Mechanical Systems and Signal Processing 241 (2025) 113537

Contents lists available at ScienceDirect

Mechanical Systems and Signal Processing

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

Full length article

Smart adaptive trigger sensing powered by edge intelligence and digital twin for energy-efficient wireless Structural Health Monitoring

Shuaiwen Cui [a,1], Yuguang Fu [a,*,2], Hao Fu [b,3], Xiao Yu [a,4], Wei Shen [c,5]

[a] School of Civil and Environmental Engineering, Nanyang Technological University, 50 Nanyang Ave, 639798, Singapore
[b] College of Mechanical and Vehicle Engineering, Chongqing University, No. 174 Shazheng Street, Shapingba District, 400030, Chongqing, China
[c] Institute for Risk and Reliability, Leibniz University Hannover, Callinstr. 34, Hannover, 30167, Germany

ARTICLE INFO

Communicated by M. Faes

Keywords:
Structural Health Monitoring
Edge intelligence
Trigger sensing
Energy efficiency
Feedback control
Bayesian Optimization
Digital twin

ABSTRACT

Capturing structural transient responses under sudden events, e.g., earthquakes, is a critical challenge for wireless Structural Health Monitoring (SHM) systems deployed on resource-constrained edge devices. To address it, researchers are developing event-triggered sensing to detect events while enhancing energy efficiency. However, this technology is fundamentally constrained by the selection of triggering conditions, e.g., vibration threshold, which are mostly based on empirical values and lack the adaptability required to handle dynamic and unpredictable environments. Such limitations give rise to an inherent trade-off between missed triggers and false triggers, hence possibly failing to satisfy practical application demands. To achieve the optimal triggering conditions that may vary under changing environments, this study introduces an adaptive triggering mechanism powered by edge intelligence and accelerated by Digital Twin (DT). By embedding adaptivity into the sensing process, the proposed approach dynamically optimizes critical parameters to achieve both detection accuracy and energy efficiency. In particular, anchored in a Feedback Control (FC) framework, the proposed mechanism leverages Bayesian Optimization (BO) and lightweight neural networks to address the challenges of data scarcity, data imbalance, partial observability, and high observation costs. Experimental validation on LiftNode, a low-power IoT sensor node, demonstrates the framework's capability to enhance detection performance (F_β) by approximately 30% compared to conservative manual parameter setting methods, while achieving similar or higher precision with computational overhead reduced by 2–3 orders of magnitude compared to batch exhaustive processing methods, marking a transformative step toward more reliable and sustainable SHM deployments.

* Corresponding author.
E-mail addresses: SHUAIWEN001@e.ntu.edu.sg (S. Cui), yuguang.fu@ntu.edu.sg (Y. Fu), fuhowe@cqu.edu.cn (H. Fu), XIAO004@e.ntu.edu.sg (X. Yu), wei.shen@irz.uni-hannover.de (W. Shen).
[1] Ph.D. Candidate, Nanyang Technological University, Singapore.
[2] Asst. Professor, Nanyang Technological University, Singapore.
[3] Ph.D. Candidate, Chongqing University, China.
[4] Ph.D. Candidate, Nanyang Technological University, Singapore.
[5] Postdoc, Leibniz University Hannover, Germany.

https://doi.org/10.1016/j.ymssp.2025.113537
Received 1 April 2025; Received in revised form 10 September 2025; Accepted 17 October 2025

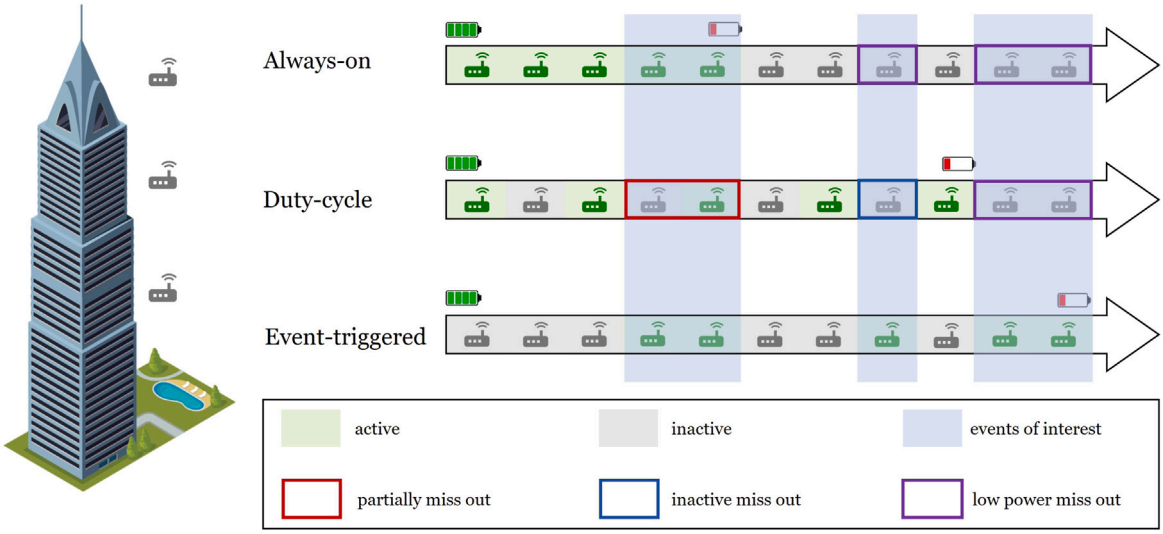


Fig. 1. Schematic illustration of different sensing schemes: always on scheme quickly depletes battery; duty-cycle scheme may miss events of interest; event-triggered scheme outperforms both above schemes.

1. Introduction

The convergence of Structural Health Monitoring (SHM) and the Internet of Things (IoT) has revolutionized infrastructure management, enabling real-time monitoring of critical assets such as buildings, bridges, and tunnels. However, real-world deployment often faces significant constraints, particularly in remote or inaccessible locations lacking stable power supplies [1–3]. As a result, many SHM systems rely on battery-powered sensor nodes to operate autonomously and wirelessly over extended periods [4–6]. This reliance introduces a critical challenge: frequent battery replacements are labor-intensive and often impractical in remote locations, while energy harvesting systems typically provide insufficient or unstable power supply. Therefore, optimizing power consumption and enhancing energy efficiency becomes essential for prolonging battery life, minimizing maintenance demands, and ensuring the consistent functionality of IoT-enabled wireless SHM systems [7–9].

To conserve power and extend the operational life of sensor nodes, various sensing schemes have been developed, as shown in Fig. 1, each with distinct trade-offs [1,3,10]. The always-on scheme, where sensors remain continuously active, ensures no event is missed but is only viable for systems with stable and abundant power supplies; for battery-powered devices, it rapidly depletes energy reserves [11]. To overcome this limitation, the duty-cycling scheme alternates sensors between active and sleep modes in predefined cycles, significantly reducing power consumption and extending battery life [5]. While simple in structure and logic, this approach risks missing critical sudden events during sleep periods, limiting its applicability in dynamic monitoring scenarios. The event-triggered scheme addresses these shortcomings by keeping sensors in a low-power state until specific conditions activate them, allowing for efficient data capture only when necessary. This approach significantly reduces the workload for downstream data processing and analysis, such as damage detection, localization, and assessment [12–14]. Although it may introduce hardware and software complexity, this sensing scheme is well justified by substantial gains in energy efficiency [1,15–17].

Among the three sensing schemes, event-triggered sensing stands out as the most agile and energy-efficient solution for battery-powered wireless SHM systems [18]. However, its implementation is inherently more complex [7,11,15–17]. Event-triggered sensing typically relies on the coordinated operation of two types of sensors: a low-power, low-resolution sensor and a high-power, high-resolution sensor, both capable of switching between sleep and active modes [1,15,19]. The low-power sensor acts as a sentinel, continuously monitoring for potential events of interest, while the high-power sensor is reserved for capturing detailed high-fidelity data when such events are detected. These sensor nodes operate in two modes: sentinel mode and active mode. In sentinel mode, the low-power sensor remains active to detect events, while the entire sensor node is in deep sleep mode, consuming minimal energy. Upon detecting an event, it wakes the sensor node and activates the high-power sensor, transitioning the system to active mode to record detailed information about the event. This collaborative approach effectively achieves both detection accuracy and energy efficiency, making it a pivotal innovation in modern event-triggered sensing systems for SHM applications [1,15,17,19,20].

The effectiveness of event-triggered sensing relies heavily on the event detection capabilities of low-power, low-resolution sensors, such as the ADXL362 [1,15,19]. These sensors use pre-defined conditions, including both thresholds and duration settings to determine whether an event of interest has occurred. However, the predefined and fixed nature of these parameters inherently lacks adaptivity, making it difficult to respond effectively to dynamic and complex conditions in real-world monitoring environments [21]. Without the ability to adjust to varying environmental factors, fixed threshold and duration mechanisms can result in frequent false triggers (i.e., false positives) or missed events (i.e., false negatives), significantly undermining the accuracy and reliability of wireless SHM systems [1,7,22]. The key challenge, therefore, lies in developing an intelligent event-triggered sensing mechanism

capable of dynamically adjusting its detection settings based on changing environmental conditions. Such a mechanism must strike an ideal optimization to achieve high accuracy and maintain low power consumption. More specifically, the challenge is to minimize missed detections while keeping the false trigger rate at an acceptably low level. Addressing this issue requires innovative solutions that leverage contextual awareness, adaptive optimization techniques, and edge intelligence for onboard realization to respond dynamically to environmental variations [23].

To this end, this study introduces a smart adaptive event-triggered sensing framework that addresses the challenge of harmonizing detection accuracy and energy efficiency in battery-powered wireless SHM systems. In particular, the framework leverages edge intelligence, where edge computing and artificial intelligence meet, to realize onboard optimization and adaptation for wireless SHM applications [24]. By enabling local data processing, edge intelligence reduces communication latency, enables adaptive parameter tuning, and context-aware control, making it ideal for resource-constrained environments [25]. The key contributions of this study are summarized as follows. (1) A feedback control mechanism is proposed as the backbone of smart adaptive trigger sensing, in which Bayesian Optimization (BO) dynamically adjusts triggering conditions to simultaneously optimize detection accuracy and energy consumption. (2) Moreover, edge intelligence is achieved by developing lightweight neural networks and deploying them directly on edge devices, thereby enabling context-aware, real-time decision-making and adaptive responses in wireless SHM systems. (3) Additionally, a staged optimization strategy. A staged trigger sensing parameter optimization strategy based on Digital Twin (DT) technology is introduced. In the first stage, pre-deployment optimization leverages abundant DT data and computational resources to perform a coarse optimization; in the second stage, onboard parameter fine-tuning is conducted using limited, partially observable real-world data, substantially accelerating the overall optimization process. (4) Finally, experimental validation on the LiftNode platform demonstrates the framework's capability to enhance detection performance (F_β) by approximately 30% compared to conservative manual parameter setting methods, while achieving similar or higher precision with computational overhead reduced by 2–3 orders of magnitude compared to batch exhaustive processing methods.

The remainder of this paper is organized as follows. Section 2 reviews related work, tracing the evolution of sensing schemes and identifying key research gaps. Section 3 introduces the proposed Smart Adaptive Triggering Mechanism (SATM), providing a detailed explanation of its core components, including the environment, system model, estimator, and controller. Section 4 describes the staged deployment strategy, emphasizing the optimization processes in both pre-deployment and onboard implementation phases. Section 5 presents the experimental validation results, comparing the proposed approach with traditional methods. Finally, Section 6 concludes the paper.

2. Related works

2.1. Triggering performance characterization

Before delving into technical details, it is essential to establish a mathematical framework for characterizing the performance of triggering mechanisms. This mathematical foundation enables a systematic understanding of hardware, software, and algorithmic design considerations. The trigger sensing mechanism essentially functions as a binary classifier, where the input is the time series signals, e.g., vibration, and the output is the event detection result. The performance of a binary classifier can be described using a confusion matrix [26], which consists of four elements: true positive (TP), false positive (FP), true negative (TN), and false negative (FN), as shown in Fig. 2. In this context, the two most critical types of errors are missed triggers and false triggers, corresponding to false negatives and false positives, respectively. Specifically, missed trigger events occur when events of interest fail to activate the sensing mechanism, while false trigger events occur when events of no interest erroneously activate the mechanism. The missed trigger rate is closely related to recall (recall = TP/(TP + FN)), while the false trigger rate corresponds to precision (precision = TP/(TP + FP)). In SHM applications, the primary goal for trigger sensing is to minimize the missed trigger rate while keeping the false trigger rate at an acceptably low level. Consequently, the performance of the trigger sensing mechanism can be quantified by an index synthesizing both recall and precision. This index can be further characterized using the F_β score [26], defined as:

$$F_\beta = (1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{\beta^2\cdot\text{precision}+\text{recall}} \tag{1}$$

Here, β is a parameter that adjusts the emphasis between recall and precision. The F_β score provides a robust metric to evaluate the effectiveness of the triggering mechanism in achieving the desired balance between detection accuracy and energy efficiency. For cases recall is more important, β can be set to a value greater than 1, while for cases precision is more important, β can be set to a value less than 1. In practice, the optimal value of β is determined by the specific requirements of the SHM application.

2.2. Hardware foundations

Hardware serves as the fundamental foundation for implementing trigger sensing mechanisms in wireless structural health monitoring systems [17,27]. Current research has demonstrated various hardware configurations and sensor combinations that enable energy-efficient event detection. As a representative example, Xnode [1,15] adopts the previously introduced dual-sensor architecture, comprising a continuous low-power, low-resolution sensor for event detection and a high-power, high-resolution sensor for detailed data collection, as illustrated in Fig. 3. A Metal–Oxide–Semiconductor Field-Effect Transistor (MOSFET) circuit is integrated onboard to switch the microcontroller and the high-power, high-resolution sensor between sleep and active modes

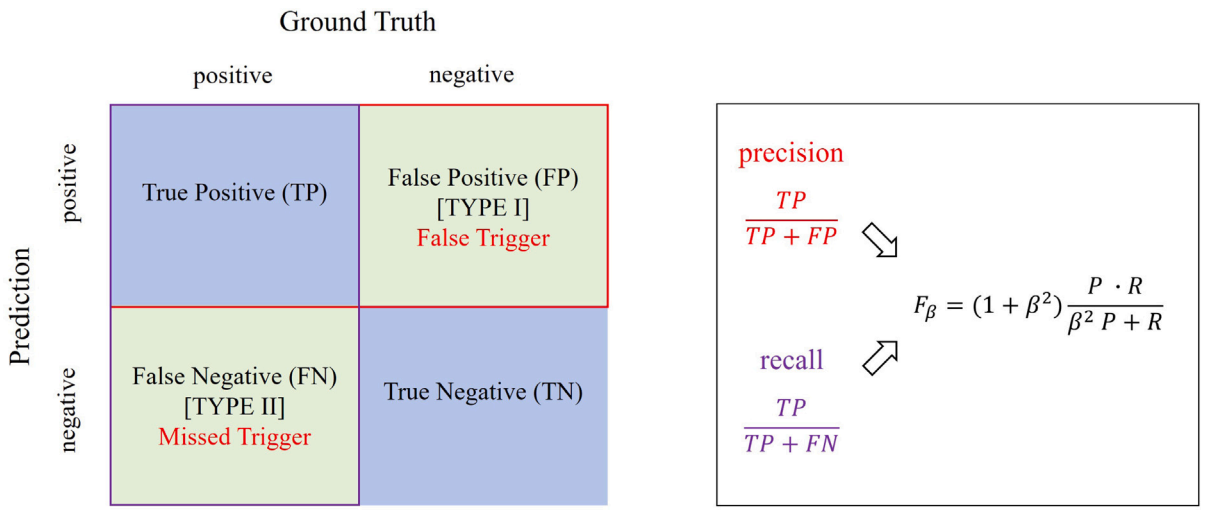
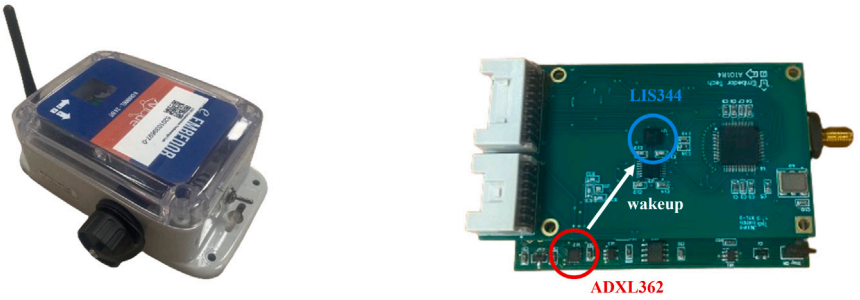


Fig. 2. Confusion matrix for trigger sensing performance evaluation: TP, FP, TN, FN with error types (False Trigger/Type I, Missed Trigger/Type II), and performance metrics including precision, recall, and F_β score calculation formula.



