection of hazards. The information generated by the safety monitor, including the predicted hazard type and reaction time, will define the requirements for context-specific hazard mitigation strategies.

3.1 Model of System Dynamics

In every control cycle t, the CPS controller uses the sensor measurements $x_t = (x_{1_t}, \ldots, x_{n_t}) \in \mathbb{R}^n$ to estimate the physical system status and decide on a control action, u_t, from a finite set of high-level control actions $U = \{u_1, \ldots, u_r\}$ (e.g., in ADS, high-level control commands *Acceleration* and *Deceleration*). Each high-level control action will be translated into the values of different low-level control output variables (e.g., *gas* and *brake*) which are then sent to the actuators. Upon execution of the issued control command by the actuators, the physical system will transition to a new state estimated by x_{t+1} in the state space.

3.1.1 Regions of Operation

We identify the unsafe/hazardous region $\mathcal{X}_h$ as the set of system states that lead to accidents and can be further partitioned into regions associated with particular safety hazard types H_i. The safe/target region $\mathcal{X}_*$ can be defined based on the goals and guidelines of the specific application. The set of states not included by either of these regions is referred to as the possibly hazardous region $\mathcal{X}_{*<h}$. Example regions of operation for APS and ADS are presented in Fig. 3. The goal of the APS controller is to keep the patient's Blood Glucose (BG) in the target range of 120-150 mg/dL, while the ADS controller's goal is to maintain a safe following distance of 2-4 seconds to the lead vehicle [41]. The unsafe regions for each example are indicated based on the definitions of hazards as later described in Section 4.1.

In this paper, we define the regions of operation similar to previous works [42], [43] but more conservatively, by minimizing the safe region and maximizing the unsafe region [44], so that we do not miss detecting any unsafe

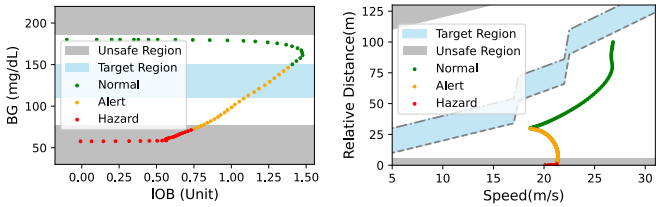


Fig. 3: Example Regions of Operation: Artificial Pancreas System (APS) (Left). Autonomous Driving System (ADS) (Right).

control actions. The possible overlaps between regions or uncertain parts of the safe region are included in the potentially unsafe region.

3.1.2 Unsafe Control Actions

A sequence of *cyber control actions* $U_t = \{u_{t-k+1}, ..., u_{t-1}, u_t\}$ issued in k consecutive control cycles are considered unsafe if upon their sequential execution in a given state sequence $X_t = \{x_{t-k+1}, ..., x_{t-1}, x_t\}$, the system will eventually transit to a state in $\mathcal{X}_h$ within the period T that U_t can affect the state space. The length of the control action sequence varies in different applications. For example, in a robot control system with tight real-time constraints, only one or a few unsafe control actions might lead to a safety hazard [27]. In contrast, in a slower control system like APS, a sequence of unsafe control actions may need to last for an extended time period (e.g., 30 minutes) to result in a hazard eventually [38].

3.2 Safety Context Specification (SCS) Framework

We propose a formal framework for the specification of safety context, consisting of two parts: (i) the Unsafe Control Action Specification (UCAS) that describes the system context under which specific control actions are potentially unsafe and can be used for predicting hazards; and (ii) the Hazard Mitigation Specification (HMS) that identifies mitigation actions to prevent potential hazards resulting from the unsafe control actions issued by the controller.

3.2.1 Unsafe Control Action Specification (UCAS)

To reduce the complexity in specifying the overall system context, we define $\mu(x_t) = (\mu_1(x_t), \ldots, \mu_m(x_t)) \in \mathbb{R}^m$, where $\mu_i(x_t)$ is a transformation of x_t, which could be the polynomial, derivative, or other possible functions of x_t, modeling more complex combinations of state variables and their rates of change. The set of all possible values of $\mu(x_t)$ is denoted by $\mathcal{M}$. We describe the *system context* $\rho(\mu(x_t))$ as the subsets of $\mathcal{M}$, defined by ranges of variables in $\mu(x_t)$, that can be mapped to the regions $\{\mathcal{X}_*, \mathcal{X}_{*<h}, \mathcal{X}_h\}$.

We define the set of all tuples $(\rho(\mu(x_t)), u_t, H_i)$ as the unsafe control action specification (UCAS) such that $(\rho(\mu(x_t)), u_t) \mapsto H_i \subset \mathcal{X}_h$, specifying the system context $\rho(\mu(x_t))$ under which by issuing a control action u_t the system eventually transitions to a new context within the H_i hazard partition in the hazardous region $\mathcal{X}_h$. The UCAS can be generated using the following steps:

1) Define the set of accidents (A) and hazards (H) of interest for the system using the control-theoretic hazard analysis method.
2) Determine the targeted transformations $\mu(x_t)$ and the sets $\rho(\mu(x_t))$ related to the hazard as thoroughly as possible based on an observable set of variables x_t. The exact thresholds for each variable that identify each subset do not need to be known.
3) Enumerate all the combinations of $\rho(\mu(x_t))$ and $u_t \in U$.
4) Determine the combinations that might lead to transitions to a hazardous region $H_i \subset \mathcal{X}_h$, and add tuples $(\rho(\mu(x_t)), u_t, H_i)$ into the UCAS set.

Steps 1 and 2 need to be defined manually based on domain knowledge and input from domain experts. Step 3 can be automated according to definitions in the first two steps [45], and so can step 4 using dynamic modeling and simulation [46].

3.2.2 Hazard Mitigation Specification (HMS)

HMS is defined as a set of tuples that have the form $(\rho(\mu(x_t)), \boldsymbol{u}^\rho)$, where $\boldsymbol{u}^\rho$ is the set of safe control actions under the context $\rho(\mu(x_t))$ that lead to the transition to the safe region $\mathcal{X}_*$ and prevent hazards. The HMS can be generated through the following steps:

1) For each specified context $\rho(\mu(x_t)$ in UCAS, find all the control actions $u_t^c \in U$ such that $(\rho(\mu(x_t)), u_t^c) \mapsto \mathcal{X}_*$ and add them to $\boldsymbol{u}^\rho$, the set of safe mitigating control actions for that context.
2) Add tuples $(\rho(\mu(x_t)), \boldsymbol{u}^\rho)$ into the HMS set.

3.3 Formalization of SCS in Temporal Logic

To synthesize the SCS into machine-checkable language/logic that can be used for safety monitoring, we convert the unsafe control action specifications (UCAS) into a set of safety properties described in STL formalism. STL is a formal language for specifying temporal properties of continuous signals, and is widely used for rigorous specification and run-time verification of requirements in CPS [47]. We utilize the bounded-time variant of STL, where all temporal operators are associated with upper and lower time bounds.

The STL formula ϕ_h for a specific UCAS $(\rho(\mu(x_t)), u_t, H_i)$ is described as follows:

$$G_{[t_0,t_e]}(\varphi_1(\mu_1(x_t)) \wedge \ldots \wedge \varphi_m(\mu_m(x_t)) \wedge u_t \implies F_{[0,T]}H_i) \quad (1)$$

where, F is the eventually operator $\Diamond$ and each $\varphi_i(\mu_i(x_t))$ is an atomic predicate representing an inequality on $\mu_i(x_t)$ in the form of $\mu_i(x_t)\{<, \leq, >, \geq\}\beta_i$ or its combinations, with the thresholds β_i defining the boundary of each dimension $\rho(\mu_i(x_t))$ in the system context $\rho(\mu_(x_t))$. The formula ϕ_h holds true globally (denoted by the G operator) between start time t_0 and end time t_e during the system operation.

The UCAS for a sequence of control actions U_t, issued under a state sequence $X_t = \{x_{t-k+1}, ..., x_{t-1}, x_t\}$ over a window of k control cycles, is formalized as Φ_h:

$$G_{[t_0,t_e]}(\varphi_1(f(\mu_1(X_t))) \wedge \ldots \wedge \varphi_m(f(\mu_m(X_t))) \wedge f(U_t) \implies F_{[0,T]}H_i)$$

where, $\mu_i(X_t) \doteq \{\mu_i(x_{t-k+1}), \ldots, \mu_i(x_t)\}$ and $f(\cdot)$ represents an aggregation function such as average, Euclidean norm, or regression over k transformed measurements. When k takes the value of 1, this equation is identical to Eq. 1 which considers the consequences of a single control action.

Likewise, we convert the HMS $(\rho(\mu(x_t)), u_t^c) \mapsto \mathcal{X}_*$ to

$$G_{[t_0,t_e]}((F_{[0,t_s]}(u_t^c))\mathcal{S}(\varphi_1(\mu_1(x_t)) \wedge \ldots \wedge \varphi_m(\mu_m(x_t)))) \quad (2)$$

which requires that $u_t^c \in \boldsymbol{u}^\rho$ should be taken within period t_s since (denoted by the S operator) the system enters context $(\varphi_1(\mu_1(x_t)) \wedge \ldots \wedge \varphi_m(\mu_m(x_t)))$. This should hold globally during the system operation.

The time parameter t_s specifies the requirement for the latest possible time a mitigation action should be initiated after a potential unsafe control action is detected to prevent hazards. This time is dependent on many factors, including the context $\rho(\mu(x_t))$, the nature of the various safe control actions $u_t^c \in \boldsymbol{u}^\rho$, and the CPS controller being fast or slow, and could be specified based on the domain knowledge and practical settings. The specifics of determining this time requirement, in general, are beyond the scope of this paper. The estimated time between the activation of a fault in the system and the occurrence of a hazard (defined as Time-to-Hazard in Section 5) can provide an upper bound for specifying this time requirement.

3.4 Knowledge and Data-Driven SCS Learning

The STL formulas generated for SCS can be further integrated with the data collected from the closed-loop cyber-physical system to design the monitor logic. In this section, we present two different methods based on STL parameter optimization and machine learning with customized loss functions that enable the integration of data with knowledge for monitoring safety properties.

3.4.1 Optimization of STL Formulas

We develop an approach for learning the unknown boundary parameters β_i in the STL formulas (Eq. 1) using actual or simulated data collected from the system using ML methods [48] [49]. Then the final STL formulas with the estimated parameters will be synthesized into logic for runtime monitoring. Specifically, we use software fault injection (FI) on a closed-loop CPS to generate example hazardous data traces that satisfy the STL formulas for UCAS and use them for learning unknown STL parameters and for adversarial training of the monitor. As shown in Fig. 2, data traces from real system operation can also be used for the development of simulation models and faulty data traces and/or for active learning and updating of the monitor at run-time in an actual application.

We solve the problem of learning unknown thresholds β_i from a set of data traces $\mathcal{D}$ by formulating the following optimization problem:

$$\begin{aligned} &\text{minimize} \sum_{\mathcal{H}} loss(r);\ s.t. \\ &r = \mu_i(d(t)) - \beta_i \geqslant 0, \forall d \in \mathcal{H} : d \models \phi_h \end{aligned} \tag{3}$$

If the STL formula ϕ_h (Eq. 1) is satisfied (denoted by $\models$ operator that takes a binary value from $\{True, False\}$) by a subset of hazardous traces $\mathcal{H} \subset \mathcal{D}$, the degree of satisfiability of ϕ_h for a data trace $d \in \mathcal{H}$ at time t can be measured by a robustness metric $r = \mu_i(d(t)) - \beta_i$ (for predicate $\mu_i(x_t) \geq \beta_i$). The goal of optimization is to minimize the absolute value of r as a loss function over all traces in $\mathcal{H}$ to achieve tight properties [50]. In this paper, we designed a Tight Mean Exponential Error (TMEE) loss function, as shown below:

$$loss(r) = E[e^{-r} + r - \frac{1}{r + e^{-2r}}],\ r = \mu_i(d(t)) - \beta_i \tag{4}$$

which learns tight thresholds while ensuring that the faulty data traces satisfy the UCAS STL formulas. We adopted a quasi-Newton optimization method, called L-BFGS-B [51], for learning the unknown thresholds. This algorithm uses the gradient of the proposed loss function (Eq. 4) and the estimated inverse Hessian matrix, which is calculated using two-loop recursion [52], to guide the optimization.

Although our STL learning approach is similar to a previous work TeLEx [50], the proposed TMEE loss function achieves a faster convergence in learning unknown thresholds for the STL formulas. Specifically, our preliminary experiments involving 50 simulation runs of APS controller with the data from a simulated diabetic patient showed that, on average, our optimization method could lead to convergence within a very short time (0.02s vs. 21.79s) with a much smaller loss value (1.15 compared to $\gg$100) compared to TeLEX, leading to learning tighter thresholds. Further, the safety monitor synthesized based on the tight thresholds learned using our approach had a higher accuracy than the monitor rules learned using TeLEx (F1-score of 0.94 vs. 0.60).

3.4.2 ML Optimization with Customized Loss Functions

Another data-driven approach to integrating the SCS STL formulas into the safety monitor is to train an ML model using data traces collected from the cyber-physical system and guide the learning process using a custom loss function [53]. Specifically, we model the task of detecting an unsafe control action as a context-specific conditional event, as shown below:

$$y_t = p(\exists t' \in [t, t+T] : x_{t'} \in \mathcal{X}_h | f(X_t), f(U_t)) \tag{5}$$

Given the control action sequence U_t executed under the system state sequence X_t, the ML model outputs a binary y_t that classifies U_t to safe or unsafe. Section 5.3.3 will present different neural-network architectures for implementing such ML-based monitor.

We encode the STL formulas generated for UCAS as a custom loss function that penalizes the ML model during the training process if the prediction does not match the specified safety properties:

$$loss = loss_{ex} + w \left| y_t - I\left(\bigvee_{\Phi_h \in UCAS} f(\mu(X_t)) \models \Phi_h \right) \right| \tag{6}$$

where, $loss_{ex}$ is the baseline ML model loss function (e.g., cross-entropy loss), w is a weight parameter, y_t is the output prediction of the ML model, and $I(\cdot)$ is an indicator function indicating whether the aggregated values of the estimated state variables for a measurement window, $f(\mu(X_t))$, satisfy any of the UCAS STL formulas Φ_h (with unrefined thresholds). The specific value of weight parameter w depends on the design requirements and system scenarios. The larger the weight parameter is, the more the system context and safety specification would interfere with the training process. In this work, we choose a value so that the additional loss is comparable to the original loss of the output layer of the ML model.

3.5 Run-time Cyber-Physical Context Inference

The SCS STL formulas for the synthesis of the monitor are described in terms of high-level and human-interpretable

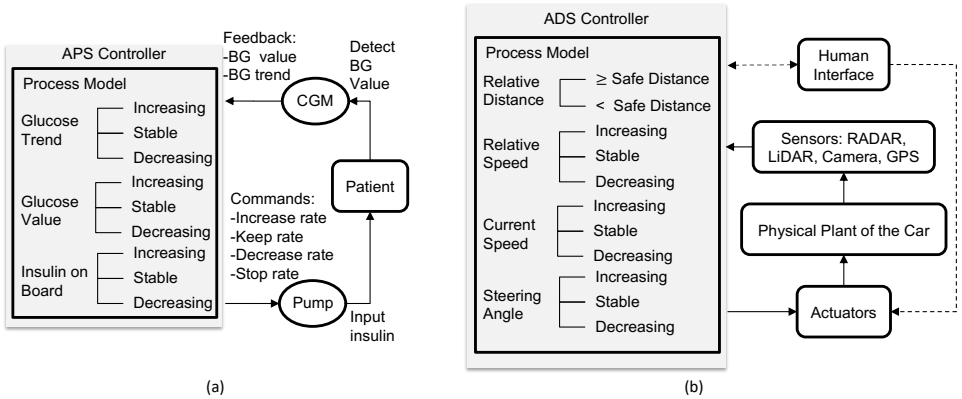


Fig. 4: (a) Artificial Pancreas System and a Typical APS Controller; (b) Autonomous Driving System and a Typical ADS Controller.

estimated states (e.g., Headway Time (HWT)) and control actions (e.g., $acceleration$ in ADS) and might be different from the low-level sensor measurements (e.g., RADAR data) and output control commands (e.g., the amount of gas or $brake$) executed on the actuators.

In order to close the semantic gap between human-interpretable safety requirements and low-level measurements observed by the safety monitor and map the system's state to the STL formulas, the monitor needs to be equipped with capabilities for run-time inference of the cyber and physical states. Specifically, the monitor will infer the high-level control actions issued by the control software based on the low-level control commands sent to the actuators and the non-observable physical states used by the control algorithm based on the sensor measurements. This can be considered a partial replication of the controller's state estimation and control algorithms inside the monitor.

3.6 Reaction Time Estimation

For further evaluation of the quality of predictions made by the monitor and synthesis of a context-aware mitigation mechanism based on the HMS (Equation (2)), we need to identify the unknown time parameter t_s in addition to safety thresholds β_i. This time is bounded by the maximum reaction time and is dependent on many factors, including the context $\rho(\mu(x_t))$ and the nature of the various safe control actions $u_t^c \in \boldsymbol{u}^\rho$. In this work, we develop a reaction time estimator using a two-layer stacked LSTM model that could predict the latest possible time a mitigation action should be initiated (after a potential unsafe control action is detected) to prevent hazards.

4 Case Studies

To demonstrate the generalization and effectiveness of our approach, we evaluated our methodology for run-time monitoring in two different case studies of Artificial Pancreas Systems (APS) and Autonomous Driving Systems (ADS).

The APS controller (Fig. 4a) estimates the current patient status (BG value and Insulin on Board (IOB)) based on Continuous Glucose Monitor (CGM) readings and injects the proper amount of insulin into the patient through a pump. The ADS Adaptive Cruise Control (ACC) system (Fig. 4b) measures relative distance and relative speed to the lead vehicle based on the RADAR and car sensors readings, estimates the steering angle and brake status based on the measurements of car sensors, and maintains a target following distance with the lead vehicle by issuing acceleration or deceleration control actions and outputting the appropriate amount of gas and brake pressure.

The following subsections present the process for generating SCS, labeling data for SCS learning, and cyber-physical context inference for mapping measurements to SCS formulas for these two case studies.

4.1 SCS Generation

Step 1: We first identified the set of accidents and the hazardous system states as the result of potential unsafe control actions issued by the controller that could lead to accidents.

For the APS, the set of accidents (A) and hazards (H) of interest include:

- **A1:** Complications from hypoglycemia, including seizure, loss of consciousness, and death.
- **A2:** Complications from hyperglycemia, including tissue damage and morbidities such as retinopathy and in extreme cases, death [54].
- **H1:** Too much insulin is infused, which will reduce the BG and might lead to A1.
- **H2:** Too little insulin is infused, which will cause the BG to increase and might lead to A2.

For the ADS, the following set of possible accidents and hazards were considered:

- **A3**: Forward collision with the lead vehicle.
- **A4**: Collision with the trailing vehicle or causing traffic congestion.
- **H3**: Autonomous vehicle violates maintaining safety distance with the lead vehicle, which may result in A3.
- **H4**: Autonomous vehicle decelerates to a complete stop without a lead vehicle, which may lead to A4.

Step 2: We then identified the transformations of interest ($\mu(x_t)$) for specifying system context. For example, for APS with sensor measurements x_t = (BG_t, IOB_t) we define $\mu(x_t)$=(BG_t, dBG_t/dt, IOB_t, $dIOB_t/dt$), including the state variable BG_t and IOB and their rates of change.

Steps 3-4: Then, the list of potential UCAS for each system was generated by identifying the combinations of specific ranges in $\mu(x_t)$ and control actions (e.g., $u_t \in \{u1, u2, u3, u4\}$ that can potentially be hazardous and lead to accidents of interest (see Table 1)).

TABLE 1: STL Safety Context Specifications for APS and ADS

CPS	Rule No.	STL Description of Safety Context	Implied Hazard Type
APS	1	$G_{[t_0,t_e]}((BG > BGT \wedge BG' > 0) \wedge (IOB' < 0 \wedge IOB < \beta_1) \wedge u_1 \implies$	$F_{[0,T]}H2)$
	2	$G_{[t_0,t_e]}((BG > BGT \wedge BG' > 0) \wedge (IOB' = 0 \wedge IOB < \beta_2) \wedge u_1 \implies$	$F_{[0,T]}H2)$
	3	$G_{[t_0,t_e]}((BG > BGT \wedge BG' < 0) \wedge (IOB' > 0 \wedge IOB < \beta_3) \wedge u_1 \implies$	$F_{[0,T]}H2)$
	4	$G_{[t_0,t_e]}((BG > BGT \wedge BG' < 0) \wedge (IOB' < 0 \wedge IOB < \beta_4) \wedge u_1 \implies$	$F_{[0,T]}H2)$
	5	$G_{[t_0,t_e]}((BG > BGT \wedge BG' < 0) \wedge (IOB' = 0 \wedge IOB < \beta_5) \wedge u_1 \implies$	$F_{[0,T]}H2)$
	6	$G_{[t_0,t_e]}((BG < BGT \wedge BG' < 0) \wedge (IOB' > 0 \wedge IOB > \beta_6) \wedge u_2 \implies$	$F_{[0,T]}H1)$
	7	$G_{[t_0,t_e]}((BG < BGT \wedge BG' < 0) \wedge (IOB' < 0 \wedge IOB > \beta_7) \wedge u_2 \implies$	$F_{[0,T]}H1)$
	8	$G_{[t_0,t_e]}((BG < BGT \wedge BG' < 0) \wedge (IOB' = 0 \wedge IOB > \beta_8) \wedge u_2 \implies$	$F_{[0,T]}H1)$
	9	$G_{[t_0,t_e]}((BG > BGT \wedge IOB < \beta_9) \wedge u_3 \implies$	$F_{[0,T]}H2)$
	10	$G_{[t_0,t_e]}((BG < \beta_{12}) \wedge \neg u_3 \implies$	$F_{[0,T]}H1)$
	11	$G_{[t_0,t_e]}((BG > BGT \wedge BG' > 0) \wedge (IOB' <= 0 \wedge IOB < \beta_{10}) \wedge u_4 \implies$	$F_{[0,T]}H2)$
	12	$G_{[t_0,t_e]}((BG < BGT \wedge BG' < 0) \wedge (IOB' >= 0 \wedge IOB > \beta_{11}) \wedge u_4 \implies$	$F_{[0,T]}H1)$
ADS	1	$G_{[t_0,t_e]}((HWT < \beta_{21}) \wedge (RS > 0 \wedge RS' > 0) \wedge \neg u_{22} \implies$	$F_{[0,T]}H3)$
	2	$G_{[t_0,t_e]}((HWT < \beta_{22}) \wedge (RS > 0 \wedge RS' = 0) \wedge \neg u_{22} \implies$	$F_{[0,T]}H3)$
	3	$G_{[t_0,t_e]}((HWT < \beta_{23}) \wedge (RS > 0 \wedge RS' < 0) \wedge u_{21} \implies$	$F_{[0,T]}H3)$
	4	$G_{[t_0,t_e]}((HWT > \beta_{24}) \wedge (RS < 0 \wedge RS' > 0) \wedge \neg u_{21} \implies$	$F_{[0,T]}H4)$
	5	$G_{[t_0,t_e]}((HWT > \beta_{25}) \wedge (RS < 0 \wedge RS' = 0) \wedge \neg u_{21} \implies$	$F_{[0,T]}H4)$
	6	$G_{[t_0,t_e]}((HWT > \beta_{26}) \wedge (RS < 0 \wedge RS' < 0) \wedge u_{22} \implies$	$F_{[0,T]}H4)$

* BGT: BG target value; $BG' = dBG/dt$, $IOB' = dIOB/dt$;
* HWT: Headway Time = Relative Distance/Current Speed [55]; RS: Relative Speed = Current Speed - Lead Speed; $RS' = dRS/dt$;
* $u_{1,2,3,4}$:decrease_insulin, increase_insulin, stop_insulin, keep_insulin;
* $u_{21,22}$: Acceleration, Deceleration; t_0, t_e: start time and end time of the simulation.