Mode \ Sensor	ADXL362 (12 bit)	LIS344 (24 bit)
Sentinel Mode	Working 270nA	Sleeping 1uA
Working Mode	Sleeping 10nA	Working 680uA

Fig. 3. Dual-sensor architecture of Xnode: ADXL362 (12-bit sentinel sensor) and LIS344 (24-bit working sensor) with wakeup mechanism and power consumption comparison in different operation modes.

according to triggering signals from the low-power sensor, while preserving the timer-based periodic wake-up functionality [7]. Similar dual-sensor architectures have been successfully implemented in recent SHM wireless sensor node designs [19].

The scope of trigger sensing extends beyond single-signal detection, evolving into multi-metric sensing triggering, also known as sensor fusion triggering. Through circuit design modifications, the triggering mechanism can incorporate multiple types of trigger signals from diverse sensors, including vibration, strain, and timer-based triggers [15,20]. With proper sensor and circuit design optimization, the operational current consumption can be reduced to levels even below the battery leakage current, thereby extending the lifespan of battery-powered sensor nodes to approach the theoretical battery life [20]. Furthermore, sophisticated trigger sensing design can significantly reduce the number of sampling points by implementing intelligent sampling strategies that minimize data collection during periods of low or negligible metric changes [16].

2.3. Software and algorithmic foundations

Complementing the hardware foundation, sophisticated software algorithms and computational frameworks are essential for realizing the full potential of trigger sensing mechanisms [11]. The core of trigger sensing algorithms can be mathematically formulated as a decision-making process based on monitoring metrics and predefined thresholds. Monitoring metrics in trigger sensing can be either absolute values (e.g., acceleration amplitude) or relative values (e.g., gap between two points), but they can all be unified within a relative error framework [28,29]. Therefore, we use the symbol $\boldsymbol{e}$ to represent the monitoring metric vector,

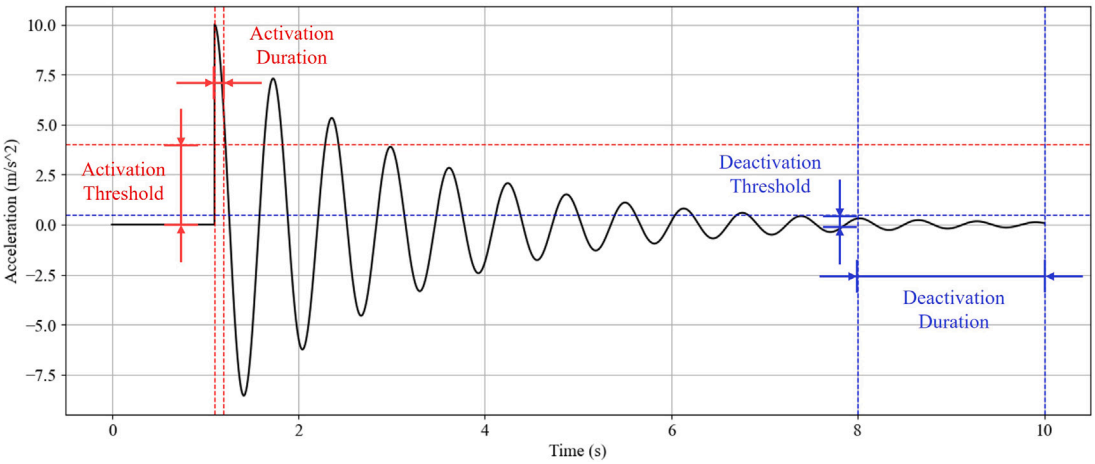


Fig. 4. Schematic illustration of the triggering and detriggering mechanism on Xnode: acceleration signal with activation and deactivation thresholds and durations in time-domain response.

embodying the concept of deviation from a reference state [5,11]. At its most fundamental level, a trigger sensing mechanism operates as a binary classifier, where the input is the monitoring metric vector $\boldsymbol{e} = [e_1, e_2, \ldots, e_n]^T$ and the output is a binary trigger decision:

$$\text{Trigger} = \begin{cases} 1 & \text{if } \boldsymbol{e} \geq \boldsymbol{\tau} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_n]^T$ represents the threshold vector. This is the most basic formulation of trigger sensing and the most popular one being used for trigger sensing in wireless SHM systems [5,16,18]. For most cases, the number of elements in $\boldsymbol{\tau}$ is 1, i.e., $\boldsymbol{\tau} = [\tau]^T$. However, the basic formulation (2) often leads to frequent false triggers due to transient signal fluctuations. To address this limitation, practical trigger sensing mechanisms incorporate temporal considerations by introducing a duration threshold d:

$$\text{Trigger}(t) = \begin{cases} 1 & \text{if } \boldsymbol{e}(t) \geq \boldsymbol{\tau} \text{ and } \text{duration}(\boldsymbol{e} \geq \boldsymbol{\tau}) \geq d \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This enhanced formulation ensures that triggering occurs only when the monitoring metrics consistently exceed their thresholds for a specified duration, thereby significantly reducing false positive rates while maintaining high detection sensitivity. A typical implementation of the trigger sensing mechanism with temporal considerations is the Xnode [1,15]. As shown in Fig. 4, the low-power sensor utilizes predefined thresholds and durations to determine the occurrence of an event of interest. When the signal amplitude exceeds the predefined threshold for a specified duration, an event is considered detected, and the low-power sensor activates the high-power sensor, transitioning the system into the working mode to capture detailed event data.

Building upon the mathematical formulation (3), the threshold vector $\boldsymbol{\tau}$ plays a crucial role in determining the sensitivity and adaptability of the system. In the literature, threshold vectors can be categorized into two primary types based on their temporal characteristics. The most fundamental type is the static constant threshold, where all elements in $\boldsymbol{\tau}$ remain fixed over time, i.e., $\boldsymbol{\tau}(t) = \boldsymbol{\tau}_0$ for all t [5,7,19]. While this approach offers simplicity and computational efficiency, it lacks environmental adaptability, as the fixed thresholds cannot respond to changing environmental conditions or varying structural behaviors [7]. In contrast, dynamic variable thresholds represent a more sophisticated approach [11], where the elements of the threshold vector are time-dependent and can be expressed as a function of multiple variables:

$$\boldsymbol{\tau}(t) = f(t, \boldsymbol{e}_{\text{hist}}, \boldsymbol{v}_{\text{env}}, \boldsymbol{u}_{\text{user}}) \tag{4}$$

where t represents time, $\boldsymbol{e}_{\text{hist}}$ denotes the historical monitoring metrics, $\boldsymbol{v}_{\text{env}}$ represents environmental variables, and $\boldsymbol{u}_{\text{user}}$ represents user-defined parameters. This formulation enables the threshold vector to adapt to environmental changes, structural variations, and evolving event characteristics, thereby improving both detection accuracy and energy efficiency while maintaining computational tractability [25].

2.4. Research gaps and challenges

The majority of trigger sensing systems still rely on a limited number of monitoring metrics, often using only a single amplitude value, and employ static constant parameters without considering adaptivity [5]. Triggering parameter optimization methods have been developed to achieve adaptivity in dynamic environments, and can be categorized into three main approaches. (1) The first approach employs conservative initial parameter settings followed by manual adjustment based on pioneer sensing data. This

method typically starts with low threshold values to ensure minimal missed triggers, then relies on human intervention to fine-tune parameters based on observed performance. While straightforward to implement, this approach is labor-intensive and lacks real-time adaptability [1]. (2) The second approach utilizes historical observation data to compute statistical indicators such as mean, variance, and percentiles for parameter setting. This data-driven method provides more objective parameter determination, but treats historical data as a static whole and typically uses it only once or a few times, without implementing iterative dynamic adjustments [29]. (3) The third approach implements predictive parameter setting, considering time-varying characteristics such as drift in monitoring metrics. This method addresses the limitations of static approaches by anticipating environmental changes, but often makes oversimplified assumptions about time-varying characteristics, such as linear changes, failing to capture the complex, non-linear nature of real environmental variations [5].

The primary research gap lies in the fundamental limitations of these existing approaches in achieving true real-time adaptivity. While the three methods discussed above represent progress toward adaptive triggering, they all suffer from inherent shortcomings that prevent them from achieving optimal performance in dynamic environments. Current solutions either rely heavily on human intervention for parameter tuning and decision-making, or require remote cloud computing assistance for complex optimization tasks, both of which are impractical for autonomous edge deployment. Furthermore, none of these approaches adequately address the challenge of balancing multiple conflicting objectives, such as detection accuracy versus energy efficiency, in a unified optimization framework that can operate independently on resource-constrained edge devices.

Several challenges impede the design and implementation of adaptive triggering mechanisms. (1) The first challenge is the fundamental limitation of existing parameter optimization approaches in achieving true real-time adaptivity. Current methods, whether based on manual adjustment, static statistical analysis, or simplified predictive models, all fail to provide continuous, iterative adaptation to dynamic environmental changes. (2) The second challenge involves data unobservability and uncertainty. In real-world deployments, some data cannot be fully observed; for example, events that fail to trigger data collection are never recorded and thus cannot be analyzed further. Moreover, certain data may not be collected at all (e.g., the ground truth label of an event type), while other data may exhibit high uncertainty during collection (e.g., the distribution of events such as earthquakes). (3) The third challenge is the high observation cost, as the computation of metrics like the F_β score requires extensive data collection and processing, thereby imposing substantial time and computational overheads on resource-constrained edge devices.

2.5. Potential enablers

Building upon the analysis of the research gaps and challenges, several key enablers can be synthesized into a cohesive solution for a smart adaptive triggering mechanism, including (1) feedback control, (2) Bayesian Optimization (BO), (3) digital twin technology, and (4) edge intelligence. Together, these enablers can address the major challenges introduced in Section 2.4.

Feedback control [30] can serve as the cornerstone of adaptivity by incorporating the F_β score into its feedback loop, directly addressing Challenge (1) (real-time adaptivity limitation). This integration can unify recall and precision into a single optimization metric, thereby transforming a multi-objective optimization problem into a single-objective one. Complementing this, Bayesian Optimization [31] can act as the engine to power feedback control. By leveraging surrogate models and iterative updates, BO can efficiently guide parameter adjustments, effectively handle black-box problems, and reduce observation costs in resource-constrained environments, making it a promising solution to Challenge (3) (high observation cost). To enhance the performance of the adaptive triggering, digital twin technology [32] and edge intelligence [4] can be introduced as acceleration engines. Digital twin technology can enable controlled data augmentation through the generation of synthetic datasets, mitigating issues such as unknown event distributions (as mentioned in Challenge (2) (data unobservability and uncertainty)) and substantially lowering real-world optimization costs (as mentioned in Challenge (3) (high observation cost)) by facilitating iterative computations in virtual environments. Meanwhile, edge intelligence can be realized by deploying lightweight neural networks directly on edge devices, which can allow for real-time ground truth estimation and the inference of unobservable data (e.g., missed events) as mentioned in Challenge (2) (data unobservability and uncertainty), further enhancing system adaptability and efficiency [26,33].

3. Smart adaptive triggering mechanism

As depicted in Fig. 5, this study introduces a smart adaptive triggering mechanism that integrates multiple enabling technologies to address the inherent challenges outlined in Section 2. This section begins with an overview of the proposed mechanism, followed by a detailed discussion of its constituent components.

3.1. Overview

The proposed SATM employs feedback control as its backbone, serving as the key mechanism for adaptive optimization toward global optimal configurations. A feedback control loop typically consists of four major components: the environment, the system model, the estimator, and the controller [30], as illustrated in Fig. 5(a). In the context of SHM and trigger sensing, the environment represents the target structure's responses to various excitations. The system component corresponds to the threshold- and duration-based triggering mechanism, which serves as the primary optimization target. The estimator assesses the system's performance, providing essential feedback to guide adjustments. Lastly, the controller functions as the driving engine, steering the system toward its optimal configuration by addressing two critical aspects: (1) identifying the optimal state and (2) determining the pathway to achieve it.

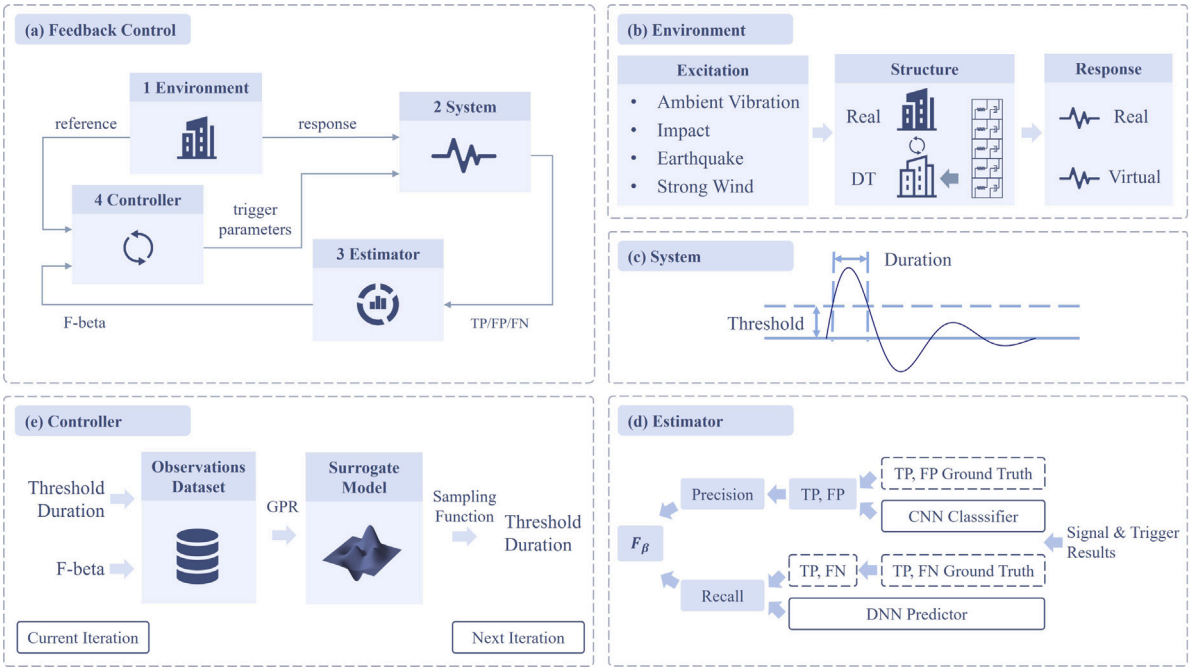


Fig. 5. Feedback control for adaptive triggering mechanism: (a) overall framework combining all four components — feedback control loop, (b) environment component — input signals, (c) system component — trigger sensing mechanism, (d) estimator component — performance assessment, and (e) controller component — parameter adjustment.