Table 1 shows the final safety specifications described in STL formalism for both APS and ADS. For example, the last row for APS is the formal representation of a UCAS, $(\rho(\mu(x_t)) = (BG < BGT,\ BG' < 0,\ IOB' \geqslant 0,\ IOB > \beta_{11}),\ u_4, H1)$, specifying that under the system context where the BG is less than the target and has been decreasing and IOB is more than a certain threshold β_{11} and keeps increasing, the control action u_4 (*keep_insulin*) is unsafe and will most likely result in an H1 hazard if issued by the controller. This is an example of a safety rule that can be identified or verified in consultation with domain experts. Further, these rules can be synthesized as monitor logic and applied to different implementations of the CPS controllers (e.g., different APS or ADS controllers) with the same functional specifications.

4.2 Hazard Labeling for SCS Learning

For data-driven refinement and adversarial training of SCS, we need examples of faulty data traces collected from closed-loop cyber-physical system simulations or actual operations. This data should include the time-series sensor measurements as well as the control actions issued by the controller and be labeled with the time instances when the system is in a hazardous state. Note that the goal of the monitor is to detect unsafe control actions and predict these hazardous states ahead of time, so the method used for labeling the hazards cannot be used by the monitor. In this paper, we label the data automatically using common objective metrics introduced and the guidelines used by the research community and in practice, as described next.

For APS, we utilized the notion of the Risk Index (RI) [56], [57] that captures both the glucose variability and its associated risks for hypo- and hyperglycemia to label the data. We calculated low (LBGI) and high (HBGI) BG risk for a data-trace $\mathcal{D}$ of BG readings using the following equations:

$$risk(BG) = 10 * (1.509 * [(ln(BG))^{1.084} - 5.381])^2 \quad (7)$$
$$LBGI = 1/n \sum_{\mathcal{D}} risk(BG); \text{for each } BG < 112.517$$
$$HBGI = 1/n \sum_{\mathcal{D}} risk(BG); \text{for each } BG \geqslant 112.517$$

We identified a window (e.g., one hour) of BG readings as hazardous if the risk indices exceeded a high-risk threshold (e.g., $LBGI > 5$ and $HBGI > 9$ as defined by previous works [57], [58]) and kept increasing, indicating a high chance of hypo- or hyperglycemia.

For ADS, we label the data points as hazardous if the relative distance between the autonomous vehicle and the leading vehicle is non-positive or the autonomous vehicle decelerates to a complete stop with a considerable relative distance (e.g., greater than 100 meters [59] that is the range of a medium-range radar [60]) to the leading vehicle, which might lead to a potential collision with the trailing vehicle or causing congestion.

4.3 Context Inference for SCS Matching

For APS, the state variable, Insulin On Board (IOB), might not be observable directly by the safety monitor. Therefore, we need to derive its value from a sequence of insulin rate history. After insulin is injected into a patient's body, the IOB will gradually increase and reach the maximum level at t_{peak} (e.g., 75 minutes) and then start to decrease to zero. We calculated the IOB before the peak time using the equation [61]:

$$IOB(t) = I(t_0) * [-k_1(0.2(t-t_0)+1)^2 + k_1(0.2(t-t_0)+1) + 1] \quad (8)$$

and derived the IOB between peak time and the end of the duration of insulin action using the following equation:

$$IOB(t) = I(t_0) * [k_2(t-t_0-t_{peak})^2 - k_3(t-t_0-t_{peak}) + 0.55556] \quad (9)$$

where k_i are coefficients. The accumulated IOB under the effect of an insulin rate sequence is the integral of IOB calculated using the above equations.

For ADS, we estimate the high-level state variables headway time and relative speed based on the current speed of the autonomous vehicle and relative distance between the autonomous vehicle and the leading vehicle measured by low-level car sensors like GPS and Radar.

In both case studies, the monitor maps the low-level control commands issued by the controller into the high-level control actions described in SCS by calculating the rate of change in a window of measurements. For example, an *increase_insulin* or *decrease_insulin* control action will be detected by calculating the slope of insulin samples.

5 Experimental Evaluation

This section presents our experiments and results on the evaluation and comparison of the proposed context-aware monitors designed using two SCS learning approaches presented in Section 3.4: (i) the STL optimization with threshold learning (referred to as CAWT, short for *Context-Aware With refined Thresholds*) and (ii) ML optimization with customized loss functions (referred to as ML-Custom).

We developed an open-source simulation environment (see Fig. 5) that integrated the closed-loop simulation of two example APS and ADS control systems with a software FI engine to evaluate different safety monitors.

For APS, we integrated two widely-used APS controllers (OpenAPS [61] and Basal-Bolus [62]) with two different patient glucose simulators, including Glucosym [63] and UVA-Padova Type 1 Diabetes Simulator [64]. The Glucosym simulator contains models of 10 actual Type I diabetes patients aged 42.5 $\pm$ 11.5 years [65]. The state-of-the-art

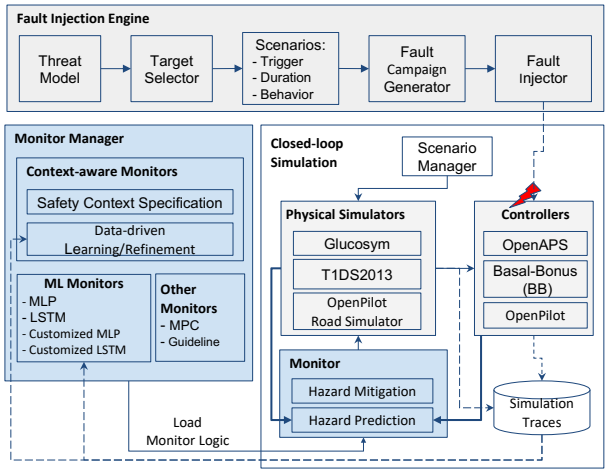


Fig. 5: Experimental Setup for Evaluation of Different Safety Monitors, Integrating Three Closed-loop Simulation Platforms (Glucosym Simulator with OpenAps Controller, UVA-Padova T1DS2013 Simulator with Basal-Bolus Controller, and OpenPilot Road Simulator and Controller), and Software FI Engine. [Available Online: https://github.com/UVA-DSA/CPS-Runtime-Monitor]

UVA-Padova Type 1 Diabetes Simulator S2013 (T1DS2013) contains 30 virtual patients, which were demonstrated to be representative of the T1DM population observed in a clinical trial [66], and has been approved by the FDA for pre-clinical testing of APS [64], [67]. This closed-loop testbed was also validated using actual data from a clinical trial, and the simulated data was shown to satisfy the requirements of relevance, completeness, accuracy, and balance [68] for developing ML models [69].

For ADS, we used OpenPilot [70], an open-source alpha quality driving agent introduced by Comma.ai which has been used in actual cars on the road by over 1,500 monthly active users. OpenPilot controller provides the adaptive cruise control (ACC) and automated lane centering (ALC) capabilities to over 150 supported car makes and models (e.g., Honda Civic 2016-2020, Acura RDX 2016-2021, Toyota RAV4 2016-2021) using an additional hardware EON Dashcam DevKit [71] that can control the gas, brake, and steering.

We ran the experiments with both APS controllers and simulators as well as OpenPilot (v.0.4.2) on an x86_64 PC with an Intel Core i9 CPU @ 3.50GHz and 32GB RAM running Linux Ubuntu LTS. We used TensorFlow v.2.5.0 to train our ML models.

5.1 Scenario Simulations

For APS simulations, we ran the experiments with initial BG values ranging from 80 to 200 mg/dL for 150 iterations (representing 12.5 hours in an actual APS control system) without add-on meals, imitating a patient eating dinner, going to sleep, and having the next meal the following day after our simulation period. In addition, we evaluated our approaches on 20 different patient profiles (10 patients in the Glucosym simulator and 10 in the T1DS2013 simulator) to consider possible inter-patient variability.

For ADS, we ran the OpenPilot simulator and controller for 150 iterations, with each iteration simulating 200ms of real road driving. We simulated four driving scenarios that are considered high-risk in the pre-collision scenario topology report by the National Highway Traffic Safety Administration (NHTSA) [81]:

- The lead vehicle is driving at a constant speed (40mph).
- The lead vehicle accelerates and then slows down.
- The lead vehicle slows down and then accelerates.
- The lead vehicle slows down to a complete stop.

5.2 Fault Injection Experiments

We collected experimental data from closed-loop CPS simulations with FI for adversarial training and testing of the proposed safety monitor and other baseline monitors.

Threat Model: We assume that both accidental faults or malicious attacks, similar to those reported for CPS can target the CPS controller and, once activated, can manifest as errors in inputs, outputs, and the internal state variables of the CPS control software and result in the hazards defined in Section 4.1 and adverse events. For malicious attacks, we assume attackers have acquired unauthorized remote access [82] to a CPS control system through stolen credentials [30], exploiting vulnerable services [83], or insider attacks by penetrating the network [27], [84] that the target CPS controller connects to. Even for a CPS control system with no network connectivity, the attacker can exploit a USB port or Bluetooth connection to get access to the target device and deploy malware. Table 2 shows examples of such fault/attack scenarios and vulnerabilities in the control system that led to real recalls and possible adverse events.

We developed a source-level FI engine that directly perturbs the values of the controller's state variables within their acceptable ranges over a random period of time to simulate the effect of such fault and attack scenarios. We assume that errors are transient and only occur once for a specific duration per simulation. For each FI scenario shown in Table 2, the FI engine determines (i) the target state variable, (ii) the error value to inject, (iii) the trigger condition of the error, and (iv) the duration of the injected fault. We randomly chose from several different start times and durations to inject the fault, resulting in a total of 882 and 1200 fault injections for each patient and driving scenario, which translated into a total number of 2,646,000 simulation

TABLE 2: Simulated Fault and Attack Scenarios

Type	Approach	Simulated Scenario	Representative FDA Recalls	Possible Adverse Events
Truncate	Change output variables to zero value [72] [73]	Availability attack [74]	Z-1074-2013, Z-1034-2015[1]	Device Malfunction/ Hypoglycemia/ Hyperglycemia/ Injury [78]/ Death [79]
Hold	Stop refreshing selected input/output variables [14] [73]	DoS attack [75] [76]	Z-1359-2012, Z-0929-2020	
Max/Min	Change the value of targeted variables to their maximum or minimum allowed values [14] [77]	Integrity attack [72]/ Memory fault	Z-1562-2020, Z-2165-2020	
Add/Sub	Add or subtract an arbitrary or particular value to or from a targeted variable [14] [80]			

[1] Recall IDs assigned by FDA which can be searched for on https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfres/res.cfm.

samples used for training and testing different monitors. We used a 4-fold cross-validation setup for threshold learning and evaluation of our context-aware safety monitors as well as model training and testing of the baseline ML monitors.

To evaluate the performance of the safety monitor against adaptive adversaries, we also consider a more powerful attacker with all the required knowledge about the target controller and safety monitor to launch certain types of stealthy attacks (Section 5.4.5). However, a comprehensive evaluation of the proposed approach against all possible stealthy attacks (e.g., replay, zero-dynamics, pole-dynamics, and covert attacks [85], [86]) is beyond this paper's scope.

5.3 Baseline Monitors

To evaluate and compare the performance of the proposed context-aware monitors, CAWT and ML-Custom (e.g., MLP-Custom, LSTM-Custom), in accurate and timely prediction of hazards, we developed several baseline monitors representative of the existing state-of-the-art safety monitoring and defense approaches for CPS.

5.3.1 Medical Guidelines Monitor

We designed a baseline safety monitor (referred to as Guideline) according to generic medical guidelines proposed in [87] without considering the patient characteristics or control software. The Guideline monitor generates alerts when the BG value is beyond a normal range [70, 180] mg/dL, has a sharp change, or stays lower than its tenth percentile λ_{10} or higher than its ninetieth percentile λ_{90} for more than a safe period (e.g., 30 minutes).

5.3.2 Model Predictive Control Monitor

We developed two Model Predictive Control (MPC) [88], [89] baseline monitors for APS and ADS as an example of the widely used technique in process control systems.