3.2. The environment

The environment component is characterized in relation to the system component and serves as the source of input signals for the system in trigger sensing for SHM. Specifically, it represents the structural responses generated by external excitations. As illustrated in Fig. 5(b), the environment component comprises three interconnected elements: (1) excitations, (2) structures, and (3) responses. In practical applications, responses are typically collected directly by sensors. However, to leverage the advantages of digital twin technology, this section focuses on constructing an effective surrogate model for the environment component.

3.2.1. Excitations

Excitations refer to external forces or disturbances acting on the structures, caused by various events, whether of interest or not. For the purpose of this study, four types of events are considered: (1) ambient vibration, (2) impact, (3) earthquake, and (4) strong wind. Among these, ambient vibration is categorized as an uninteresting event, while the remaining three are classified as events of interest. In engineering practice, these events often occur naturally within the environment. However, in the digital twin framework, excitation signals can be synthetically generated using computational methods to accelerate the optimization process. The technical details for signal generation can obscure the core concepts of the proposed mechanism, and thus only the key points are highlighted below. Detailed technical specifications for signal generation can be found in Appendix.

Firstly, ambient vibration refers to environmental noise, which can be modeled as Gaussian white noise characterized by its mean value and standard deviation. It acts on every degree of freedom of the structure, representing the background noise present in the monitoring environment. Secondly, impact events can be modeled as single impulse signals with predefined amplitude and duration. In practice, these signals represent spike-like forces caused by sudden external disturbances acting on the structure. For simulation purposes, impacts can be applied to a random degree of freedom with random amplitude and direction for a very short duration. Thirdly, earthquakes can be modeled as the superposition of multiple sinusoidal waves with varying frequencies, amplitudes, and phases. These signals should first be modulated in the frequency domain to align with the power spectral density of real earthquake data, then further modulated in the time domain to generate time-history signals. The forces can be applied at the ground level and distributed across the structure's degrees of freedom using a predefined force distribution matrix. Fourthly, strong wind can first be modeled based on wind pressure distribution acting on the structure and then equivalently converted into concentrated forces applied to the structure's mass points. In addition to frequency domain modulation, time domain modulation can be applied to capture the temporal evolution of wind forces. Most importantly, the spatial interrelation between different points on the structure should be considered to accurately represent the complex nature of wind loading.

3.2.2. Structures

Structures represent the physical systems being monitored, serving as both the targets of excitations and the sources of responses. Acting as intermediaries between excitations and responses, the dynamic behaviors of structures govern how they react to various external forces. These behaviors are typically characterized by the equation of motion, which, for an n-degree-of-freedom system, is expressed as:

$$\boldsymbol{M}\ddot{\boldsymbol{x}}(t) + \boldsymbol{C}\dot{\boldsymbol{x}}(t) + \boldsymbol{K}\boldsymbol{x}(t) = \boldsymbol{G}\boldsymbol{P}(t), \tag{5}$$

where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, $\boldsymbol{C} \in \mathbb{R}^{n\times n}$, and $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ are the mass, damping, and stiffness matrices, respectively; $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the displacement vector; $\dot{\boldsymbol{x}}(t)$ and $\ddot{\boldsymbol{x}}(t)$ represent the velocity and acceleration vectors, respectively. On the right-hand side, $\boldsymbol{G} \in \mathbb{R}^{n\times m}$ is the force distribution matrix, and $\boldsymbol{P}(t) \in \mathbb{R}^m$ is the time-dependent force vector representing excitations acting on the structure. In this study, for simplicity, it is assumed that $m = n$, meaning that each degree of freedom has a corresponding input force. For numerical analysis and simulation, the equation of motion can be reformulated into a state-space representation to facilitate computation and control analysis [34].

3.2.3. Responses

Responses are signals generated by the dynamic interaction of excitations and structures, representing structural behaviors under external forces. These signals, whether real (from physical environments) or virtual (from digital twins), serve as the critical input to the system component for detecting events of interest. A widely used method for simulating structural responses is the Newmark-beta method [34], which is an iterative numerical integration scheme for solving the equation of motion in structural dynamics. The algorithmic process is summarized as Algorithm 1.

Algorithm 1 Newmark-beta Method for Structural Response Simulation [34]

Input: mass matrix $\boldsymbol{M}$, damping matrix $\boldsymbol{C}$, stiffness matrix $\boldsymbol{K}$, force distribution matrix $\boldsymbol{G}$, external force $\boldsymbol{P}(t)$, initial displacement $\boldsymbol{x}_0$, initial velocity $\dot{\boldsymbol{x}}_0$, initial acceleration $\ddot{\boldsymbol{x}}_0$, time step Δt, integration parameters β_{newmark}, γ (typically set to 0.25 and 0.5, respectively)
Output: displacement $\boldsymbol{x}_t$, velocity $\dot{\boldsymbol{x}}_t$, acceleration $\ddot{\boldsymbol{x}}_t$
Initialization:
Compute helper coefficients: $a_0 = 1/(\beta_{\text{newmark}}\Delta t^2)$, $a_1 = \gamma/(\beta_{\text{newmark}}\Delta t)$, $a_2 = 1/(\beta_{\text{newmark}}\Delta t)$, $a_3 = 1/(2\beta_{\text{newmark}}) - 1, a_4 = \gamma/\beta_{\text{newmark}} - 1, a_5 = \Delta t/2(\gamma/\beta_{\text{newmark}} - 2), a_6 = \Delta t(1-\gamma), a_7 = \gamma\Delta t$
Compute effective stiffness matrix: $\boldsymbol{K}^e = \boldsymbol{K} + a_1\boldsymbol{C} + a_0\boldsymbol{M}$
$\boldsymbol{x} \leftarrow \boldsymbol{x}_0$, $\dot{\boldsymbol{x}} \leftarrow \dot{\boldsymbol{x}}_0$, $\ddot{\boldsymbol{x}} \leftarrow \ddot{\boldsymbol{x}}_0$
Iteration:
for each time step $t = 1$ to T **do**
  Compute effective force: $\boldsymbol{G}\boldsymbol{P}^e_{t+\Delta t} = \boldsymbol{G}\boldsymbol{P}_{t+\Delta t} + (a_0\boldsymbol{x}_t + a_2\dot{\boldsymbol{x}}_t + a_3\ddot{\boldsymbol{x}}_t)\boldsymbol{M} + (a_1\boldsymbol{x}_t + a_4\dot{\boldsymbol{x}}_t + a_5\ddot{\boldsymbol{x}}_t)\boldsymbol{C}$
  Compute the displacement at $t + \Delta t$: $\boldsymbol{x}_{t+\Delta t} = (\boldsymbol{K}^e)^{-1}\boldsymbol{G}\boldsymbol{P}^e_{t+\Delta t}$
  Compute the acceleration at $t + \Delta t$: $\ddot{\boldsymbol{x}}_{t+\Delta t} = a_0(\boldsymbol{x}_{t+\Delta t} - \boldsymbol{x}_t) - a_2\dot{\boldsymbol{x}}_t - a_3\ddot{\boldsymbol{x}}_t$
  Compute the velocity at $t + \Delta t$: $\dot{\boldsymbol{x}}_{t+\Delta t} = \dot{\boldsymbol{x}}_t + a_6\ddot{\boldsymbol{x}}_t + a_7\ddot{\boldsymbol{x}}_{t+\Delta t}$
  Update current state: $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_{t+\Delta t}$, $\dot{\boldsymbol{x}}_t \leftarrow \dot{\boldsymbol{x}}_{t+\Delta t}$, $\ddot{\boldsymbol{x}}_t \leftarrow \ddot{\boldsymbol{x}}_{t+\Delta t}$
end for

3.3. The system

Algorithm 2 Threshold- and Duration-Based Triggering Mechanism

Input: signal value s, threshold τ, duration d
Output: trigger flag T (binary: 1 for trigger, 0 for no trigger; initialized as 0)
Internal Variable: counter c (initialized as 0, used to track consecutive time steps)
For each time step t:
if $|s| \geq \tau$ **then**
  Increment counter: $c \leftarrow c + 1$
  if $c \geq d$ **then**
    Set trigger flag: $T \leftarrow 1$
    Reset counter: $c \leftarrow 0$
  end if
else
  Reset counter: $c \leftarrow 0$
  Set trigger flag: $T \leftarrow 0$
end if

The system component refers to the threshold- and duration-based triggering mechanism, which functions as a binary classifier and serves as the primary optimization target within the feedback control loop [1]. In practical applications, this mechanism is

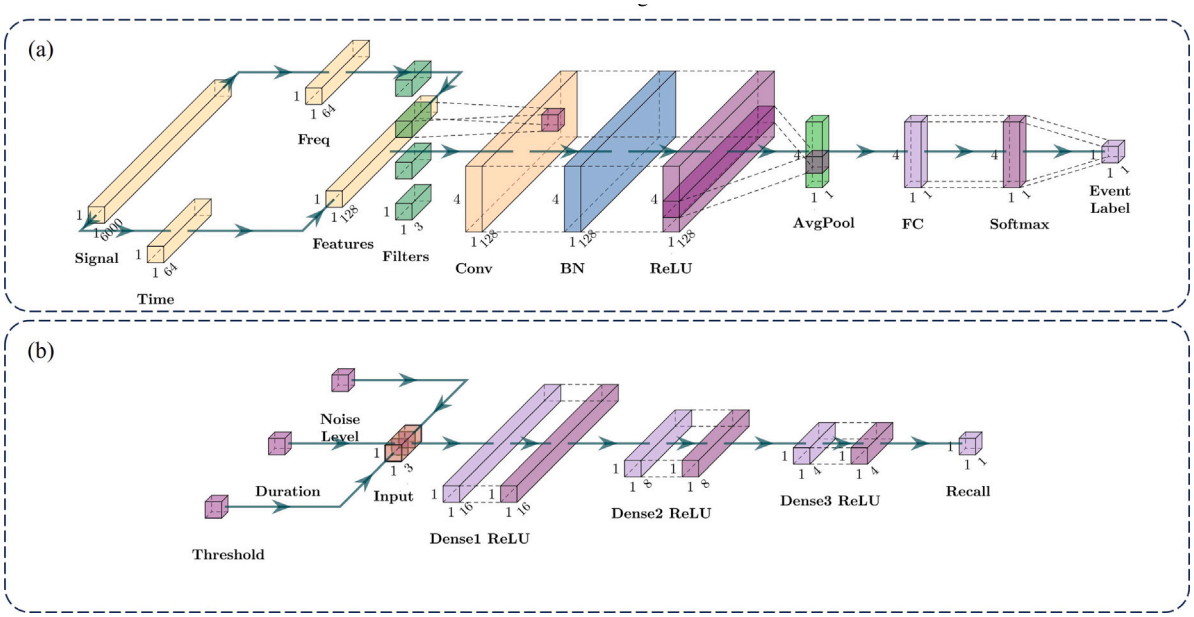


Fig. 6. Neural networks for onboard inference: (a) CNN as surrogate classifier for ground truth absence, enabling precision calculation; (b) DNN as recall estimator for trigger mechanism's partial observability, providing recall estimation.

implemented in low-level hardware, such as the ADXL362 sensor [7,19,35], with users able to adjust its triggering parameters – thresholds and durations – via software interfaces. The threshold parameter operates within a continuous value space, whereas the duration parameter is defined in a discrete space, corresponding to an integer number of sampling points. Complementing the triggering mechanism is a deactivation mechanism, as shown in Fig. 4, which, although outside the scope of this study, plays a critical role in real-world applications. The deactivation mechanism resets the system to sentinel mode after capturing an event, ensuring readiness for subsequent detection cycles and enabling seamless operation.

As illustrated in Fig. 5(c), the triggering mechanism relies on monitoring consecutive time steps where the signal amplitude surpasses the predefined threshold. When the count of such steps meets or exceeds the specified duration, an event is triggered, transitioning the system from sentinel mode to active mode. The underlying logic of this mechanism can be formalized in Algorithm 2, which serves two critical roles: first, it encapsulates the operational principles governing real hardware implementations; second, it provides the foundation for the system component within the digital twin framework, facilitating virtual testing and optimization of the triggering mechanism under diverse conditions.

3.4. The estimator

The estimator evaluates the system's performance using the F_β score, a comprehensive metric that harmonizes recall and precision. This simplifies the optimization process by transforming a multi-objective problem into a single-objective one. The F_β score, defined in Eq. (1), adjusts the trade-off between recall and precision based on the parameter β. For SHM applications, $\beta > 1$ prioritizes recall, while $\beta < 1$ emphasizes precision, depending on specific monitoring requirements.

The F_β score calculation relies on precision and recall, which are derived from true positives (TP), false positives (FP), and false negatives (FN). In digital twin simulations, these values are known or synthetic. However, real-world implementations face challenges: (1) the lack of ground truth for triggered events and (2) the unobservability of untriggered events. Edge intelligence can address these issues by inferring ground truth and estimating unobservable data, as illustrated in Fig. 5(d).

Given the constraints of SHM applications, which often use low-cost, low-power sensors with limited resources, the neural networks deployed at the edge must be lightweight. This study designs efficient neural networks for event labeling and recall estimation, ensuring a synergy of high accuracy and computational efficiency. These lightweight architectures are feasible for deployment on resource-constrained edge devices while maintaining reliable performance.

3.4.1. CNN model for onboard event labeling

In onboard computing, the absence of ground truth labels for collected data complicates determining true positives (TP) and false positives (FP). To address this, a lightweight convolutional neural network (CNN) is designed for deployment on resource-constrained devices. This CNN estimates labels for collected data, enabling TP and FP calculation. The architecture, shown in Fig. 6(a), consists of two stages: feature extraction and classification.

Table 1
Lightweight neural networks: CNN event classifier and DNN recall estimator.

Parameter type	CNN event classifier	DNN recall estimator
Total Parameters	142 (572.00 B)	209 (836.00 B)
Trainable Parameters	44 (176.00 B)	209 (836.00 B)
Non-trainable Parameters	8 (32.00 B)	0 (0.00 B)
Optimizer Parameters	90 (364.00 B)	0 (0.00 B)

The feature extraction stage reduces computational complexity while retaining essential signal characteristics. The raw time-domain signal is downsampled, and its frequency-domain features are derived using the Short-Time Fourier Transform (STFT). Both domains are further downsampled to reduce dimensionality, and their features are concatenated into a unified input vector. This process reduces a 60-second, 100 Hz signal to as few as 128 features, significantly lowering computational load while preserving sufficient information for accurate classification.

The classification stage employs a compact 1D CNN architecture tailored for time-series data. As shown in Fig. 6(a), the architecture begins with a 1D convolutional layer featuring four 1-by-3 filters, extracting information from the input features to generate a four-channel output. Batch normalization and ReLU activation are applied, followed by a global average pooling layer, which condenses the information into a single vector. Finally, this vector is passed to a fully connected layer with softmax activation to produce the classification result. This streamlined design harmonizes computational efficiency and classification performance. As revealed by Table 1, the CNN classifier model is highly lightweight, with only 142 parameters (572 bytes), making it suitable for onboard deployment.