For APS, the MPC monitor estimates the possible BG value (BG_{t+1}) after executing the pump's command (I_t) on the patient's current state (BG_t) using Bergman & Sherwin model [65] :

$$dBG(t)/dt = -(GEZI + I_{EFF}) * BG(t) + EPG + R_A(t) \quad (10)$$

where, $GEZI, I_{EFF}, EPG$ are patient-specific parameters, and $R_A(t)$ is glucose appearance rate. An alarm will be generated if the predicted BG value goes beyond the patient's normal range (same as the medical guidelines).

For ADS, we used the following dynamic model of the vehicle to develop the MPC monitor [90]:

$$\begin{aligned} dv(t)/dt = 3.33 * Gas(t) * P_{peak}/m/v(t) - 3 * Bk(t) \\ - (0.01g + 0.15v^2(t)) - GD + CRP(t) \end{aligned} \quad (11)$$

where, $v(t)$ is the current speed of the vehicle; m, $Gas(t)$ and $Bk(t)$ represent the vehicle mass, output of the gas and brake, respectively; P_{peak} is the peak power, g is the gravity of earth, GD describes the road grade, and $CRP(t)$ characterizes the impact of creep force that is a function of $v(t)$. The baseline monitor issues an alert if the vehicle would brake with a deceleration of 3 m/s^2 for at least one second [91].

5.3.3 ML-based Monitors

Similar to the ML-based monitors previously proposed in [17], [38], we trained two baseline monitors using the state-of-the-art ML approaches, Multi-layer Perceptron (MLP) and Long-Short Term Memory (LSTM).

For MLP, we modeled the task of detecting an unsafe control action as a context-specific conditional event, as shown below:

$$y_t = p(\exists t' \in [t, t+T] : x_{t'} \in \mathcal{X}_h | \bar{X}_t, \bar{U}_t) \quad (12)$$

Given an issued control action $\bar{U}_t$ at current system state $\bar{X}_t$, which are the average values of U_t and X_t, respectively, the ML model outputs a binary y_t that classifies U_t to safe or unsafe. We used a fully connected two-layer MLP, comprising 256 and 128 neurons, followed by a fully connected layer with ReLU activation and a final softmax layer to obtain the hazard probabilities.

During the training process, we marked y_t as positive if any hazard occurred within a time window (e.g., the duration of insulin action for APS) after sequential execution of U_t. We labeled a simulation data trace as hazardous if any sample within it was unsafe. For a specific patient or autonomous vehicle, we trained the model on eighty percent of the data traces while keeping the time sequence of samples within the data trace, with a validation split rate of 0.1, and kept the remaining twenty percent of the data set untouched for testing. We used 4-fold cross-validation to evaluate the overall performance of the final ML model.

Furthermore, considering its advantage in capturing the temporal connection of time-series data, we trained an LSTM model as a baseline monitor using input data $X_t = \{x_{t-k+1}, ..., x_{t-1}, x_t\}$ and $U_t = \{u_{t-k+1}, ..., u_{t-1}, u_t\}$ with a sliding time-window of k:

$$y_t = p(\exists t' \in [t, t+T] : x_{t'} \in \mathcal{X}_h | X_t, U_t) \quad (13)$$

We explored different model architectures, and the best model we obtained was a two-layer (128-64 units) stacked LSTM with input time steps of 30 minutes and 1 second for APS and ADS, respectively. We trained both the LSTM and MLP models using the Adam [92] optimizer with the sparse categorical cross-entropy loss function and a learning rate of 0.001. We also added a dropout layer and early stopping on a held-out validation set to avoid over-fitting.

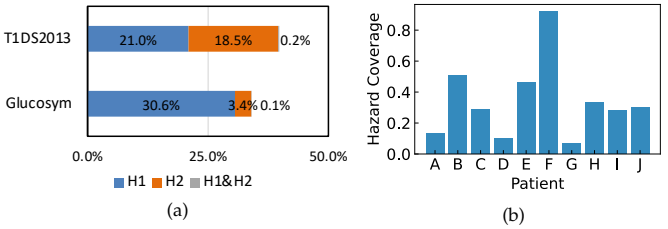
5.3.4 Other Baseline Monitors

In order to evaluate the impact of scenario-specific adversarial training of the monitor, we also designed three context-aware baseline monitors that implemented the generated SCS STL logic (same logic used for the proposed context-aware monitors), but: (1) without refining the thresholds (referred to as the CAWOT monitor), (2) with the thresholds learned from the fault-free data set, and (3) with the thresholds learned from all the populations' faulty data.

5.4 Results

5.4.1 Resilience of Baseline Systems without Monitors

We first analyzed the resilience of the baseline OpenPilot and OpenAPS control software, which are already designed with safety features (e.g., forward collision warning [91] or a



(a) (b)

Fig. 6: (a) Hazard Coverage of Each Hazard Type for APS; (b) Hazard Coverage of Each Patient.

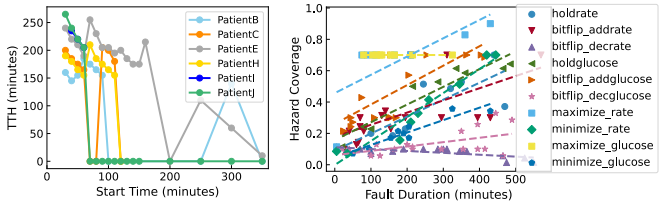


Fig. 8: (a) TTH vs. Start Time of Faults; (b) Relation between Fault Duration and Hazard Coverage of Different Scenarios/Fault Types. (Dotted line in the graph represents the best fit line of each situation).

maximum threshold and an auto-adjusted control algorithm [93]), in the presence of faults without any safety monitors.

Effectiveness of FI: We defined *Hazard Coverage* as the conditional probability that a given safety-critical fault injected into the system leads to an unsafe system state or a hazard. Experimental results showed that our FI could achieve an overall 33.9% hazard coverage on the Glucosym simulator, 39.3% on the T1DS2013 simulator, and 39.9% on the OpenPilot, which reflects our FI engine's efficiency in introducing faulty data for adversarial training as well as the inadequacy of the control software in tolerating safety-critical faults and attacks. Fig. 6 shows that FI covered all the hazard types, however, the majority of FI experiments in the Glucosym simulator led to the hazard type H1, increasing the risk of hypoglycemia. Also, the hazard coverage was quite different across different patient profiles, ranging from 6.7% to 92.4% across ten patients, suggesting that it may be essential to specify patient-specific and context-dependent safety requirements when designing monitors.

System Resilience: We evaluated the resilience of OpenPilot and OpenAPS using the Time-to-Hazard (TTH) metric that measures the time between the activation of a fault and the occurrence of a hazard (Fig. 7) to help specify time requirements for hazard prediction and mitigation.

Fig. 7 shows an average TTH of about 3 hours based on all the simulation data from OpenAPS. It should be noted that the human body has a considerable lag and is a slow dynamic system, so it usually takes hours for the BG to transmit into the vessel and for insulin to take effect. Moreover, 7.1% of hazardous simulations had TTH less than zero, which means that the hazards happened even before the injection of any faults to the controller, indicating the inadequacy of the APS control algorithm.

On the other hand, OpenPilot is a much faster control system with an average TTH of 6.4 seconds. So, a different length of control action sequence should be considered for safety monitoring. We do not observe as many simulations with negative TTH as in the APS case study. Comparatively, the APS has a stricter hazard labeling method due to the complexity and dynamics of patient bodies. We further analyzed the relationship between TTH and the start time

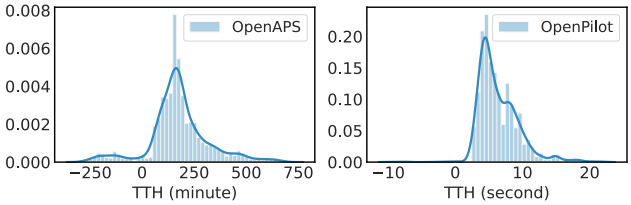


Fig. 7: Time to Hazard (TTH) Distribution.

of the faults and presented an example for APS in Fig. 8a. We see that, in general, TTH decreased with the delay of fault activation time. In addition, the fault being activated in the middle of the simulation did not have any hazardous impacts since the system might have entered a stable state. We also observe that the TTH increased with the delay of the fault start time in some cases. This might be because the insulin rate dropped with time and needed to accumulate before becoming hazardous to cause a hazard. Activating faults at the end of a simulation may cause hazards for some patients since the BG values were low after sleeping at night. Although we activated the faults randomly in our experiments, further exploration of the period of time needed to activate a fault could improve the effectiveness of fault injection.

Fault Duration: For APS, we also analyzed the relationship between the overall hazard coverage (averaged across all the patients) and fault types and fault duration in Fig. 8b. We observe that the hazard coverage increases with the increase in fault duration for most fault types, indicating that the faults need to persist for a period of time to cause hazards. This motivated us to refine the design of the safety monitor based on a sequence of control actions. In contrast, the hazard coverage for the fault type *max_glucose* was always high regardless of increasing the fault duration, which might be due to having reached the maximum glucose value. We also observe a drop in hazard coverage with the increase in fault duration for the fault type of *bitflip_decrate* that led to hazard in only one patient. This patient had an even more significant number of hazards without any fault injections, which indicates that decreasing insulin might be helpful to prevent hazards for this patient.

5.4.2 Monitor Prediction Accuracy

To evaluate the performance of different monitors in accurate prediction of hazards, we used false-positive rate (FPR), false-negative rate (FNR), accuracy (ACC), and F1 score metrics, calculated using two modified approaches appropriate for prediction based on sequential data, including *Sample Level with Tolerance Window* and *Simulation Level with Two Regions* [38]. Using the *Sample Level with Tolerance Window* approach, any alarms generated within a time window δ before the start time of hazard (t_h) are considered to be true positive (TP) (Fig. 9), because it is desirable that the monitor generates alerts before a hazard happens. Using the *Simulation Level with Two Regions* metric [38], we consider the whole data trace of a simulation as a single case, divide the data trace into two regions based on the time of activation of a fault (t_f) ($[0, t_f]$ and $[t_f, t_e]$ in Fig. 9), and then calculate the classification metrics separately for

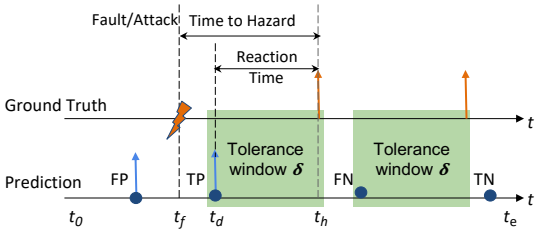


Fig. 9: Hazard Prediction Accuracy with Tolerance Window δ Marked with Green Area (with t_f, t_d, T_h representing the time when a fault/attack is activated, is detected by the monitor, and leads to an hazard, respectively).

each region. A TP is declared whenever an alert is generated during a hazardous data trace, regardless of when hazards happen for both regions.

Context-Aware Monitors vs. Non-ML Monitors: Table 3 presents the average performance of the CAWT monitor over all the patients/fault scenarios in comparison to the non-ML-based baseline monitors, Guidelines and MPC.

For APS, the CAWT monitor achieved the best performance in both Glucosym and T1DS2013 simulators. Although the Guideline monitor had a slightly lower FNR than the CAWT monitor in the T1DS2013 simulator, it generated more false alarms and had a 22.7% lower F1 score.

For ADS, the MPC monitor failed to generate alerts for 90% of the hazards, showing the weakness of the integrated FCW safety mechanism to attacks. On the other hand, the CAWT monitor still held a stable performance in accurately predicting hazards with up to 4.7 times improvement in average F1 score.

Context-Aware Monitors vs. Baseline ML Monitors: The overall performance of the Context-Aware monitors compared to two ML-based monitors in faulty scenarios (8820 simulations on each of the APS simulators and 4800 on the OpenPilot simulator) is shown in Table 4. For developing the ML-Custom monitors, the baseline LSTM and MLP monitors were upgraded with the proposed custom loss function (shown in Eq. 6).

For ADS, the LSTM monitor outperformed both the MLP and the Context-Aware monitors. However, the CAWT monitor demonstrated a comparable F1 score (using *Simulation Level with Two Regions metric*) to the LSTM monitor through a more straightforward and transparent model. Additionally, by adjusting the length of the control action sequence considered at each time step, the CAWT monitor can achieve even a higher F1 score and accuracy than the LSTM monitor. We will discuss this further in Section 6.3.

For two case studies of APS, we observe that the CAWT monitor outperformed other baseline ML monitors in the Glucosym and T1DS simulators using both sample level and simulation level metrics with a 7.9%-36.6% improvement in F1 score while keeping both FNR and FPR low. Also, the ML-Custom monitors performed the best using the *Simulation Level with Two Regions metric*.

The MLP-Custom monitor achieved a 1.9% and 16.2% improvement in F1 score and 28.6% and 38.6% reduction in FNR while keeping FPR low for the Glucosym and T1DS2013 simulators, respectively. We observed a similar improvement in the overall performance of the upgraded LSTM monitors (LSTM-Custom) for both the Glucosym and T1DS2013 simulators using at least one prediction accuracy metric. For the OpenPilot case study, we did not observe a similar improvement in the overall performance of both newly designed ML-Custom monitors. This is because the unrefined safety specifications did not do well in inferring the unknown system context. Nevertheless, the SCS STL formulas improved the transparency of the black-box ML models and can serve as a way to verify the learned neural network models.

Overall, the context-aware monitors achieve better performance than the baseline ML monitors, indicating the advantage of combining domain knowledge with machine learning for designing safety monitors. CAWT monitors achieve higher F1 scores using sample level metrics, while ML-Custom monitors perform better based on simulation level metrics. A more detailed discussion and comparison of these methods are provided in Section 6.2.

Context-Aware Monitors vs. Other Baselines: We further evaluated the CAWT monitor's performance without refining thresholds (CAWOT), with the thresholds learned from all the patients' data traces with and without fault injection, the patient-specific thresholds learned from each patient's faulty data traces, and the population-based thresholds learned from all the patients' erroneous data. For the population-based model, we learned the thresholds from the data of seventy percent of patients randomly selected from the population and tested the model on the data from the remaining thirty percent of the patients.

Table 3 shows that without learning the refined threshold of SCS rules, the CAWOT monitor suffered an 8.2%-32.0% drop in F1 score, reflecting the importance of optimizing safety requirements. But it still outperformed the MPC and Guideline monitors in the Glucosym simulator, demonstrating the benefit of context-awareness.

As shown in Table 5, using the thresholds learned from fault-free data, the context-aware monitor detected the unsafe control actions before the hazard happened 95.1% of

TABLE 3: Performance of CAWT Monitor vs. Non-ML Monitors

Simulator	Monitor	No. Sim.	Hazard%	FPR	FNR	ACC	F1 Score
Glucosym	Guideline	8820	33.9%	0.02	0.32	0.95	0.72
	MPC	8820	33.9%	0.02	0.34	0.95	0.71
	CAWOT	8820	33.9%	**0.01**	0.30	0.96	0.81
	CAWT	8820	33.9%	**0.01**	**0.02**	**0.99**	**0.96**
T1DS2013	Guideline	8820	39.3%	0.07	**<0.01**	0.93	0.75
	MPC	8820	39.3%	**<0.01**	0.02	**1.00**	0.96
	CAWOT	8820	39.3%	0.02	0.04	0.98	0.89
	CAWT	8820	39.3%	**<0.01**	0.03	**1.00**	**0.97**
OpenPilot	MPC	4800	39.9%	0.01	0.90	0.79	0.17
	CAWOT	4800	39.9%	0.29	0.12	0.76	0.66
	CAWT	4800	39.9%	**<0.01**	**0.05**	**0.99**	**0.97**

TABLE 4: Performance of Context-Aware (CAWT and ML-Custom) Monitors vs. ML-based Monitors

Simulator	Metric	Sample Level (Tolerance Window)				Simulation Level (Two Regions)			
	Monitor	FPR	FNR	ACC	F1 Score	FPR	FNR	ACC	F1 Score
Glucosym	MLP	0.02	0.07	0.97	0.89	0.13	0.06	0.89	0.79
	LSTM	0.04	0.06	0.96	0.81	0.16	0.06	0.87	0.78
	CAWT	0.01	**0.02**	**0.99**	**0.96**	0.10	**0.01**	0.92	0.86
	MLP-Custom	0.02	0.05	0.98	0.91	0.12	0.05	0.90	0.81
	LSTM-Custom	**0.00**	0.23	0.97	0.86	**0.03**	0.15	**0.94**	**0.87**
T1DS2013	MLP	<0.01	0.56	0.94	0.71	0.05	0.28	0.90	0.78
	LSTM	<0.01	0.06	0.99	0.95	0.08	0.06	0.93	0.87
	CAWT	<0.01	**0.03**	**1.00**	**0.97**	0.06	**0.02**	0.95	0.91
	MLP-Custom	0.01	0.27	0.96	0.82	0.10	0.18	0.88	0.78
	LSTM-Custom	**0.00**	0.17	0.98	0.90	**0.02**	0.10	**0.96**	**0.92**
Open Pilot	MLP	0.01	0.11	0.97	0.93	0.06	0.09	0.94	0.88
	LSTM	0.01	<0.01	**1.0**	**0.99**	0.05	<0.01	**0.96**	**0.93**
	CAWT	**<0.01**	0.05	0.99	0.97	**0.04**	0.05	**0.96**	**0.93**
	MLP-Custom	0.01	0.19	0.96	0.88	0.06	0.15	0.91	0.84
	LSTM-Custom	0.03	**0.00**	0.98	0.95	0.18	**0.00**	0.87	0.80

TABLE 5: Performance of Context-aware Monitor Using Thresholds Learned from Different Data Traces in APS

Threshold	FPR	FNR	ACC	F1 Score	EDR
Fault-free	0.01	0.27	0.96	0.83	95.1%
Faulty	0.01	**0.02**	**0.99**	**0.96**	**99.2%**
Population-based	0.08	0.08	0.92	0.89	92.2%
Patient-specific	**0.01**	**0.00**	**0.99**	**0.96**	**100.0%**

the time (for the true-positive cases) and failed to generate an alert for a hazardous situation in 27% of the simulations. Adversarial training and refinement of SCS formulas with the faulty data improved the monitor's performance with 4.1% in early detection rate (*EDR*) and 15.7% in F1 score.

We also observe that the context-aware monitor with patient-specific thresholds held an advantage over a population-based monitor with a 7.6% and 7.8% increase in accuracy and EDR, respectively. Moreover, the patient-specific context-aware monitor kept both FPR and FNR low, and therefore, achieved a 7.9% higher F1 score.

5.4.3 Monitor Timeliness

Fig. 10 shows the reaction time (defined in Section 2 and Fig. 9) for each monitor. We observe that:

- The CAWT monitor consistently performed well on ensuring safe reaction time for all the simulators. For APS, we have an average reaction time of about 100 minutes, which is larger than insulin activity peaks between 60 and 90 minutes [94]. For ADS, the average reaction time matches the safe headway time of 2 to 3 seconds [55].
- The non-ML baselines had the worst performance in timely detection of unsafe control actions. This might be because the baseline monitors were designed with fixed thresholds and could not do well across different patients/scenarios. Moreover, the MPC monitor could only predict hazards in a short window ahead of time in the Glucosym and T1DS2013 simulators and had a negative average reaction time in the OpenPilot, which indicates late detection and no possibility of stopping the potential hazards.
- Benefiting from a large amount of collected data and scenario-specific models, the baseline ML monitors performed better than the Guideline and MPC monitors. However, the performance of the baseline ML monitors varied a lot (with large standard deviations) and was not as stable as the CAWT monitor.
- The ML-Custom monitors achieved more stable performance, indicating the advantage of combining safety context with ML models. However, these ML-based monitors still have a more complex architecture than the CAWT monitor.

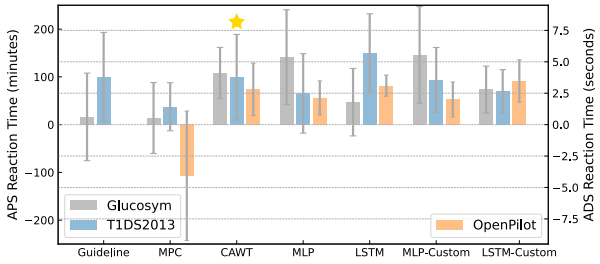


Fig. 10: Average Reaction Time for Each Monitor.

TABLE 6: Summary of Reaction Time Estimator Performance

Application	Predict Error (Avg ± Std)	Predict Error (Minimum)	Accuracy* (Percentage)	FPR Reduction of CAWT Monitor
APS	45.3 ± 23.6 min	1.7 min	74.5%	61.8%
ADS	0.12 ± 0.05 s	0.036 s	88.1%	61.3%

* calculated using $1-|\hat{t}_{rt}-t_{rt}|*100\%/t_{rt}$ and then averaged over all the samples, where $\hat{t}_{rt}/t_{rt}$ is the predicted/actual reaction time.

5.4.4 Reaction Time Estimator

We evaluated the performance of the reaction time estimator by comparing its predictions with the ground truth reaction times calculated from the labeled data. Table 6 shows that for ADS the reaction time predictor achieved an average prediction error of 0.12 seconds and a minimum error of 35.6 ms. For APS, the minimum prediction error was 1.7 minutes, with an average of 45.3 minutes across all the patients. We can see that the reaction time estimator performed worse in APS than ADS, which might be due to the complex physiological dynamics and the patients' unpredictable behavior. However, in both case studies, it offered an estimation of the maximum time within which a recovery action has to be issued to prevent potential hazards.

The reaction time estimator predicts the time left until the occurrence of a future hazard when the safety monitor generates an alert. So, it could work together with the safety monitor to identify the recovery requirements and detect any potential false alarms. For example, experimental results show that after filtering the alarms with estimated reaction times that exceeded a threshold (e.g., mean TTH), the CAWT monitor achieves better performance with 44.3%, 61.8%, and 61.3% reduction of FPR for T1DS2013, Glucosym, and OpenPilot, respectively.

5.4.5 Adaptive Adversaries

A common challenge in anomaly detection is the existence of adaptive adversaries that use the knowledge of the existing safety mechanisms to adjust the attack parameters to evade detection and cause adverse events [85], [86], [95]. To test the efficiency of our proposed safety monitoring approach against such stealthy attacks, we consider a very powerful attacker with the knowledge of (1) the logic of our safety monitor, (2) the parameters (e.g., β_i in Table 1), and (3) the control command format.

In our implementation, to evade detection, the stealthy attacks are only launched when the target monitor is not triggered to check the possibility of unsafe control actions. More specifically, the malicious changes to the controller state variables are still injected at random start times, but only remain active when none of the safety context conditions specified for the monitor (in the third column of Table 1) are held true. For example, to cause a forward collision with the lead vehicle, the attacker can keep accelerating the Ego vehicle until *Headway Time* (HWT) reaches an unsafe threshold. To avoid being detected by the context-aware monitor, the Ego vehicle is required to decelerate until HWT goes back to a safe range (e.g., β_{21}-β_{26} in Table 1) or until the ego vehicle is slower than the lead vehicle (e.g., RS<0), which will not trigger the context condition for the monitor.

Our experiments cover three types of stealthy attacks [14] by injecting the attack scenarios in Table 2 as follows: (1) surge attacks that maximize the attack impact as soon as

possible by maximizing the attack value (scenario *Max*), (2) bias attacks that maximize the time for the attack remaining undetected by minimizing the attack value (scenario *Min*), and (3) random attacks that randomly select attack value (scenario *Add/Sub*). We tested these scenarios with, respectively, 8820, 8820, and 4800 simulations in Glucosym, T1DS, and OpenPilot simulators. Experimental results show that the stealthy attacks targeted at the context-aware monitors with refined thresholds (CAWT) do not cause any hazards because of not triggering any of the safety context conditions.