3.4.2. DNN model for onboard recall estimation

Unlike precision, which is derived from collected data, recall estimation addresses the partial observability issue due to events that fail to trigger high-resolution sensing. To overcome this, a lightweight deep neural network (DNN) is developed to directly estimate recall, as illustrated in Fig. 6(b). The DNN evaluates system performance under specific triggering parameters, with threshold, duration, and noise level selected as inputs. Noise level is critical, as events submerged in noise are harder to detect [36]. The network architecture, optimized through iterative refinements, consists of three fully connected layers with 16, 8, and 4 neurons, each followed by ReLU activation, and a single output neuron predicting recall. This lightweight model has 209 parameters (836 bytes), as detailed in Table 1.

System performance depends on both triggering parameters and data distribution. While noise level is included as an input, other factors like event frequency and distribution are harder to quantify. To address this, a digital twin generates artificial data distributions for optimization and evaluation. For training, datasets of events and noise at varying levels are combined, and recall values are calculated for discretized triggering parameters. This yields recall data under diverse conditions, which are used to train the DNN. In real-world applications, the system calculates noise levels from input signals and predicts recall for given triggering parameters. In practice, as more real-world event records are collected over time, artificial data distributions can be gradually replaced, enabling continuous system improvement. Further details on artificial data generation will be discussed in Section 4.

3.5. The controller

In the whole feedback loop, the controller acts as both the brain and the heart, steering the system toward its optimal configuration. Its primary responsibilities are to identify the optimal configuration and determine how to reach it efficiently. In the context of SATM, the controller dynamically adjusts the triggering parameters based on feedback from the estimator, optimizing the F_β score. This process faces two significant challenges: (1) the system operates as a black-box optimization problem due to its complex nature without explicit mathematical form, and (2) high observation costs arise from extensive data collection and processing required to evaluate the F_β score, which imposes substantial overhead on resource-constrained edge devices. To address these challenges, the controller employs Bayesian Optimization (BO), a technique well-suited for black-box problems with high observation costs [31].

3.5.1. Bayesian optimization framework

The goal of BO is to iteratively find the optimal parameters $\hat{\boldsymbol{x}}$ in a predefined search space $\mathcal{X}$ that maximize an objective function $f(\boldsymbol{x})$:

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}). \tag{6}$$

BO uses a surrogate model $\mathcal{M}$ and an acquisition function $a(\boldsymbol{x}, \mathcal{M})$ to iteratively select parameters $\boldsymbol{x}_t$. The acquisition function balances exploration and exploitation to determine the next parameters to evaluate:

$$\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}\in\mathcal{X}} a(\boldsymbol{x}, \mathcal{M}). \tag{7}$$

After evaluating $f(\boldsymbol{x}_t)$, the resulting pair $(\boldsymbol{x}_t, f(\boldsymbol{x}_t))$ is added to the dataset $\boldsymbol{D}$. This process repeats until a stopping criterion, such as reaching a maximum number of iterations T or a negligible improvement threshold, is met. Algorithm 3 outlines the BO process [31].

Algorithm 3 Bayesian Optimization Algorithm Framework

Input: Search space $\mathcal{X}$, objective function f, surrogate model $\mathcal{M}$, acquisition function a
Output: Dataset $\mathcal{D}$ (set of sampled points and their evaluations)
Initialize dataset: $\mathcal{D} \leftarrow \text{InitSamples}(f, \mathcal{X})$
for $i = |\mathcal{D}|$ to T **do**
  Fit the model: $p(y \mid \boldsymbol{x}, \mathcal{D}) \leftarrow \text{FitModel}(\mathcal{M}, \mathcal{D})$
  Select next point: $\boldsymbol{x}_i \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{X}} a(\boldsymbol{x}, p(y \mid \boldsymbol{x}, \mathcal{D}))$
  Evaluate objective function: $y_i \leftarrow f(\boldsymbol{x}_i)$
  Update dataset: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{x}_i, y_i)\}$
end for
Pick the best point from the dataset $\mathcal{D}$ as the optimum: $\hat{\boldsymbol{x}} \leftarrow \arg\max_{(\boldsymbol{x},y) \in \mathcal{D}} y$

The BO process can be roughly divided into two stages: initialization and iteration. Initialization stage is to generate an initial dataset $\mathcal{D}$ by randomly sampling points from the search space $\mathcal{X}$. The following stage, iteration, is to use the acquisition function a to determine the next sampling point to observe and to update the surrogate model $\mathcal{M}$. This process continues until the stopping criterion is met, ensuring efficient optimization even under constraints.

Through the whole process of BO, there are some key concepts to be noted:

- Search Space $\mathcal{X}$: The search space defines the feasible region where the optimal parameters are sought. Based on prior knowledge and constraints, the search space is confined to save computational resources and time. Note that for SATM, the threshold is continuous, while the duration is discrete (integer values standing for the number of sampling points). The duration can be treated as continuous variable first and then rounded to the nearest meaningful integer.
- Objective Function f: The objective function maps the triggering parameters to the F_β score, quantifying the performance of the system under specific configurations. Yet, there is no explicit mathematical form for f in SATM, making it a black-box problem.
- Surrogate Model $\mathcal{M}$: As the objective function is unknown, the surrogate model is used to approximate it. The surrogate model is updated iteratively based on the dataset $\mathcal{D}$ to predict the objective function at unobserved points [37]. Since there is no analytical expression for the underlying objective function, for simplicity in the application part, symbol f is directly used to represent the surrogate model.
- Acquisition Function a: The acquisition function guides the selection of the next point to evaluate. It balances exploration and exploitation, aiming to find the optimal parameters efficiently.
- Dataset $\mathcal{D}$: The dataset $\mathcal{D}$ stores the sampled points and corresponding evaluations. It is updated iteratively as new points are evaluated, which also records the history of the optimization process. Note that based on the additive nature of the dataset, the optimization can be divided into multiple stages, which is the foundation for the multi-stage deployment strategy proposed in Section 4.

3.5.2. Surrogate model based on Gaussian process regression (GPR)

In SATM, the surrogate model is constructed using GPR [25]. Compared to the unknown latent objective function, the surrogate model features many great mathematical properties, such as differentiability, continuity, and smoothness. These properties make GPR an ideal choice for approximating the objective function in the black-box optimization problem. A Gaussian Process (GP) is defined as a collection of random variables, any finite number of which follow a joint Gaussian distribution [31]. It is typically represented by a mean function $\mu(x)$ and a covariance function $\text{Cov}(x, x')$. A GP $f(x)$, including noise, can be expressed as:

$$f(x) \sim \mathcal{GP}(\mu(x), \text{Cov}(x, x') + \sigma_n^2 \delta_{xx'}) \tag{8}$$

where σ_n^2 represents the noise variance, and $\delta_{xx'}$ is the Kronecker delta function. The mean and covariance functions are defined as $\mu(x) = \mathbb{E}(f(x))$ and $\text{Cov}(x, x') = \mathbb{E}((f(x) - \mu(x))(f(x') - \mu(x')))$, respectively.

With noisy observation vector $\boldsymbol{y}$, the joint distribution between the observed values $\boldsymbol{y}$ and the predicted values $\boldsymbol{y}^*$ follows a joint Gaussian distribution:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}) \\ \mu(\boldsymbol{x}^*) \end{bmatrix}, \begin{bmatrix} \text{Cov}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \mathbf{I}, & \text{Cov}(\boldsymbol{x}, \boldsymbol{x}^*) \\ \text{Cov}(\boldsymbol{x}^*, \boldsymbol{x}), & \text{Cov}(\boldsymbol{x}^*, \boldsymbol{x}^*) \end{bmatrix} \right) \tag{9}$$

where $\text{Cov}(\boldsymbol{x}, \boldsymbol{x})$ is the covariance matrix of the observed point vector, $\text{Cov}(\boldsymbol{x}, \boldsymbol{x}^*)$ is the covariance between the observed point vector $\boldsymbol{x}$ and the prediction point vector $\boldsymbol{x}^*$, and σ_n^2 accounts for the noise.

Given this joint distribution, the conditional distribution of the prediction vector $\boldsymbol{y}^*$ can be derived as:

$$\boldsymbol{y}^* | \boldsymbol{y} \sim \mathcal{GP}(\mu(\boldsymbol{x}^*), \sigma^2(\boldsymbol{x}^*)) \tag{10}$$

with

$$\mu(\boldsymbol{x}^*) = \text{Cov}(\boldsymbol{x}^*, \boldsymbol{x}) \left(\text{Cov}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \mathbf{I} \right)^{-1} \boldsymbol{y} \tag{11}$$

$$\sigma^2(\boldsymbol{x}^*) = \text{Cov}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \text{Cov}(\boldsymbol{x}^*, \boldsymbol{x}) \left(\text{Cov}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \mathbf{I} \right)^{-1} \text{Cov}(\boldsymbol{x}, \boldsymbol{x}^*) \tag{12}$$

These equations enable predictions while accounting for noisy observations, with the noise variance σ_n^2 incorporated into the covariance matrix.

In this research, the Matern 2.5 kernel is chosen as the kernel function for covariance computation due to its flexibility in modeling both smooth and rough functions. The Matern 2.5 kernel is defined as:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha \left(1 + \frac{\sqrt{5}d}{\lambda} + \frac{5d^2}{3\lambda^2}\right) \exp\left(-\frac{\sqrt{5}d}{\lambda}\right) \tag{13}$$

where $d = \|\boldsymbol{x} - \boldsymbol{x}'\|$ is the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$, α is the scaling factor, and λ is the length scale parameter.

As shown above, the Gaussian process model involves three key hyperparameters: the scaling factor α, the length scale λ, and the noise variance σ_n^2. To optimize these hyperparameters, the marginal likelihood of the Gaussian process is maximized, which can be expressed as:

$$Pr(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}) = \mathrm{Norm}_{\boldsymbol{y}}(\mu, \mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2 \mathbf{I}) \tag{14}$$

where $\boldsymbol{\theta} = (\alpha, \lambda, \sigma_n^2)$ represents the hyperparameters, $\boldsymbol{y}$ is the observed data, $\boldsymbol{x}$ is the input data, μ is the mean function, and $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x})$ is the covariance matrix. Note that the mean function μ is not included in the marginal likelihood, as it is a constant.

Maximizing the marginal likelihood helps to find the set of hyperparameters that best fit the data while balancing model complexity. Even if the hyperparameters are not perfectly optimized, the Gaussian process can still provide a reasonable approximation of the objective function. A simple and effective approach to hyperparameter optimization is grid search, which can identify a good set of hyperparameters.

3.5.3. Acquisition function based on upper confidence bound (UCB)

In BO, the acquisition function plays a critical role in determining the next observation point for evaluation. These functions guide the trade-off between exploration (sampling in areas of high uncertainty) and exploitation (sampling in areas expected to yield high values based on prior observations). In SATM, the UCB acquisition function [31] is chosen for its effectiveness and ease for implementation. It combines the mean prediction $\mu(x^*)$ defined in Eq. (11) and the uncertainty estimate $\sigma(x^*)$ defined in Eq. (12), and is defined as:

$$UCB(x^*) = \mu(x^*) + \beta_{\mathrm{ucb}}^{1/2} \sigma(x^*) \tag{15}$$

where $\mu(x^*)$ represents the predicted mean (exploitation), $\sigma(x^*)$ represents the predicted uncertainty (exploration), and $\beta_{\mathrm{ucb}}^{1/2}$ is a tunable parameter that controls the balance between exploration and exploitation. Here, UCB serves as the specific acquisition function a for guiding the optimization process. UCB considers both the predicted value and uncertainty to select the next observation point, thereby effectively exploring areas of uncertainty while refining regions with higher predicted values. As shown in Algorithm 3, by maximizing the acquisition function, herein UCB, the next observation point is determined, and the optimization process continues iteratively. In SATM, Monte Carlo method is used to maximize the acquisition function, which is easy and efficient for edge deployment.

4. Edge deployment strategy

The primary objective of SATM is to facilitate onboard adaptive fine-tuning of trigger parameters to optimize the F_β score. Given the resource constraints inherent in edge devices, the deployment strategy must prioritize computational efficiency and lightweight implementation without compromising reliability or accuracy. The overarching philosophy is to leverage the digital twin to offload computationally intensive tasks, thereby reducing the computational burden during the actual deployment phase. This approach ensures a high-performance starting point for lightweight online fine-tuning. The strategy comprises two critical phases: (1) utilizing the digital twin to pre-train a surrogate model and optimize trigger parameters, and (2) deploying the pre-optimized model and parameters to edge devices for onboard fine-tuning, enabling real-time adaptive adjustments with minimal computational overhead. Section 3 introduces the building blocks of SATM from a spatial perspective, while this section focuses on the temporal perspective, outlining the deployment pipeline in chronological order.

4.1. Stage I: Pre-deployment optimization

The first stage focuses on pre-deployment optimization using the digital twin to establish a foundation for onboard fine-tuning. The digital twin provides a virtual environment for optimizing the surrogate model, training neural networks, and generating artificial signals for real-time adjustments during deployment. This stage ensures that the system is well-prepared for efficient onboard fine-tuning.

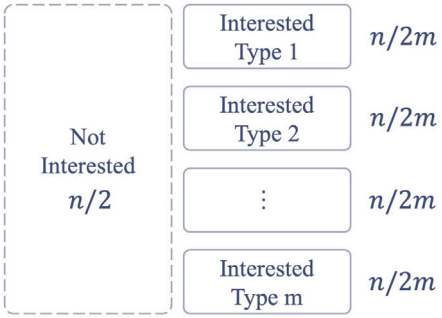


Fig. 7. Schematic illustration of artificial event distribution generation with proportional allocation: $n/2$ for non-interested events and $n/2m$ for each of the m interested event types.

4.1.1. Digital twin and pre-optimization

The construction of the digital twin can be considered from two perspectives: spatial and temporal. From the spatial perspective, all building blocks of SATM, as detailed in Section 3, must be properly established before integration into a unified framework. This section emphasizes the temporal perspective, focusing on how the digital twin simulates or enhances real-world scenarios in a time-sequenced manner, such as controlling the frequency distribution of events or generating rare events that are difficult to observe in the real world. It is important to note that the key point of the digital twin here is to accelerate optimization and provide functionalities that may not be feasible in real-world deployments, rather than to replicate them exactly.

As discussed in Section 2.4, the uncertainty in event distribution poses significant challenges, which can be addressed through digital twin technology by generating artificial event distributions and providing sufficient data for rare events to support optimization and neural network training. To facilitate virtual optimization, we propose a straightforward method for constructing artificial event distributions: for a total of n events in each SATM performance evaluation, half $n/2$ are designated as events of no interest, while the remaining half $n/2$ represent events of interest. If there are m types of events of interest, each type is allocated $n/2m$ events, as illustrated in Fig. 7. The event can be characterized by maximum signal amplitude, and the distribution of the maximum signal amplitude can follow a certain distribution, such as uniform or Gaussian. Compared to real-world cases, the artificial event distribution is more controllable and can be adjusted to cover a wide range of scenarios, especially for rare events that are difficult to observe in practice. If possible, the data distribution can be corrected based on real-world observations.

One critical issue that requires special attention is ensuring the range of signal amplitudes in the artificial distribution aligns with real-world data, particularly for events of no interest, which in this study are primarily represented by ambient vibrations. This alignment is crucial because a major source of misclassification arises when events of interest are obscured by background noise. Consequently, accurately representing noise levels is paramount in the optimization process and must be carefully addressed to ensure the reliability and effectiveness of the digital twin in simulating real-world conditions. To some extent, the noise level determines whether the results from the digital twin can be effectively transferred to real-world applications. In short, for pre-optimization, the noise level should align well with real-world data, while the events of interest can be more flexible to cover a wide range of scenarios.

Compared to the real-world cases, the virtual environment can generate much more events for pre-deployment optimization, enabling the surrogate model to be trained on a more comprehensive dataset and achieve better performance. Since the pre-optimization is conducted in a controlled and resource-rich environment, the computation can be more intensive, so that the surrogate model can be well-trained to provide a high-quality starting point for the onboard fine-tuning. Actually, it is quite likely that the optimal configuration found in the digital twin can also be optimal in the real-world cases, which is the ultimate goal of the pre-deployment optimization.

4.1.2. Artificial datasets and neural networks training

Building upon the digital twin framework introduced in Section 4.1.1, this section focuses on leveraging the digital twin to generate datasets to facilitate real-world fine-tuning and neural network training for onboard inference. The digital twin enables the characterization of SATM performance based on artificial event distributions, ensuring consistency between simulated and real-world scenarios. It is important to maintain similar event distributions across both environments to ensure the applicability of results from virtual environment to real-world deployments. However, unlike the digital twin, real-world optimization typically operates on smaller datasets, as its primary objective is parameter fine-tuning rather than coarse surrogate model training. Given the ease of capturing ambient vibrations in real-world settings, the artificial datasets for real-world fine-tuning should concentrate exclusively on events of interest.

As outlined in Section 3.4, the system employs two neural networks: a CNN for event labeling and a DNN for recall estimation. The training of these networks is conducted in this stage using data generated by the digital twin. For CNN training, the digital twin generates a sufficient number of diverse event types to ensure robust model training. DNN training, however, presents a more complex scenario, as its input comprises not raw signals but triggering parameters, noise levels, and recall values. To construct the

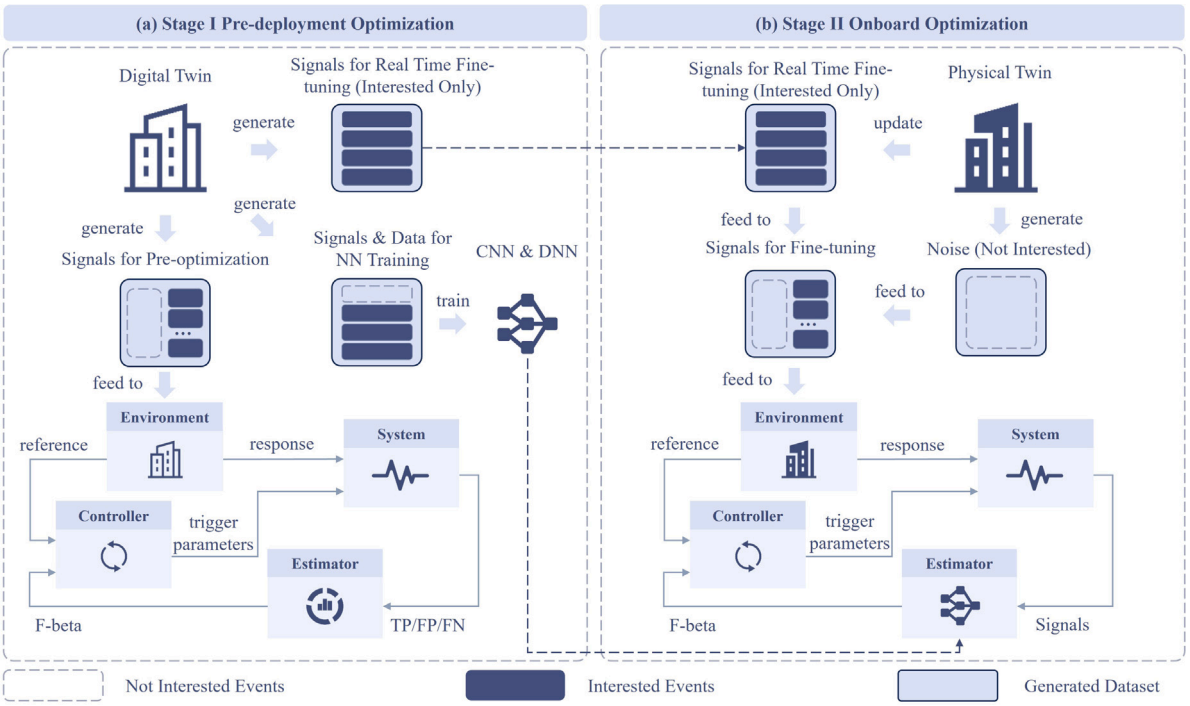


Fig. 8. Two-stage optimization strategy: (a) Stage I generates three datasets via digital twin: pre-optimization, fine-tuning facilitation, and NN training, with math-based performance metrics. (b) Stage II combines digital twin events and real-world noise, progressively updating events via real environment, using neural networks for performance calculation.

DNN training dataset, the process involves generating events of interest, mixing them with ambient vibration data at varying noise levels, and calculating recall values across different triggering parameters. This dataset is then used to train the DNN for recall estimation. Importantly, the event distribution for DNN training must align with the artificial event distribution established in the digital twin framework to maintain consistency and ensure reliable performance in real-world applications.

4.2. Stage II: Onboard optimization

In SHM practice, most wireless sensing devices are micro-controller unit (MCU) level devices, featuring limited computational resources, memory, and power supply. To support the deployment of the proposed SATM, the sensor should have a large volume storage media like SD card. For neural network deployment, the trained CNN and DNN models need to be quantized and optimized for MCU execution. This typically involves converting floating-point weights to 8-bit or 16-bit integers, pruning redundant connections, and optimizing the model architecture for memory-constrained environments. Preferably, the MCU vendor should have a good support for onboard signal processing and AI, for example, STM32Cube-AI from STMicroelectronics, ESP-DSP [38] and ESP-DL [39] from Espressif. These frameworks provide optimized inference engines, quantization tools, and hardware acceleration for common neural network operations like convolutions and matrix multiplications. Otherwise, the user should have the ability to port the AI model to the MCU [40,41]. This porting process typically involves using open-source tools like TensorFlow Lite Micro, CMSIS-NN, or X-CUBE-AI to convert pre-trained models into C/C++ code optimized for the target MCU architecture. Most importantly, the SATM should be incorporated into the sensor working pipeline, from sensing, data processing, to performance evaluation.

As elaborated in Section 4.1.2, for onboard optimization, considering the uncertain real world event distribution and consistency with the surrogate model in stage I, the artificial event distribution and associated datasets are adopted. Note that, the ambient vibration, i.e., the noise part can use real world data as they can be easily captured and matter a lot for event detection.

In nature, the performance evaluation for onboard optimization is based on the synthesized dataset following artificial event distribution which is consistent with the digital twin. To make it more realistic and reliable, the datasets can be gradually replaced with real world data. As mentioned, the noise data can first be constructed with real world data all at once, and then the interested event data can be gradually replaced as more real world data is collected. This is a continuous improvement process, which can be conducted as a side quest without interrupting the main quest for parameter fine-tuning.

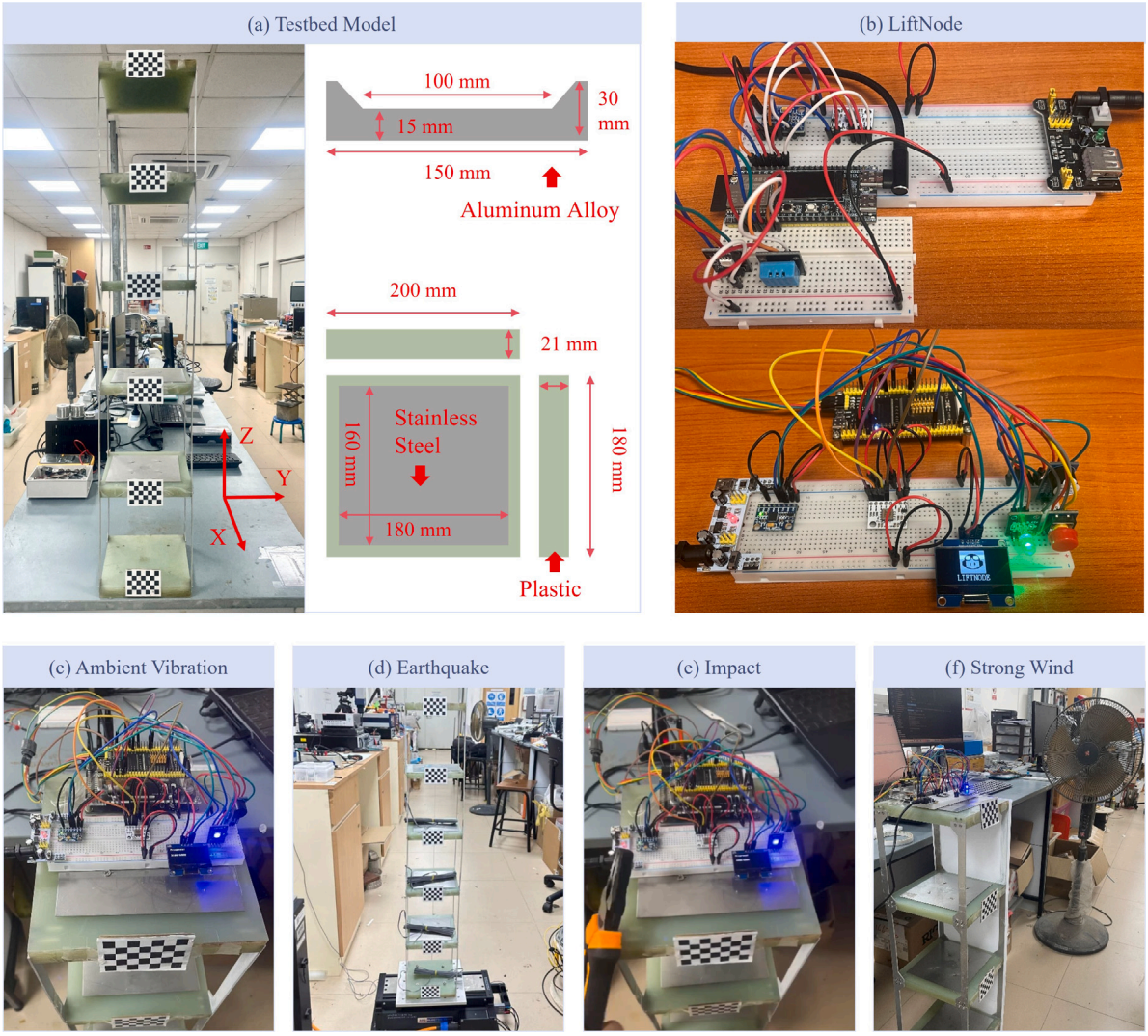


Fig. 9. Multi-storey structural vibration testbed and experimental configurations: (a) 5-storey aluminum-plastic composite model with dimensional specifications, (b) LiftNode wireless sensor node for acceleration sensing, (c) ambient vibration monitoring setup, (d) earthquake simulation using shake table excitation, (e) impact testing configuration, (f) strong wind simulation using industrial fan.

5. Deployment and validation

5.1. Experimental setup

As illustrated in Fig. 9, a 5-storey structural model serves as the testbed for monitoring and evaluation, and among the 3 axes, Y-axis is selected for evaluation as it features the most significant response. The testbed consists of a 5-storey aluminum-plastic composite structural model with precise dimensional specifications, designed to simulate real-world building behavior under various loading conditions. The model features a modular design with standardized floor heights and structural elements, allowing for consistent and repeatable experimental conditions. Each floor is constructed using green resin-based panels with a density of approximately 1.2 g/cm^3 and a mass of approximately 900 g per floor. This study utilizes LiftNode, a low-power, cost-effective, and high-performance IoT sensing platform actively developed by the Laboratory of Intelligent Infrastructure (LIFT) at Nanyang Technological University (NTU), for data collection and edge intelligence. To ensure cross-platform compatibility, LiftNode is prototyped with two different main control boards: one based on STM32 and the other on ESP32. In this study, validation is conducted using the ESP32-based LiftNode, which features a dual-core 32-bit MCU running at 480 MHz, 8 MB of RAM, and 16 MB of flash memory. The edge computing deployment leverages the TinySHM middleware, which is being developed by the LIFT laboratory. TinySHM is a computational library specifically designed for enabling edge intelligence on resource-constrained MCUs. Architecturally, from bottom to top, TinySHM supports various toolboxes (e.g., time management, configuration), mathematical

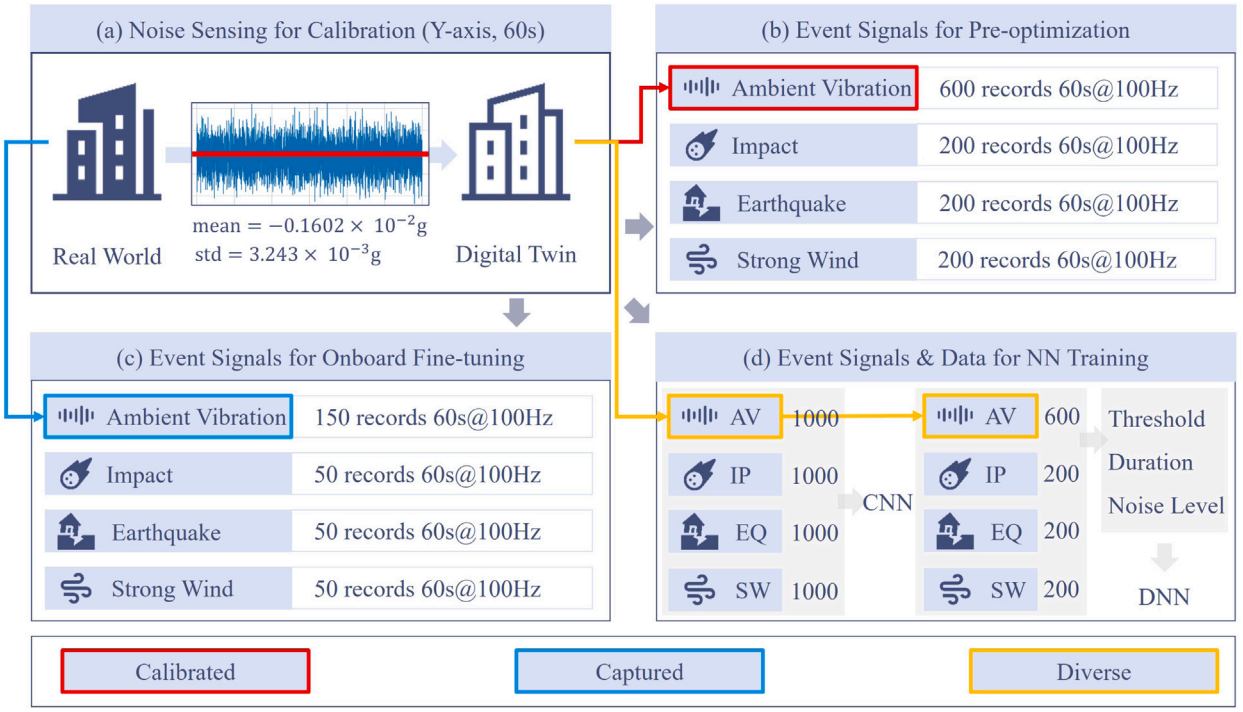


Fig. 10. Signal synthesizing (60s, 100 Hz): (a) noise level calibration — calculation of mean and standard deviation, (b) generation for pre-optimization — total 1200, (c) generation for onboard optimization — total 300, (d) neural network training — CNN (4000) and DNN (1200).

operations (e.g., vector, matrix computation), digital signal processing (e.g., convolution, resampling, wavelet decomposition, SHM-specific functions), and AI functionalities (e.g., neural network inference). Based on TinySHM, the SATM process was deployed on LiftNode for computation. During validation, LiftNode is positioned atop the testbed model, as shown in subplot (c).