However, the success of the stealthy attacks in causing hazards and the efficiency of the CAWT monitor against adaptive adversaries depends on the completeness and accuracy of the generated SCS and the monitor parameters (further discussed in *Section 7*). Taking ADS as an example, Fig. 11 shows the effect of thresholds used for the CAWT monitor on the success rate of the stealthy attacks. Specifically, we see the percentage of simulations across different types of stealthy attacks in which a hazard happened while remaining undetected by the monitor for different values of the safety thresholds (β_{21}-β26 in Table 1) for parameter HWT. The attack success rate decreases with the increase of HWT thresholds, and no hazard happens anymore when the thresholds are larger than 1 second (the thresholds we learned for HWT to develop CAWT monitors are between 2-3 seconds). The same effect might be observed, if the attacker can identify any additional safety context conditions under which hazards can happen (additional rows in Table 1) but were not used in the design of the target monitor.

5.4.6 Evaluation on a Clinical Trial Dataset

To further evaluate the effectiveness of the proposed safety monitor in real applications with realistic data, we tested the performance of the proposed monitors using one of the largest publicly-available diabetic datasets, called DCLP3 [96]. This dataset was collected from a clinical trial of the only FDA-approved closed-loop APS, t:slim X2 with Control-IQ Technology, for the six-month treatment of 168 diabetic patients aged 14 to 71 years old. We break down each patient's data into 180 days and use 150 iterations of the glucose readings and insulin records each day, resulting in 30,240 days of data and 4,536,00 samples (30,240 *150) in total. We labeled the hazard events in the dataset using the same risk index approach shown in Section 4.2. For a specific patient, we train the model or learn STL parameters on eighty percent of the data traces while keeping the time sequence of samples within the data trace (with a validation split rate of 0.1 for ML model training) and keep the remaining twenty percent of the data set untouched for testing. We used 4-fold cross-validation to evaluate the overall performance of the final safety monitor. Experimental results show that the context-aware monitors developed using either the CAWT or ML-Custom methods achieve F1 scores of 0.86 and 0.85, respectively. In addition, our context-aware monitors can predict the actual adverse events seen in the data, including both the CGM-measured hyperglycemic events[1] and the CGM-measured hypoglycemic events[2] [96], with a success rate of 99.5%.

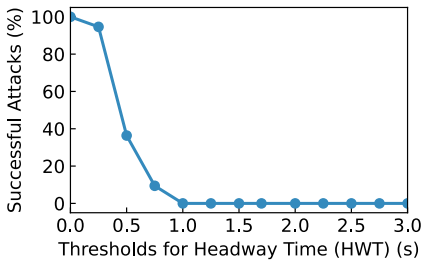


Fig. 11: Stealthy attack success rate for different settings of the thresholds (β_{21}-β_{26} in Table 1) for the monitor parameter *Headway Time (HWT)*.

5.4.7 Resource Utilization

We ran the simulations with the different safety monitors and without any monitors a thousand times and calculated the average time overhead for each safety monitor. Results showed that the CAWT monitor has the lowest average time overhead of 15.8 μs and 2.6 μs for APS and ADS, respectively, among all the safety monitors. In contrast, the Guideline monitor's time overhead was 3.1 ms, and the time overhead for the MPC monitor, the MLP monitor, and the LSTM monitor was 104.4 μs/25.1 μs, 15.9 ms/15.7 ms, and 18.1 ms/18.2 ms for APS/ADS, respectively. Since the ML-Custom monitors have the same architecture as the MLP and LSTM monitors and only use a different offline training procedure, they do not add any additional run-time overhead compared to the MLP and LSTM monitors.

6 Discussion

Our experiments provided the following key insights.

6.1 Both the OpenPilot and OpenAPS control software cannot tolerate safety-critical faults.

Although OpenPilot is an advanced control system widely used in actual road driving and has an integrated forward collision warning function, and OpenAPS is a fully automated system already equipped with some safety features, they failed to tolerate the simulated attacks and faults. In 13.8% of the APS simulations, patients were predicted to suffer from severe hypoglycemia. Also, in 81% of the simulated driving scenarios, OpenPilot failed to generate alerts before a collision occurred. Furthermore, some hazards occurred even without any fault injections.

6.2 Hybrid knowledge and data-driven context-aware monitors outperform solely knowledge-based and ML-based monitors.

We proposed two approaches to integrating domain knowledge into a safety monitor by either (1) optimizing STL safety specifications with data traces collected from the closed-loop cyber-physical system (CAWT), or (2) training an ML model under the guidance of a custom loss function (ML-Custom) (see Eq. 6) that enforces satisfying the SCS STL formulas.

Our experimental evaluation of the context-aware safety monitors (in Section 5.4) showed the benefit of integrating

1. A period of at least 15 consecutive minutes with BG $< 54\ mg/dL$
2. A period of at least 120 consecutive minutes with BG $> 300\ mg/dL$

domain knowledge with data-driven techniques with better performance than several baselines designed with the state-of-the-art techniques using solely the domain knowledge (including medical guidelines and dynamic models) or data (ML optimization) in most situations.

The advantages of the STL learning method over the ML optimization with custom loss function include: (1) it can also work well with an unbalanced dataset or when negative examples are unavailable or expensive to collect, (2) it requires less computational resources at run-time, and (3) it can be easily verified because of having a more transparent architecture and less overhead. However, its performance heavily relies on the complete generation of SCS rules, which might be challenging to achieve for more complex case studies. Therefore, this method is suggested for situations where negative examples are unavailable, or there are strict requirements for transparency, safety, or run-time resources.

On the other hand, the ML optimization method can achieve about the same performance as the baseline ML monitors even without the complete specification of SCS rules and is easier to implement. It also achieves slightly better performance than the monitor developed using the STL optimization method (see APS study case in Table 4) but requires a relatively balanced dataset for training and more computational resources at run-time. Therefore, it is preferred to use this method when there are enough balanced training datasets and resources available.

6.3 By considering the sequence of control actions, the CAWT monitors can generate more accurate alerts.

Fig. 12 presents the performance of the CAWT and MLP monitors that considered either a sequence of control actions or a single control action in comparison to the LSTM monitor. Both the MLP and CAWT monitors that take into account a sequence of control actions achieved a lower FPR and a higher F1 score, with a slightly higher FNR, than the same monitors that used only a single control action for both the Glucosym and T1DS2013 simulators. Additionally, after considering a sequence of control actions, the MLP monitor achieved a slightly better performance than the LSTM monitor in the Glucosym simulator and reduced the difference in performance with the LSTM monitor in the T1DS2013 simulator. However, for ADS, which is a much faster control system than APS, the CAWT monitor based on a single control action achieved better performance, indicating that control action sequence length is an important consideration in designing safety monitors for different CPS.

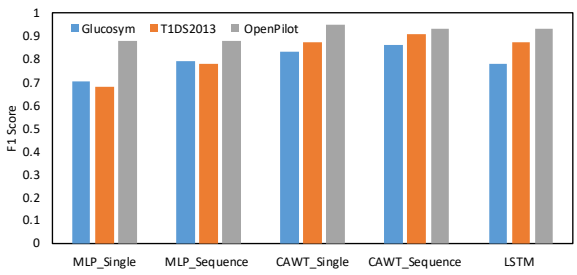


Fig. 12: Performance of Each Monitor based on a Single Control Action or a Sequence of Control Actions.

6.4 Adversarial training using scenario-specific data improves the performance of CAWT monitors.

As shown in Section 5.4.2 the CAWT monitor with thresholds learned from faulty data of a specific patient outperformed other context-aware monitors that were designed with the exact SCS logic but with different thresholds. These results reconfirm that adversarial training and refinement of SCS formulas using the faulty data is important in improving the CAWT monitor's performance. Furthermore, each patient has different biomedical characteristics and tolerance levels to the injected insulin amounts. Thus, the safety monitor logic needs to be refined for each patient or scenario.

7 Threats to Validity

Sensor Perturbations: This paper mainly focuses on the faults and attacks targeting the control software of safety-critical CPS. Any perturbations in the sensor data will potentially affect both the controller and the safety monitor's behavior. However, several existing methods in the literature [14], [17], [80], [97] can be integrated with our safety monitor to protect the sensor data and actuator commands observed by the monitor. Furthermore, slight disturbances in sensor data brought by environment noise can be constrained by the typical robustness features of the control system (e.g., use of PID algorithm) to not comprise the control actions. Also, combing domain knowledge with data could improve the monitor's robustness against small accidental or malicious perturbations in the input data [98].

Data Impact: The performance degradation in predicting unseen data/scenarios or corner cases is a common problem for data-driven methods. We try to address this problem by integrating domain knowledge with data and using adversarial training. The integrated domain knowledge is extracted based on a high-level control-theoretic hazard analysis method which is independent of the attack types. However, the tightness of the learned thresholds may be affected by the variances in the data. Adversarial training is one of the most effective approaches to defending against adversarial examples in machine learning. However, it is essential to include adversarial examples produced by all known attack scenarios for anomaly detection [99]. We tried to include the most common types of attack scenarios reported in the literature and real databases for adversarial training. Besides, online learning or transfer learning techniques could be utilized to apply the models trained using simulated fault/attack scenarios to real-world systems. However, transferring patient-specific models from simulation to reality might be challenging in the case of MCPS. Potential solutions might include either estimating the target individual patient profile for model training through a system identification method or physical exams or relying on fault-free clinical or synthetic data. Our experiments (see Table 5) show that population-based fault-free models can achieve reasonable performance but are not comparable to patient-specific models trained with both faulty and fault-free data.

SCS Completeness: The proposed monitor's performance heavily relies on the accuracy and completeness of the generated SCSs, which might not be easy to derive for

highly complex systems. However, our method only uses a subset of state variables that can fully represent the system's dynamics. In addition, inaccurate or incomplete specification is a common problem in the design and verification of any controller/monitor. We try to reduce such manual errors by proposing a formal framework for designers, in collaboration with domain experts, to generate safety context specifications (SCS).

False Alarms: Both false negative and false positive detections by the monitor might impose additional risks. In case of false positives (although as low as 1% as shown in the results), the follow-up mitigation actions, depending on whether manual or automated, might lead to new types of hazards and potential accidents. In this paper, we assume that the monitor generates an alert to warn the system users to manually suspend the potential unsafe control commands. For example, in the APS, the patient will be required to check the control commands after an alert, and in the ADS, the driver will be warned to take over control of the vehicle. So the false positives will not cause any hazards themselves but may bother the system users.

Sim-to-Real Gap: The use of simulation is essential for enabling research progress at an accelerated rate while avoiding/reducing unnecessary risks of testing in the actual operating environments and harming real patients, drivers, and pedestrians. However, the differences between the simulation and the actual implementation in the real world might threaten the validity of the proposed method. For example, in the APS simulations, we only simulate a patient going to sleep after dinner without considering other physical activities or multiple meal consumption. We try to reduce the sim-to-real gap by using real control software (used with actual diabetic patients [100], [101] or on actual vehicles [70]) and real patient simulators that are either approved by the FDA for clinical testing or use actual patient profiles [64], [65] as well as realistic high-risk driving scenarios defined by NHTSA. Furthermore, a test using a publicly-available dataset collected from a clinical trial [96] is conducted to attest to the effectiveness/validity of the proposed approach [69].

8 Related Work

8.1 Anomaly Detection in CPS

Previous efforts on anomaly detection mainly focused on safety-critical attacks targeting sensor data [14], [18], [102], [103]. Less attention has been paid to detecting attacks that compromise the controller functionality by directly manipulating the controller internal logic and variables [104]. Further, the existing methods either rely on simple linear models of physical systems which cannot fully capture system dynamics [11], [12] or the black-box ML models [16], [17] that suffer from a lack of generalization and transparency. Our work distinguishes from these previous works by detecting the faults/attacks that affect the controller internal variables and output and introducing the notion of preemptive detection of early signs of hazards instead of detecting them after they have occurred, which might be too late for successful mitigation and recovery.