Various types of events are examined in this study, each generated using different methods to ensure comprehensive coverage of real-world structural excitation scenarios. For ambient vibration sensing, no additional equipment or intervention is required (subplot (c)), as the natural environmental vibrations from building systems, human activities, and external sources provide sufficient baseline data for noise characterization. Earthquake simulations are performed using a shake table excitation system (subplot (d)), which can simulate earthquakes according to computer input signals with programmable frequency content, amplitude, and duration to replicate various seismic events ranging from minor tremors to moderate earthquakes. Impact events are induced with suitable tools like hammers, pliers, etc. (subplot (e)), through manual striking to simulate structural impacts from various sources such as falling objects, vehicle collisions, or construction activities. The impact locations are not limited to specific points; during the experimental process, the probability distribution of impact positions is maintained as uniform as possible across all floors to ensure comprehensive coverage of the structural response. Strong wind conditions are replicated using a common oscillating floor fan (subplot (f)), positioned approximately 1.0 m away from the model structure. During the simulation process, the oscillating function of the fan is activated, and the distance between the fan and the model structure is manually adjusted to simulate the non-stationary characteristics of natural wind loads. To enhance the strong wind simulation effect, paper was used to cover the model on the side near the fan to increase the wind-receiving area. Each simulation method is designed to provide reasonable experimental conditions, with basic parameters recorded to facilitate experimental reproducibility. More technical details for validation can be found in Table 2.

5.2. Deployment procedures

5.2.1. Stage I: pre-deployment optimization

The pre-deployment optimization stage, as discussed in Section 4.1.1 and illustrated in Fig. 8(a), primarily occurs in a virtual environment and encompasses three major tasks that leverage the digital twin framework for comprehensive signal generation and model preparation. While this stage is predominantly conducted in simulation, certain critical information can be pre-calibrated from the actual physical environment and then utilized for digital twin generation, ensuring enhanced accuracy and real-world applicability. These tasks are designed to establish a robust foundation for subsequent real-world deployment.

The three primary tasks include:

1. Pre-optimization parameter tuning and corresponding dataset generation,

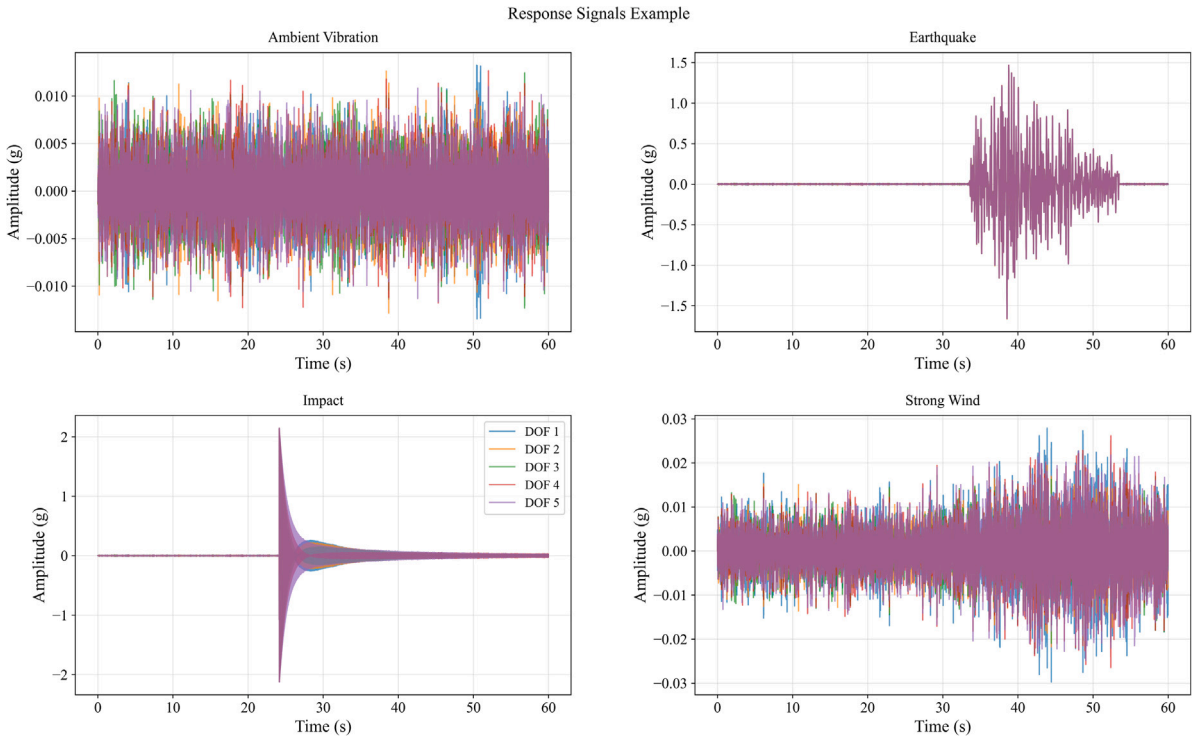


Fig. 11. Examples of synthesized data for 4 event types: ambient vibration, impact, earthquake, and strong wind.

2. Synthetic dataset creation for real-world fine-tuning applications, and
3. Neural network training and associated dataset preparation.

To ensure consistent evaluation standards, the event signal generation process in this study adheres to a standardized configuration characterized by a 60-second signal duration and a 100 Hz sampling frequency. It is assumed that each signal segment contains at most one event of interest, which simplifies computational processing and analysis. This assumption is considered safe because the primary function of the triggering mechanism is to distinguish events of interest from noise, and it is unlikely that multiple events within a single signal segment would be misclassified as noise.

The first task focuses on establishing a robust foundation for real-world deployment through comprehensive parameter optimization and dataset generation. This task begins with a preliminary sensing campaign to capture real-world ambient vibration data, which serves as the baseline for noise level calibration in the digital twin framework, as illustrated in Figs. 10 (a) and (b). The calibration process specifically involves calculating the mean and standard deviation of white noise from the captured ambient vibration data. These statistical parameters are crucial for inputting appropriate values into the Gaussian process when generating synthetic white noise, thereby enhancing the precision of the digital twin. This calibration process is straightforward for IoT sensors, as edge computing libraries can directly compute and return these statistical parameters to the cloud, or it can be accomplished through offline manual collection and analysis. To ensure comprehensive coverage of different event scenarios, 200 signal clips were generated for each type of event of interest, resulting in a total of 600 event signals. Correspondingly, 600 ambient vibration signals using Gaussian white noise with calibrated parameters were generated to maintain dataset balance, which is crucial for effective surrogate model training as demonstrated in Fig. 10(b).

This task constitutes the initial phase of the Bayesian optimization process, where 15 observation points were strategically sampled from the parameter search space defined by triggering threshold and duration parameters. The threshold parameter spans from 0 to the maximum amplitude of the ambient vibration captured during pioneer sensing, while the duration parameter ranges from 2 to 10 sampling points. The surrogate model was subsequently trained on this comprehensive dataset for 50 optimization iterations to fine-tune the triggering parameters. Given that Stage I emphasizes exploration for pre-optimization purposes, the balance factor in the UCB acquisition function was set to 2 to encourage broader parameter exploration, as detailed in Table 2. Additionally, to accelerate convergence, a performance bonus factor of 1.1 was applied to samples achieving both precision and recall values exceeding 0.9.

The second task is designed to create synthetic datasets specifically tailored for real-world fine-tuning applications. While maintaining the same statistical distribution as the first task, this task introduces two key modifications: (1) in the digital twin, we only generate events of interest, and then in Stage II, we collect uninteresting events (background noise) on-site to form the dataset for fine-tuning, as depicted in Fig. 10(c), with the actual collection process occurring during Stage II deployment; and (2)

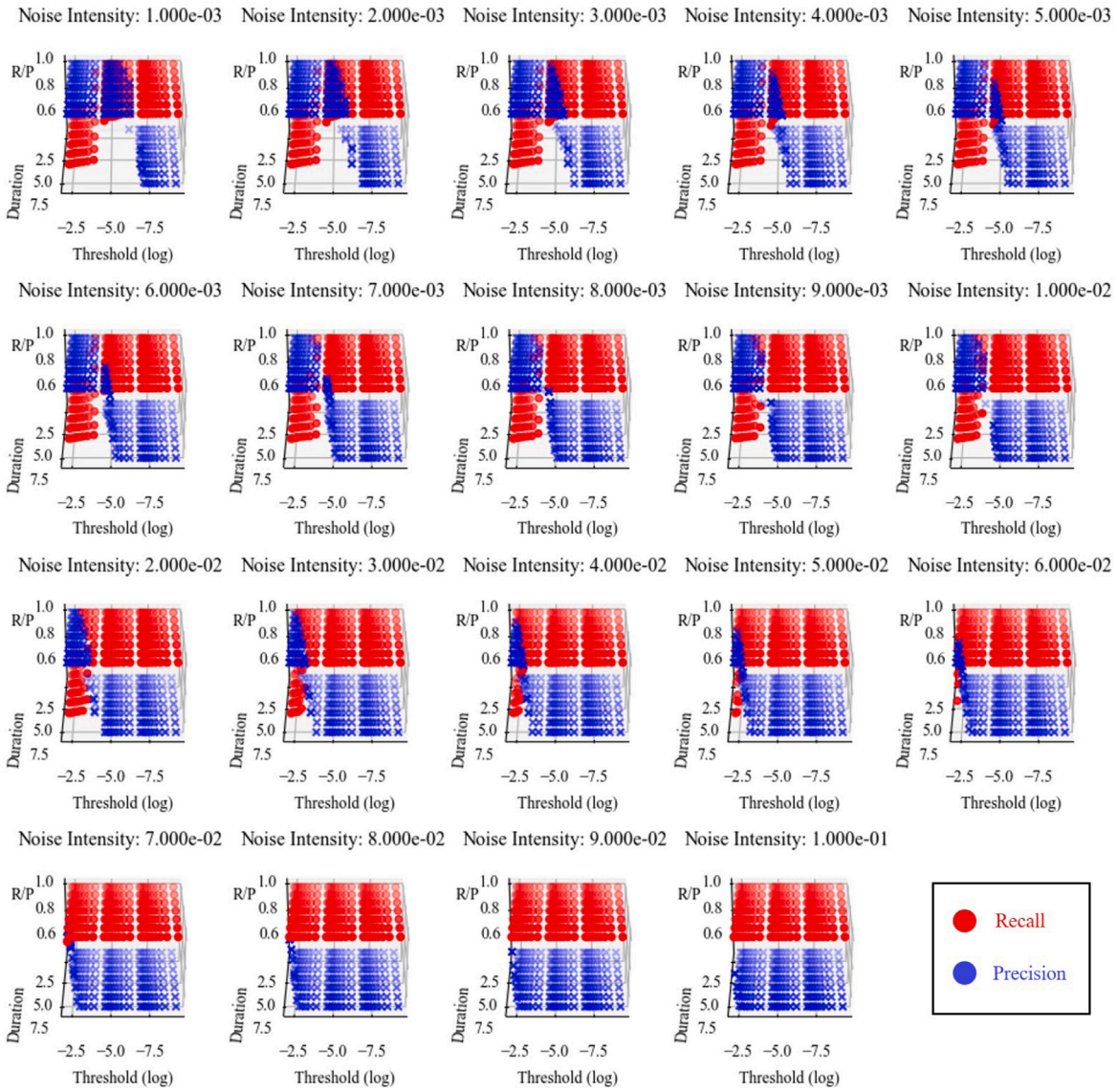


Fig. 12. Dataset for DNN training: recall and precision distributions across the search space under different noise levels.

the dataset scale is significantly reduced to accommodate real-world computational constraints. Specifically, 50 signal clips are generated for each type of event of interest, complemented by 150 ambient vibration clips, resulting in a compact dataset of 300 total clips.

The third task encompasses the comprehensive training of neural networks and the preparation of associated datasets. As illustrated in Fig. 10(d), the CNN is trained using a meticulously balanced dataset containing 1000 signal clips for each event type, with representative examples showcased in Fig. 11. In contrast, the DNN follows a distinct training approach, where it is trained on a specialized dataset of recall values rather than direct event signal data. These recall values are systematically calculated across various combinations of triggering thresholds, durations, and noise levels to ensure robust model generalization.

To construct the recall value dataset, a comprehensive dataset of 200 signals per event type (totaling 600 clips) was generated and subsequently mixed with ambient vibration data at varying noise levels to simulate diverse real-world operational scenarios. Prior to recall value calculation, a well-defined search space for triggering parameters and noise levels was established, utilizing logarithmic grid scaling to ensure comprehensive parameter coverage, as detailed in Fig. 12 and Table 2. The training performance of both neural networks is comprehensively documented in Fig. 14, while the CNN's classification accuracy is further validated through the confusion matrix presented in Fig. 13.

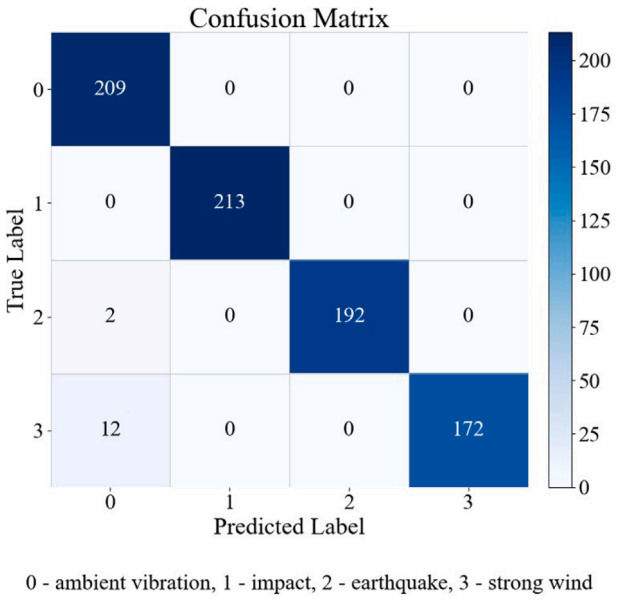


Fig. 13. Confusion matrix of the CNN network characterizing its performance in classifying different event types.

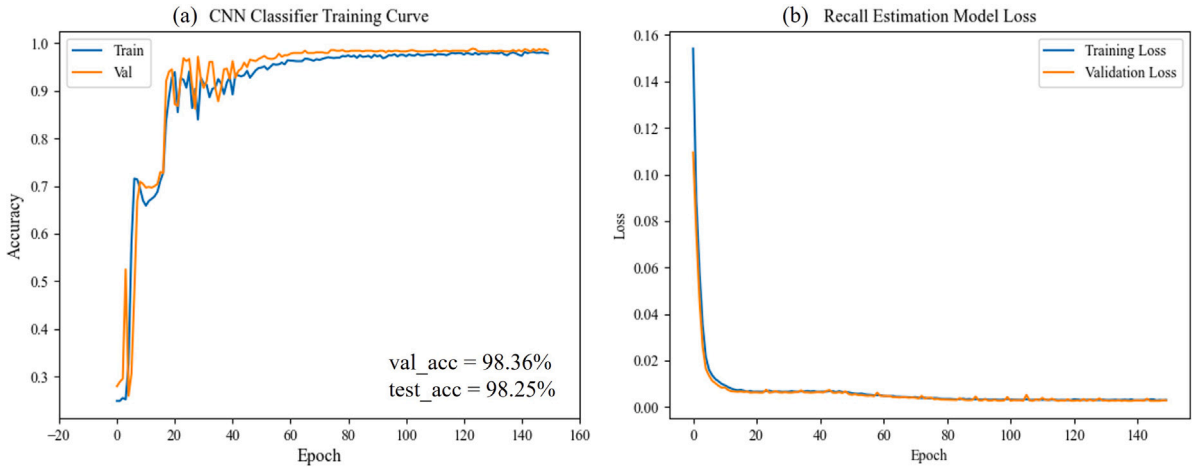


Fig. 14. Neural network training results: (a) CNN training results, (b) DNN training results.

5.2.2. Stage II: onboard optimization

In Stage II, the pre-optimized surrogate model, triggering parameters, synthesized dataset, and neural networks are deployed to the edge device for real world fine-tuning. The validation is performed on a LiftNode powered by ESP32 MCU, featuring official support for signal processing (ESP-DSP) and deep learning (ESP-DL). Moreover, LiftNode is equipped with TinySHM, a distributed edge intelligence enabling framework also developed by LIFT at NTU for IoT-based SHM, to facilitate signal processing and AI deployment.

For dataset processing, the synthesized event of interest signals were first stored onboard, using TF card storage. Afterwards, the ambient vibration data was automatically captured by programming and then stored in the same TF card to mix with the previously stored event of interest signals to form the dataset for real world fine-tuning as shown in Fig. 10(c). Note that the data captured was first calibrated by removing the mean value obtained from the pioneer sensing before downstream signal processing.

5.3. Smart adaptive trigger sensing results

The results of the proposed SATM are comprehensively presented in Table 3 and Figs. 15 and 16. Table 3 shows the performance metrics for both Stage I (pre-optimization) and Stage II (onboard fine-tuning) of the SATM implementation. Figs. 15 and 16 provide visual representations of the optimization results, with Fig. 15 showing the spatial positions in the parameter search space and Fig.

Table 2
Optimization configuration for virtual environment optimization.

Parameter	Description	Value
Ambient Vibration Intensity	The noise level range (times to captured noise level), uniform distribution.	0.9 to 1.1
Earthquake Peak Value	The amplitude of the maximum earthquake response, uniform distribution.	0.1 g to 3.0 g
Impact Peak Value	The amplitude of the maximum impact response, uniform distribution.	0.1 g to 3.0 g
Strong Wind Peak Value	The amplitude of the maximum strong wind response (times to maximum ambient vibration), uniform distribution.	1.5 to 2.5
β	The balance factor in Eq. (1).	5
α	The scaling factor in Eq. (13).	1
λ	The length scale factor in Eq. (13).	1
σ_n^2	The parameter σ_n^2 in Eq. (14).	0.1
Initial Observation Number	The number of initial observation points as described in Algorithm 3.	15
Iteration Number of Virtual Optimization	The number of iterations for virtual environment optimization as described in Algorithm 3.	50
Iteration Number of Real-world Optimization	The number of iterations for real-world optimization as described in Algorithm 3.	20
β_{ucb1}	Parameter β_{ucb} in UCB acquisition function to balance exploitation and exploration. (pre-optimization stage)	2
β_{ucb2}	Parameter β_{ucb} in UCB acquisition function to balance exploitation and exploration. (onboard deployment stage)	0.5
f_{bonus}	Bonus factor for F_β score when precision and recall are higher than 0.9.	1.1
τ_{lb}	lower bound for triggering threshold.	0
τ_{ub}	upper bound for triggering threshold (twice the average absolute amplitude of the ambient vibration or the noise).	0.01706
d_{lb}	lower bound for triggering duration, integer	2
d_{ub}	upper bound for triggering duration, integer	10

Table 3
Performance metrics of the proposed SATM across different optimization stages.

Item	F_β	Precision	Recall
Stage I Pre-optimization	1.0806	95.94%	98.50%
Stage II Onboard Fine-tuning	1.0336	90.48%	94.36%

16 displaying the iteration history of the optimization process. The associated precision and recall values are also presented in Fig. 16.

As revealed by the results, the following observations and conclusions can be drawn. The proposed SATM demonstrates strong performance across both optimization stages, achieving an F_β score of 1.0806 in pre-optimization and 1.0336 in onboard optimization. These results clearly demonstrate the effectiveness of SATM in optimizing triggering parameters for SHM applications. The data distributions observed in both pre-optimization (via the digital twin) and onboard optimization are highly similar, indicating excellent transferability from synthetic to real-world scenarios. Although some discrepancies exist due to inherent differences between artificial and real data, these are expected to diminish as more real-world data becomes available and accumulates. The surrogate model, well-trained within the digital twin environment, enables onboard fine-tuning to converge efficiently. This efficiency is reflected in both the rapid convergence toward high-scoring configurations and the consistent performance achieved during onboard optimization. The two-stage deployment strategy effectively balances exploration and exploitation: the first stage, focused on exploration, exhibits larger metric fluctuations to thoroughly search the parameter space, while the second stage exploits the gathered knowledge to refine the configuration, resulting in a robust and optimized system performance, as shown in Fig. 15 and Fig. 16.

5.4. Comparative analysis

To provide a comprehensive evaluation of different triggering parameter optimization approaches and demonstrate the superiority of the proposed method, this section presents a systematic comparison of three primary methods: (1) conservative manual optimization [1], (2) batch exhaustive parameter search [5,29], and (3) the proposed smart adaptive triggering. It should be noted that since this study does not consider time-varying characteristics such as drift, the predictive parameter setting method mentioned earlier [11] is excluded from this comparison. The comparison encompasses both qualitative characteristics and quantitative performance metrics to offer a holistic assessment of each approach and highlight the advantages of the proposed smart adaptive triggering.

5.4.1. Method description and experimental setup

To provide a comprehensive evaluation of different triggering parameter optimization approaches, we establish a standardized experimental setup for comparing three primary methods. The dataset consists of 300 signal segments, each with a duration of 60 s and a sampling frequency of 100 Hz. Specifically, the dataset includes 150 ambient vibration (white noise) segments and 150 event

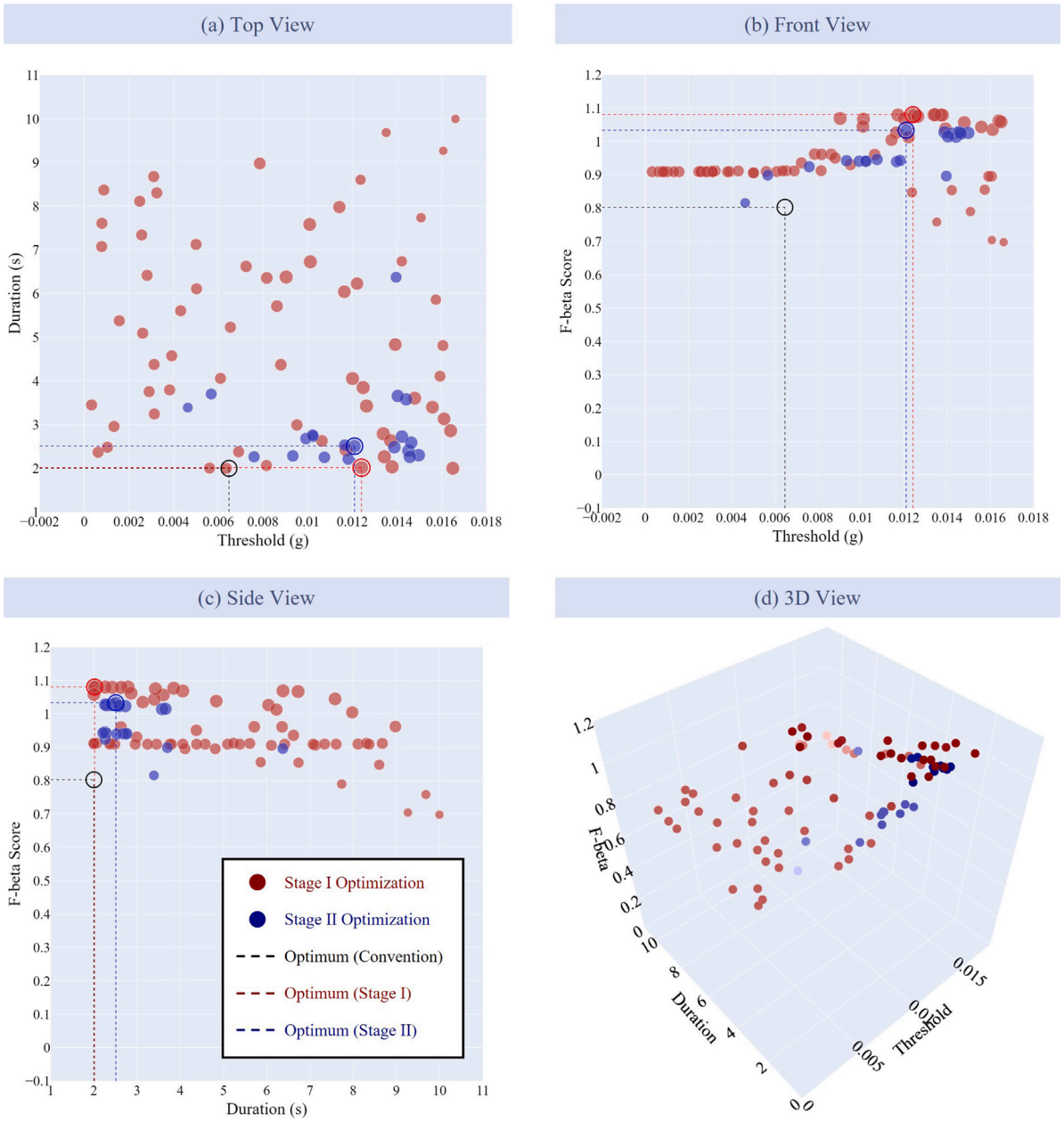


Fig. 15. Laboratory test results of stage I — pre-optimization (red) and stage II — onboard fine-tuning (blue): (a) top view, (b) front view, (c) side view, (d) 3D view.

segments, with 50 segments each for impact, earthquake, and strong wind events. Each signal segment contains at most one event. The parameter search space is defined as follows: the triggering duration parameter is discretized as integer values from 2 to 10, while the threshold parameter is discretized from 0 to 0.01706. The performance metrics reported in this comparison, including F_β scores, precision, and recall, represent the raw experimental results without any acceleration factors or scaling coefficients.

Conservative Manual Optimization: This traditional approach relies on empirical parameter setting based on pioneer sensing data, followed by manual adjustments by human operators. In our experimental implementation, the triggering parameter was set to 0.00648 g (twice the standard deviation of ambient vibration captured in pioneer sensing) with a duration of 2 sampling points, as can be found in Fig. 15. This conservative strategy prioritizes recall over precision, ensuring comprehensive event capture but often resulting in high false positive rates. While simple and reliable, this method lacks adaptability and requires continuous human intervention, making it unsuitable for long-term autonomous deployment.

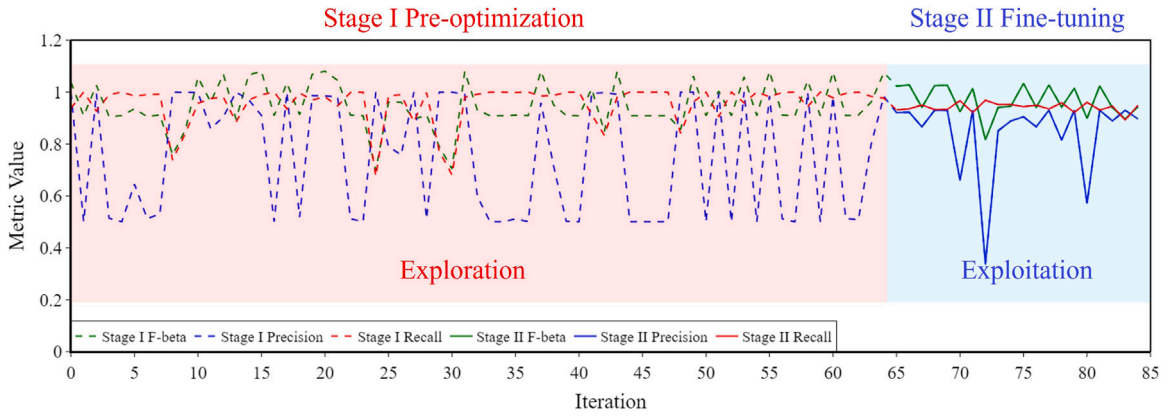


Fig. 16. Time history of metrics during optimization iteration: stage I pre-optimization focusing on exploration and stage II onboard fine-tuning focusing on exploitation.

Table 4
Qualitative comparison of triggering parameter optimization methods.

Evaluation criteria	Conservative manual [1]	Batch exhaustive [5,29]	Smart adaptive (Proposed)
Environmental Adaptability	Poor	Poor	Excellent
Real-time Learning	None	None	Yes
Autonomous Operation	None	None	Yes
Cloud Dependency	Medium	Heavy	None/Low
Communication Overhead	Low	Massive	Low
Feedback Time	Longest	Medium	Shortest
Maintenance Requirements	High	Medium	Low
Edge Device Suitability	Medium	Poor	Good

Batch Exhaustive Parameter Search: This data-driven approach collects extensive historical data and performs comprehensive analysis on the entire dataset to exhaustively search for optimal parameter combinations [5,29]. In practice, exhaustive search over all possible parameter combinations is computationally infeasible, requiring discretization of the continuous parameter space into a grid. The grid spacing cannot be too fine due to computational constraints, so we set the threshold parameter spacing to 0.0001 and use integer values from 2 to 10 for the duration parameter. However, this method heavily relies on cloud computing resources, incurs significant communication overhead, and requires long feedback times due to data accumulation requirements [11].

Smart Adaptive Triggering (Proposed): This novel approach integrates feedback control, Bayesian optimization, and digital twin technology for continuous, autonomous parameter adaptation. The two-stage deployment strategy combines pre-optimization using digital twin data with onboard fine-tuning using real-world observations. This approach treats the parameter space as continuous before discretization, enabling more sophisticated optimization compared to direct grid-based methods. The system operates entirely on edge devices, eliminating communication overhead while maintaining practical computational requirements. For detailed implementation and parameter settings, please refer to the previous sections.

5.4.2. Qualitative assessment

The qualitative assessment reveals significant differences in the fundamental characteristics of each approach. The conservative manual method, while simple to implement, suffers from poor environmental adaptability and high maintenance requirements due to its reliance on human intervention [1]. The batch exhaustive method provides better objectivity in parameter determination but suffers from massive communication overhead, enormous computational requirements, heavy cloud dependency, and long feedback time due to extensive data accumulation requirements [29]. In practice, exhaustive search is often infeasible for this method, particularly when the parameter space is continuous, leading to grid search implementations that may increase grid spacing for efficiency, potentially missing optimal values. In contrast, the proposed SATM demonstrates superior adaptability, learning capabilities, and autonomous operation through closed-loop feedback control and Bayesian optimization, which can fully leverage previous observation results to reduce the required number of iterations, making it particularly suitable for dynamic monitoring environments.