8.2 Run-time Safety Assurance

Recent works on run-time safety assurance have focused on protecting CPS against catastrophic failures by checking predefined set of safety properties (or contracts) and instantaneous reaction [105]–[107] and developing ML-based [19], [108] or linear approximation methods [13] for automated recovery. Others proposed techniques for switching to alternative controllers when the original controller is compromised but with the limitations of either sacrificing overall performance [109] or not being suitable for embedded systems with limited resources [18]. In this work, we propose a formal framework and a hybrid knowledge and data-driven approach for the synthesis of monitor logic that is simple and transparent and can be integrated with an existing controller interface for hazard prediction and mitigation.

8.3 Knowledge and Data Integration

Several studies have focused on integrating knowledge with data using STL learning. [47] provided efficient methods for automatically mining and learning STL properties from measured data. [110] presented an approach for learning optimized STL properties using positive examples. [111] applied STL learning and monitoring to anomaly detection in CPS. However, most existing works rely on the ad-hoc specification of STL properties rather than principled methods. Others have proposed adding constraints or using customized policies during the ML training process to achieve better performance. [53] designed a semantic loss function for deep learning with symbolic knowledge. [112] presented a technique for embedding a set of logic rules into the Recurrent Neural Networks using feedback masks. [113] proposed a method for using the policy of a reinforcement learning agent to train smaller and more efficient networks with improved performance. Our previous work [38] proposed a formal framework for the design and synthesis of safety monitors in APS based on STL learning and control-theoretic hazard analysis. This paper extends [38] by (i) combining STL safety requirements with customized optimization and machine learning to design and synthesize context-aware safety monitors that predict hazards and (ii) generalizing the proposed approach to other CPS by adding a case study of ADS.

9 Conclusion

This paper presented a formal framework for the hybrid knowledge and data-driven design of context-aware safety monitors that can predict and mitigate hazards in CPS. We evaluated the performance of the proposed method using two simulated closed-loop CPS for diabetics treatment (APS) and autonomous driving (ADS) as well as a clinical trial dataset. Experimental results demonstrate that the proposed context-aware monitors outperform several baselines developed using medical guidelines, MPC, and ML in the accurate prediction of hazards while ensuring sufficient reaction time for potential hazard mitigation. The stable and improved performance of the context-aware monitors in two different CPS case studies indicates the generalizability of our proposed approach. Future work will focus on extending the proposed approach to context-specific mitigation and recovery.

ACKNOWLEDGMENT

This material is based upon work partially supported by the Commonwealth of Virginia under Grant CoVA CCI: C-Q122-WM-02 and by the National Science Foundation (NSF) under Grant No. 1748737 and 2146295.

REFERENCES

[1] U.S. Food and Drug Administration, "Class 2 device recall medtronic minimed paradigm veo insulin pump," https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfres/res.cfm?id=175809, March 26, 2020.

[2] K. Kim, J. S. Kim, S. Jeong *et al.*, "Cybersecurity for autonomous vehicles: Review of attacks and defense," *Computers & Security*, vol. 103, p. 102150, 2021.

[3] Keen Security Lab of Tencent, "New car hacking research: 2017, remote attack tesla motors again." 2017. [Online]. Available: https://keenlab.tencent.com/en/2017/07/27/New-Car-Hacking-Research-2017-Remote-Attack-Tesla-Motors-Again/

[4] Z. Chris, "A google self-driving car caused a crash for the first time," *The Verge*, 2016.

[5] A. Rao, N. Carreon, R. Lysecky *et al.*, "Probabilistic threat detection for risk management in cyber-physical medical systems," *IEEE Software*, vol. 35, no. 1, pp. 38–43, 2017.

[6] N. Leveson and J. Thomas, "An stpa primer," *Cambridge, MA*, 2013.

[7] J. Wan, A. Canedo, and M. A. Al Faruque, "Functional model-based design methodology for automotive cyber-physical systems," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2028–2039, 2017.

[8] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming dr. frankenstein: Contract-based design for cyber-physical systems," *European Journal of Control*, vol. 18, no. 3, 2012.

[9] F. Cairoli, G. Fenu, F. A. Pellegrino *et al.*, "Model predictive control of glucose concentration based on signal temporal logic specifications," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019, pp. 714–719.

[10] H. Alemzadeh, R. K. Iyer, Z. Kalbarczyk *et al.*, "Analysis of safety-critical computer failures in medical devices," *IEEE Security & Privacy*, vol. 11, no. 4, pp. 14–26, 2013.

[11] H. Lin, H. Alemzadeh, and C. Daniel et al., "Safety-critical cyber-physical attacks: Analysis, detection, and mitigation," *HOTSOS*, 2016.

[12] H. Lin, H. Alemzadeh, Z. Kalbarczyk *et al.*, "Challenges and opportunities in the detection of safety-critical cyberphysical attacks," *Computer*, vol. 53, no. 3, pp. 26–37, 2020.

[13] L. Zhang, X. Chen, F. Kong *et al.*, "Real-time attack-recovery for cyber-physical systems using linear approximations," in *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 205–217.

[14] A. A. Cárdenas, S. Amin, and Z.-S. Lin et al., "Attacks against process control systems: Risk assessment, detection, and response," in *ASIACCS*, 2011, p. 12.

[15] C. Hernandez and J. Abella, "Timely error detection for effective recovery in light-lockstep automotive systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1718–1729, 2015.

[16] Y. Luo, Y. Xiao, L. Cheng *et al.*, "Deep Learning-based Anomaly Detection in Cyber-physical Systems: Progress and Opportunities," *ACM Computing Surveys*, vol. 54, no. 5, May 2021.

[17] M. Yasar and H. Alemzadeh, "Real-time context-aware detection of unsafe events in robot-assisted surgery," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2020, pp. 385–397.

[18] H. Choi, S. Kate, Y. Aafer *et al.*, "Software-based realtime recovery from sensor attacks on robotic vehicles," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. USENIX Association, Oct. 2020, pp. 349–364.

[19] D. Pritam, L. Guanpeng, C. Zitao *et al.*, "Pid-piper: Recovering robotic vehicles from physical attacks," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021.

[20] A. Singla, E. Bertino, and D. Verma, "Overcoming the lack of labeled data: Training intrusion detection models using transfer learning," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2019, pp. 69–74.

[21] Y. Zhou and K. Murat, "On transparency of machine learning models: A position paper," in *AI for Social Good Workshop*, 2020.

[22] T. Ouyang, V. S. Marco, Y. Isobe *et al.*, "Corner case data description and detection," *arXiv*, 2021.

[23] K. R. Varshney and H. Alemzadeh, "On the safety of machine learning: Cyber-physical systems, decision sciences, and data products," *Big data*, vol. 5, no. 3, pp. 246–255, 2017.

[24] A. Mehmed, W. Steiner, and A. Causevic, "Formal verification of an approach for systematic false positive mitigation in safe automated driving system," April 2020.

[25] H. Choi, W.-C. Lee, Y. Aafer *et al.*, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 801–816.

[26] M. R. Aliabadi, A. A. Kamath, and J. Gascon-Samson et al., "ARTINALI: dynamic invariant detection for cyber-physical system security," in *ESEC/FSE 2017*. ACM, 2017, pp. 349–361.

[27] H. Alemzadeh, D. Chen, and X. Li et al., "Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation," in *2016 46th Annual IEEE/IFIP International Conference on DSN)*. IEEE, 2016, pp. 395–406.

[28] U.S. Food and Drug Administration, *Policy for Device Software Functions and Mobile Medical Applications*, https://www.fda.gov/media/80958/download, 2019.

[29] H. Krasner, "The cost of poor quality software in the us: A 2018 report," *Consortium for Information & Software Quality*, 2018.

[30] E. Chien, L. OMurchu, and N. Falliere, "W32.duqu: The precursor to the next stuxnet," in *5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 12)*, San Jose, CA, 2012.

[31] P. Sun, L. Garcia, and S. Zonouz, "Tell me more than just assembly! reversing cyber-physical execution semantics of embedded iot controller software binaries," in *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019.

[32] "Trusted computing base," https://en.wikipedia.org/wiki/Trusted_computing_base.

[33] "Arm trustzone technology," https://developer.arm.com/ip-products/security-ip/trustzone.

[34] N. Leveson, *Engineering a safer world: Systems thinking applied to safety*. MIT press, 2011.

[35] I. Lee, O. Sokolsky, S. Chen *et al.*, "Challenges and research directions in medical cyber-physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 75–90, 2012.

[36] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee *et al.*, "Ensuring safety, security, and sustainability of mission-critical cyber-physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 283–299, 2012.

[37] J. Wan, D. Zhang, S. Zhao *et al.*, "Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions," *IEEE Comm. Magazine*, vol. 52, no. 8, 2014.

[38] X. Zhou, B. Ahmed, J. H. Aylor *et al.*, "Data-driven design of context-aware monitors for hazard prediction in artificial pancreas systems," in *51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021, pp. 484–496.

[39] G. Sannino and G. D. Pietro, "A mobile system for real-time context-aware monitoring of patients' health and fainting," *International Journal of Data Mining and Bioinformatics*, vol. 10, no. 4, p. 407, 2014.

[40] R. Ivanov, J. Weimer, and I. Lee, "Context-aware detection in medical cyber-physical systems," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 2018.

[41] Virginia DMV, "Safe driving," https://www.dmv.virginia.gov/webdoc/pdf/dmv39d.pdf.

[42] A. Desai, S. Ghosh, S. A. Seshia *et al.*, "Soter: A runtime assurance framework for programming safe robotics systems," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 138–150.

[43] L. Zhang, X. Chen, F. Kong *et al.*, "Real-time attack-recovery for cyber-physical systems using linear approximations," in *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 205–217.

[44] "Safety envelope requirements approach," https://www.youtube.com/watch?v=_wxlgbl4s-g.

[45] J. Thomas, "Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis," *Technical Report SAND2012-4080*, 2012.

[46] P. Asare, "A Framework for Reasoning about Patient Safety of Emerging Computer-Based Medical Technologies," Ph.D. dissertation, University of Virginia, May 2015.

[47] E. Bartocci, "Monitoring, learning and control of cyber-physical systems with stl (tutorial)," in *Runtime Verification*, C. Colombo