Table 4 presents a comprehensive comparison of the three methods across various qualitative dimensions that significantly impact practical implementation and long-term viability. It should be noted that due to the distinct operational characteristics of these methods, the following quantitative comparison is not entirely fair. For example, the batch exhaustive method assumes cloud computing, while the proposed SATM operates on sensor devices, leading to different computational constraints and resource availability.

Table 5
Quantitative comparison of triggering methods.

Performance metrics	Conservative manual [1]	Batch exhaustive [5,29]	Smart adaptive
Detection Performance			
F_β Score	0.8025	0.9350	0.9396
Precision (%)	50.00	90.56	90.84
Recall (%)	100.00	93.84	94.36
Efficiency Metrics (approximate values)			
Computation Overhead (Relative)	1.0 (cloud)	15354 (cloud)	200 (edge)
Communication Overhead (MB)	1.5	15	0
Optimal Parameter Settings			
Threshold (g)	0.0065	0.0120	0.0121
Duration (samples)	2	3	3 (obtained from 2.508)

5.4.3. Quantitative performance comparison

Building upon the experimental setup established in the previous subsection, Table 5 presents a comprehensive quantitative comparison of the three methods across detection performance and efficiency metrics. The results demonstrate distinct performance patterns that highlight the fundamental trade-offs between different optimization approaches.

The quantitative results reveal clear performance trade-offs across the three methods, providing valuable insights into the evolution of triggering approaches in wireless SHM systems. The conservative manual method achieves perfect recall (100%) but suffers from low precision (50%), exemplifying the classic engineering dilemma between comprehensive event capture and false alarm reduction. This reflects the fundamental limitation of static parameter configurations in dynamic environments, as evidenced by its conservative threshold setting (0.0065 g) and minimal duration requirement (2 samples). The batch exhaustive method demonstrates strong detection performance ($F_\beta = 0.935$) but requires massive computational resources (15,354 times relative overhead) and substantial communication (15 MB), highlighting the scalability challenges inherent in cloud-based optimization approaches. Despite its computational intensity, this method achieves optimal parameters of 0.0120 g threshold and 3 samples duration. The proposed SATM achieves the best overall performance ($F_\beta = 0.9396$) while maintaining practical computational requirements (200 times relative overhead) and zero communication overhead through edge computing, demonstrating that real-time adaptation can be achieved without sacrificing computational efficiency when properly architected. Notably, the SATM's optimal parameters (0.0121 g threshold and 3 samples duration, obtained from 2.508) are remarkably close to those of the batch exhaustive method, with nearly identical performance metrics, yet achieved with only 1.3% of the computational overhead. This demonstrates the effectiveness of continuous optimization in finding near-optimal solutions and highlights the SATM's ability to achieve exhaustive search-like performance with minimal computational resources.

These results challenge several traditional assumptions in wireless SHM system design. First, they demonstrate that high performance does not necessarily require centralized processing, as evidenced by the SATM's superior performance-to-resource efficiency compared to cloud-based approaches. Second, the digital twin approach effectively bridges the gap between theoretical optimization and practical deployment constraints, enabling sophisticated parameter adaptation within resource-limited edge devices. Third, the results suggest that future wireless SHM systems should prioritize distributed intelligence and local adaptation over centralized processing, particularly for applications requiring real-time responsiveness and energy efficiency.

The comparison reveals a fundamental design philosophy spectrum in triggering parameter optimization. The conservative manual method represents one extreme, where minimal computational effort results in suboptimal performance due to static parameter configurations. The batch exhaustive method represents the opposite extreme, where excessive computational resources achieve excellent detection performance but at the cost of impractical overhead and scalability limitations. The proposed SATM represents an optimal balance between these extremes, achieving the best performance while dramatically reducing computational and communication overhead through intelligent design choices. This balanced approach demonstrates that sophisticated optimization can be achieved through careful architectural decisions rather than brute-force computational approaches, pointing toward a paradigm shift from centralized to distributed intelligence in wireless sensing systems.

6. Conclusion

This study proposes a smart adaptive triggering mechanism that synergistically integrates feedback loop control, digital twin technology, and Bayesian optimization. Designed to address challenges including limited adaptivity, multi-objective optimization, unknown event distributions, absence of ground truth, partial observability, and high observation costs, the SATM comprises four fundamental components: the environment, system, estimator, and controller. The mechanism operates in two distinct phases: first, a digital twin-based pre-optimization phase to establish a good initial configuration, followed by an onboard fine-tuning phase to refine parameters under real-world conditions. Laboratory experiments demonstrate that the SATM effectively optimizes triggering parameters, yielding real-world performance improvements of approximately 30% in the F_β score compared to conservative manual parameter setting methods, while achieving similar or higher precision with computational overhead reduced by 2–3 orders of magnitude compared to batch exhaustive processing methods.

The comprehensive experimental validation reveals that the SATM successfully addresses the fundamental trade-offs in wireless SHM triggering systems, achieving optimal balance between detection performance and computational efficiency. The two-stage

deployment strategy, combining digital twin pre-optimization with onboard fine-tuning, demonstrates the effectiveness of hybrid optimization approaches in resource-constrained environments. The integration of feedback control, Bayesian optimization, and edge intelligence enables autonomous parameter adaptation without requiring extensive computational resources or communication infrastructure.

This work contributes to the advancement of intelligent wireless sensing systems by demonstrating that sophisticated optimization can be achieved through careful architectural design rather than computational brute force. The SATM framework provides a practical solution for long-term autonomous operation of wireless SHM systems, addressing the critical need for adaptive triggering mechanisms in dynamic monitoring environments. Overall, the SATM exhibits significant potential for enabling automatic and adaptive parameter adjustments in trigger sensing applications across diverse industries.

CRediT authorship contribution statement

Shuaiwen Cui: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Yuguang Fu:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization. **Hao Fu:** Writing – review & editing, Software, Resources, Methodology. **Xiao Yu:** Writing – review & editing, Validation, Resources. **Wei Shen:** Writing – review & editing, Validation, Resources.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve the readability and language of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declaration of competing interest

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

Acknowledgments

The authors want to gratefully acknowledge the financial support of this research from NTU Start-up Grant 03INS001210C120 and MOE AcRF Tier 1 Grants, No. RG146/23 and NTU CoE Dean's Interdisciplinary Grant 2024 (03INS002320C120).

Appendix

For code details, please refer to the GitHub repositories below.
Excitation–Structure–Response Simulation:
https://github.com/Shuaiwen-Cui/Research-Excitation_Structure_Response.git
Smart Adaptive Trigger Sensing:
https://github.com/Shuaiwen-Cui/Research-Smart_Adaptive_Trigger_Sensing.git

Data availability

Data will be made available on request.

References

[1] Y. Fu, L. Zhu, T. Hoang, K. Mechitov, B.F.S. Jr., Demand-based wireless smart sensors for earthquake monitoring of civil infrastructure, in: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2018, vol. 10598, SPIE, 2018-03-27, pp. 245–251, http://dx.doi.org/10.1117/12.2296634.

[2] M. AbdelRaheem, M. Hassan, U.S. Mohammed, A.A. Nassr, Design and implementation of a synchronized IoT-based structural health monitoring system, Internet Things 20 (2022) 100639, http://dx.doi.org/10.1016/j.iot.2022.100639.

[3] A. Fanariotis, T. Orphanoudakis, V. Fotopoulos, Reducing the Power Consumption of Edge Devices Supporting Ambient Intelligence Applications, Information 15 (3) (2024-03) 161, http://dx.doi.org/10.3390/info15030161.

[4] P. Lea, IoT and Edge Computing for Architects, in: Expert Insight, Packt Publishing, Limited, 2020, URL https://books.google.com.sg/books?id=gQpozQEACAAJ.

[5] X. Ge, Q.-L. Han, X.-M. Zhang, L. Ding, F. Yang, Distributed Event-Triggered Estimation Over Sensor Networks: A Survey, IEEE Trans. Cybern. 50 (3) (2020-03) 1306–1320, http://dx.doi.org/10.1109/TCYB.2019.2917179.

[6] A. Mardanshahi, A. Sreekumar, X. Yang, S.K. Barman, D. Chronopoulos, Sensing Techniques for Structural Health Monitoring: A State-of-the-Art Review on Performance Criteria and New-Generation Technologies, Sensors 25 (5) (2025-01) 1424, http://dx.doi.org/10.3390/s25051424.

[7] Y. Fu, T. Hoang, K. Mechitov, J.R. Kim, D. Zhang, B.F. Spencer, Sudden Event Monitoring of Civil Infrastructure Using Demand-Based Wireless Smart Sensors, Sensors 18 (12) (2018-12) 4480, http://dx.doi.org/10.3390/s18124480.

[8] S. Laflamme, F. Ubertini, A.D. Matteo, A. Pirrotta, M. Perry, Y. Fu, J. Li, H. Wang, T. Hoang, B. Glisic, L.J. Bond, M. Pereira, Y. Shu, K.J. Loh, Y. Wang, S. Ding, X. Wang, X. Yu, B. Han, Y. Goldfeld, D. Ryu, R. Napolitano, F. Moreu, G. Giardina, P. Milillo, Roadmap on measurement technologies for next generation structural health monitoring systems, Meas. Sci. Technol. 34 (9) (2023-06) 093001, http://dx.doi.org/10.1088/1361-6501/acd135.

[9] J. Fu, X. Ma, H. Yu, K. Dai, Distributed energy-efficient wireless sensing and information fusion via event-driven and state-rank activation, Wirel. Netw. 30 (4) (2024-05-01) 2697–2711, http://dx.doi.org/10.1007/s11276-024-03691-8.

[10] Y. Ni, X. Liu, C. Yang, Sensor Scheduling for Remote State Estimation with Limited Communication Resources: A Time- and Event-Triggered Hybrid Approach, Sensors 23 (21) (2023-01) 8667, http://dx.doi.org/10.3390/s23218667.

[11] X. Ge, Q.-L. Han, L. Ding, Y.-L. Wang, X.-M. Zhang, Dynamic Event-Triggered Distributed Coordination Control and its Applications: A Survey of Trends and Techniques, IEEE Trans. Syst. Man Cybern.: Syst. 50 (9) (2020-09) 3112–3125, http://dx.doi.org/10.1109/TSMC.2020.3010825.

[12] J. Lynch, H. Sohn, M. Wang, Sensor Technologies for Civil Infrastructures: Volume 2: Applications in Structural Health Monitoring, in: Woodhead Publishing Series in Civil and Structural Engineering, Elsevier Science, 2022, URL https://books.google.com.sg/books?id=_j8yEAAAQBAJ.

[13] W.-J. Yan, D. Chronopoulos, K.-V. Yuen, Y.-C. Zhu, Structural anomaly detection based on probabilistic distance measures of transmissibility function and statistical threshold selection scheme, Mech. Syst. Signal Process. 162 (2022-01-01) 108009, http://dx.doi.org/10.1016/j.ymssp.2021.108009.

[14] X. Yang, C. Fang, P. Kundu, J. Yang, D. Chronopoulos, A decision-level sensor fusion scheme integrating ultrasonic guided wave and vibration measurements for damage identification, Mech. Syst. Signal Process. 219 (2024) 111597, http://dx.doi.org/10.1016/j.ymssp.2024.111597.

[15] M.Z. Sarwar, M.R. Saleem, J.-W. Park, D.-S. Moon, D.J. Kim, Multimetric Event-Driven System for Long-Term Wireless Sensor Operation for SHM Applications, IEEE Sensors J. 20 (10) (2020) 5350–5359, http://dx.doi.org/10.1109/JSEN.2020.2970710.

[16] Y. Zhao, Y. Lian, Event-Driven Circuits and Systems: A Promising Low Power Technique for Intelligent Sensors in AIoT Era, IEEE Trans. Circuits Syst. II: Express Briefs 69 (7) (2022) 3122–3128, http://dx.doi.org/10.1109/TCSII.2022.3180689.

[17] P. Harpe, Y. Shen, H. Li, K. Pelzers, H. Xin, E. Cantatore, Ultra Low Power Event-Driven Sensor Interfaces: 9th IEEE International Workshop on Advances in Sensors and Interfaces, IWASI 2023, Proc. - 2023 9th Int. Work. Adv. Sensors Interfaces, IWASI 2023 (2023) 133–137, http://dx.doi.org/10.1109/IWASI58316.2023.10164620.

[18] N. Bouabdallah, M.E. Rivero-Angeles, B. Sericola, Continuous Monitoring Using Event-Driven Reporting for Cluster-Based Wireless Sensor Networks, IEEE Trans. Veh. Technol. 58 (7) (2009-09) 3460–3479, http://dx.doi.org/10.1109/TVT.2009.2015330.

[19] N. Navabian, S. Beskhyroun, J. Matulich, Development of wireless smart sensor network for vibration-based structural health monitoring of civil structures, Struct. Infrastruct. Eng. 18 (3) (2022) 345–361, http://dx.doi.org/10.1080/15732479.2020.1850801.

[20] R.H. Olsson, R.B. Bogoslovov, C. Gordon, Event driven persistent sensing: Overcoming the energy and lifetime limitations in unattended wireless sensors, in: 2016 IEEE SENSORS, 2016, pp. 1–3, http://dx.doi.org/10.1109/ICSENS.2016.7808398.

[21] K. Zhu, Y. Wang, Event-Triggered Sensor Fault Estimation of Unreliable Networked Unmanned Surface Vehicle System With Correlated Noises, IEEE Trans. Veh. Technol. 71 (3) (2022-03) 2527–2537, http://dx.doi.org/10.1109/TVT.2022.3142147.

[22] Y. Fu, Y. Zhu, T. Hoang, K. Mechitov, B.F. Spencer, xImpact: Intelligent Wireless System for Cost-Effective Rapid Condition Assessment of Bridges under Impacts, Sensors 22 (15) (2022-01) 5701, http://dx.doi.org/10.3390/s22155701.

[23] E. Hidalgo-Fort, P. Blanco-Carmona, F. Muñoz-Chavero, A. Torralba, R. Castro-Triguero, Low-Cost, Low-Power Edge Computing System for Structural Health Monitoring in an IoT Framework, Sensors (Basel, Switzerland) 24 (15) (2024) 5078, http://dx.doi.org/10.3390/s24155078, arXiv:39124124.

[24] J. Tu, L. Yang, J. Cao, Distributed Machine Learning in Edge Computing: Challenges, Solutions and Future Directions, ACM Comput. Surv. (2024-12-13) http://dx.doi.org/10.1145/3708495.

[25] S. Cui, T. Hoang, K. Mechitov, Y. Fu, B.F. Spencer, Adaptive edge intelligence for rapid structural condition assessment using a wireless smart sensor network, Eng. Struct. 326 (2025-03-01) 119520, http://dx.doi.org/10.1016/j.engstruct.2024.119520.

[26] C.M. Bishop, H. Bishop, Deep Learning: Foundations and Concepts, Springer International Publishing, 2024, http://dx.doi.org/10.1007/978-3-031-45468-4.

[27] J. Lynch, H. Sohn, M. Wang, Sensor Technologies for Civil Infrastructures, Volume 1: Sensing Hardware and Data Collection Methods for Performance Assessment, in: Woodhead