and M. Leucker, Eds. Cham: Springer International Publishing, 2018, pp. 35–42.
[48] B. Zhang and W. Zuo, "Learning from positive and unlabeled examples: A survey," in *2008 International Symposiums on Information Processing*, May 2008, pp. 650–654.
[49] S. Jha and S. A. Seshia, "A theory of formal synthesis via inductive learning," *Acta Informatica*, vol. 54, no. 7, Nov 2017.
[50] S. Jha, A. Tiwari, and S. Seshia et al., "TeLEx: learning signal temporal logic from positive examples using tightness metric," *Formal Methods in System Design*, vol. 54, no. 3, pp. 364–387, 2019.
[51] J. L. Morales and J. Nocedal, "Remark on "algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound constrained optimization"," in *ACM Transactions on Mathematical Software*, vol. 38, 2011, pp. 1–4.
[52] J. Erway and R. F. Marcia, "Solving limited-memory bfgs systems with generalized diagonal updates," in *World Congress on Engineering*, 2012.
[53] J. Xu, Z. Zhang, T. Friedman *et al.*, "A semantic loss function for deep learning with symbolic knowledge," 2018.
[54] L. Marcovecchio, "Complications of acute and chronic hyperglycemia," *US Endocrinology*, vol. 13, no. 1, pp. 17–21, 2017.
[55] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Analysis Prevention*, vol. 35, 2003.
[56] W. Clarke and B. Kovatchev, "Statistical tools to analyze continuous glucose monitor data," *Diabetes technology & therapeutics*, vol. 11, no. S1, pp. S–45, 2009.
[57] B. P. Kovatchev, "Metrics for glycaemic control - from hba 1c to continuous glucose monitoring," *Nature Reviews Endocrinology*, vol. 13, no. 7, p. 425, 2017.
[58] B. P. Kovatchev, D. J. Cox, L. A. Gonder-Frederick *et al.*, "Assessment of risk for severe hypoglycemia among adults with iddm: validation of the low blood glucose index." *Diabetes care*, vol. 21, no. 11, pp. 1870–1875, 1998.
[59] A. H. M. Rubaiyat, Y. Qin, and H. Alemzadeh, "Experimental resilience assessment of an open-source driving agent," *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, Dec 2018.
[60] M. Shane, "Autonomous vehicle radar perception in 360 degrees," *Nvidia developer blog*, Nov 2019.
[61] "Principles of an open artificial pancreas system (openaps)," https://openaps.org/reference-design.
[62] E. Chertok Shacham, H. Kfir, N. Schwartz *et al.*, "Glycemic control with a basal-bolus insulin protocol in hospitalized diabetic patients treated with glucocorticoids: a retrospective cohort," *BMC Endocr Disord*, vol. 18(75), pp. 1472–6823, 2018.
[63] "Glucosym," https://github.com/Perceptus/GlucoSym.
[64] C. D. Man, F. Micheletto, D. Lv *et al.*, "The uva/padova type 1 diabetes simulator: new features," *Journal of diabetes science and technology*, vol. 8, no. 1, pp. 26–34, 2014.
[65] S. Kanderian, S. Weinzimer, and G. Voskanyan et al, "Identification of intraday metabolic profiles during closed-loop glucose control in individuals with type 1 diabetes." *Journal of diabetes science and technology*, vol. 3, no. 5, pp. 1047–1057, 2009.
[66] R. Visentin, C. Dalla Man, B. Kovatchev *et al.*, "The university of virginia/padova type 1 diabetes simulator matches the glucose traces of a clinical trial," *Diabetes technology & therapeutics*, vol. 16, no. 7, pp. 428–434, 2014.
[67] B. P. Kovatchev, M. Breton, C. D. Man *et al.*, "In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes," *Journal of Diabetes Science and Technology*, vol. 3, no. 1, pp. 44–55, 2009, pMID: 19444330.
[68] R. Hawkins, C. Paterson, C. Picardi *et al.*, "Guidance on the assurance of machine learning in autonomous systems (amlas)," *arXiv preprint arXiv:2102.01564*, 2021.
[69] X. Zhou, M. Kouzel, H. Ren *et al.*, "Design and validation of an open-source closed-loop testbed for artificial pancreas systems," *the IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, 2022.
[70] "openpilot," https://github.com/commaai/openpilot.
[71] "comma.ai," https://comma.ai/.
[72] C. Li, A. Raghunathan, and N. K. Jha, "Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system," in *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*. IEEE, 2011, pp. 150–156.
[73] C. M. Ramkissoon, B. Aufderheide, B. W. Bequette *et al.*, "A review of safety and hazards associated with the artificial pancreas," *IEEE reviews in biomedical engineering*, vol. 10, 2017.
[74] U.S. Food and Drug Administration, "Class 2 device recall minimed 640g insulin infusion pump," https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfres/res.cfm?id=181608, December 02, 2020.
[75] T. Bonaci, J. Yan, J. Herron *et al.*, "Experimental analysis of denial-of-service attacks on teleoperated robotic systems," in *ACM/IEEE 6th International Conference on Cyber-Physical Systems*, 04 2015.
[76] T. Bonaci, J. Herron, T. Yusuf *et al.*, "To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots," *arXiv Preprint arXiv:1504.04339*, 04 2015.
[77] S. Jha, S. Banerjee, T. Tsai *et al.*, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019, pp. 112–124.
[78] U.S. Food and Drug Administration. (2022) Maude adverse event report. [Online]. Available: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/Detail.cfm?MDRFOI__ID=2430675
[79] ——. (2022) Maude adverse event report. [Online]. Available: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/detail.cfm?mdrfoi__id=6113784&pc=LZG
[80] V. Sadhu, S. Zonouz, and D. Pompili, "On-board deep-learning-based unmanned aerial vehicle fault cause detection and identification," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 5255–5261.
[81] W. G. Najm, J. D. Smith, M. Yanagisawa *et al.*, "Pre-crash scenario typology for crash avoidance research," Tech. Rep. DOT-VNTSC-NHTSA-06-02, Apr. 2007.
[82] U.S. Food and Drug Administration, "Class 2 device recall medtronic minimed paradigm model mmt723k insulin pump," https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfRES/res.cfm?id=175210, March 16, 2021.
[83] K. Zetter, "It's insanely easy to hack hospital equipment," *Wired Magazine*, vol. 16, no. 1, pp. 303–336, 2014.
[84] K. Kim, J. S. Kim, S. Jeong *et al.*, "Cybersecurity for autonomous vehicles: Review of attacks and defense," *Computers & Security*, vol. 103, 2021.
[85] S. Kim, Y. Eun, and K.-J. Park, "Stealthy sensor attack detection and real-time performance recovery for resilient cps," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, 2021.
[86] G. Park, C. Lee, H. Shim *et al.*, "Stealthy adversaries against uncertain cyber-physical systems: Threat of robust zero-dynamics attack," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 4907–4919, 2019.
[87] W. Young, J. Corbett, and M. S. Gerber et al., "Damon: A data authenticity monitoring system for diabetes management," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018.
[88] F. Cairoli, G. Fenu, F. A. Pellegrino *et al.*, "Model predictive control of glucose concentration based on signal temporal logic specifications," in *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2019, pp. 714–719.
[89] V. Raman, A. DonzÃl', M. Maasoumy *et al.*, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.
[90] "longitudinal-maneuvers," https://github.com/commaai/openpilot/tree/master/selfdrive/test/longitudinal_maneuvers.
[91] comma ai, "Bringing forward collision warnings to our open source self-driving car," https://comma-ai.medium.com/bringing-forward-collision-warnings-to-our-open-source-self-driving-car-7545b6e398cd, 2018.
[92] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[93] "determine-basal," https://github.com/openaps/oref0/blob/master/lib/determine-basal/determine-basal.js.
[94] OpenAPS community. (2017) Understanding insulin on board (iob) calculations. [Online]. Available: https://openaps.readthedocs.io/en/latest/docs/While%20You%20Wait%20For%20Gear/understanding-insulin-on-board-calculations.html
[95] X. Zhou, A. Schmedding, H. Ren *et al.*, "Strategic safety-critical attacks against an advanced driver assistance system," *2022 52nt Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022.
[96] S. A. Brown, B. P. Kovatchev, D. Raghinaru *et al.*, "Six-month randomized, multicenter trial of closed-loop control in type 1 diabetes," *New England Journal of Medicine*, vol. 381, no. 18, 2019.

[97] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.
[98] X. Zhou, M. Kouzel, and H. Alemzadeh, "Robustness testing of data and knowledge driven anomaly detection in cyber-physical systems," *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop on Dependable and Secure Machine Learning (DSN-DSML)*, 2022.
[99] N. Papernot, P. McDaniel, A. Sinha *et al.*, "Sok: Towards the science of security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroSP)*, 2018.
[100] D. M. Lewis, "Do-it-yourself artificial pancreas system and the openaps movement," *Endocrinology and Metabolism Clinics*, vol. 49, no. 1, pp. 203–213, 2020.
[101] D. Lewis, S. Leibrand, and . O. Community, "Real-world use of open source artificial pancreas systems," *Journal of diabetes science and technology*, vol. 10, no. 6, pp. 1411–1411, 2016.
[102] J. Sun, Y. Cao, Q. A. Chen *et al.*, "Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2020.
[103] R. Quinonez, J. Giraldo, L. Salazar *et al.*, "SAVIOR: Securing autonomous vehicles with robust physical invariants," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020.
[104] A. Ding, P. Murthy, L. Garcia *et al.*, "Mini-me, you complete me! data-driven drone security via dnn-based approximate computing," in *24th International Symposium on Research in Attacks, Intrusions and Defenses*, 2021, pp. 428–441.
[105] M. Wu, H. Zeng, C. Wang *et al.*, "Safety guard: Runtime enforcement for safety-critical cyber-physical systems," in *54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017.
[106] M. Wu, J. Wang, J. Deshmukh *et al.*, "Shield synthesis for real: Enforcing safety in cyber-physical systems," in *2019 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2019, pp. 129–137.
[107] N. Arechiga, "Specifying safety of autonomous vehicles in signal temporal logic," in *IEEE Intelligent Vehicles Symposium (IV)*, 2019.
[108] C. Lazarus, J. G. Lopez, and M. J. Kochenderfer, "Runtime Safety Assurance Using Reinforcement Learning," *arXiv*, Oct. 2020.
[109] A. Desai, S. Ghosh, S. A. Seshia *et al.*, "Soter: Programming Safe Robotics System using Runtime Assurance," *arXiv*, 2019.
[110] S. Jha, A. Tiwari, S. Seshia *et al.*, "Telex: Passive stl learning using only positive examples," 09 2017, pp. 208–224.
[111] A. Jones, Z. Kong, and C. Belta, "Anomaly detection in cyber-physical systems: A formal methods approach," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 848–853.
[112] B. Chen, Z. Hao, X. Cai *et al.*, "Embedding logic rules into recurrent neural networks," *IEEE Access*, vol. 7, 2019.
[113] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre *et al.*, "Policy distillation," *arXiv*, 2016.

Xugui Zhou obtained his M.E. degree in Control Science and Engineering from Shandong University, China, in 2015. He is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering at the University of Virginia. Before joining Dependable Systems and Analytics Lab at UVa in January 2019, he was an Project Manager and Engineer at NR Research Institute, State Grid of China. His research interests are at the intersection of computer systems dependability and control system theory and engineering. He is particularly interested in engineering safer autonomous systems through resilience assessment, security validation, and artificial intelligence techniques.

Bulbul Ahmed received the B.S. degree in Electrical and Electronic Engineering from Rajshahi University of Engineering and Technology (RUET), Rajshahi, Bangladesh, and the M.S. degree in Computer Engineering from the University of Virginia, Charlottesville, VA. He is currently pursuing a Ph.D. degree in Electrical and Computer Engineering, University of Florida, Gainesville, FL. His current research interest includes Hardware Security, System-on-chip verification, and formal verification.

James H. Aylor received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Virginia, Charlottesville, Virginia in 1968, 1971, and 1977, respectively. Over the years, he has been an active researcher in the area of complex computer system design including computer technology for persons with disabilities. He has guided more than 45 graduate students in their master's and doctoral thesis work and published over 145 research papers. Dr. Aylor has also been extremely active in professional activities both in technical and administrative capacities. In 1993, he served as the President of the Institute of Electrical and Electronic Engineers (IEEE) Computer Society and from 1994-1996 he served as a Division Director (and a Board of Directors member) of IEEE. He is a Fellow of the IEEE and past Editor-in-Chief of IEEE Computer, the flagship magazine that is received by the more than 100,000 IEEE Computer Society members. He served as President of the ECEDHA (Electrical & Computer Engineering Department Heads Association).

Philip Asare received his BSE and MSE in Electrical Engineering from the University of Pennsylvania, and his Ph.D. in Computer Engineering from the University of Virginia. He is an Assistant Professor (Teaching Stream) in the Institute for Studies in Transdisciplinary Engineering Education & Practice and Division of Engineering Science at the University of Toronto. His teaching focuses on engineering design and his project work focuses on systems approaches to address problems in many domains, with his primary domain of application being the medicine, as well as inclusive approaches to engineering education and practice. Prior to the University of Toronto, Philip was an Assistant Professor of Electrical and Computer Engineering at Bucknell University, where he was awarded the President's Diversity, Equity and Inclusion Award and the Burma-Bucknell Bowl Award.

Homa Alemzadeh received her B.Sc. and M.Sc. in Computer Engineering from the University of Tehran and her Ph.D. in Electrical and Computer Engineering from the University of Illinois Urbana-Champaign. She is currently an assistant professor in the department of Electrical and Computer Engineering, with a courtesy appointment in the department of Computer Science at the University of Virginia. She is also affiliated with the LinkLab, a multi-disciplinary center for research and education in Cyber-Physical Systems (CPS). Before joining UVA, she was a research staff member at the IBM T. J. Watson Research Center. Her research interests are at the intersection of computer systems dependability and data science. She is particularly interested in data-driven resilience assessment and design of embedded and cyber-physical systems, with a focus on safety and security validation and monitoring in medical devices and systems, surgical robots, and autonomous systems. She is the recipient of the 2022 NSF CAREER Award and the 2017 William C. Carter Ph.D. Dissertation Award in Dependability from the IEEE TC and IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance.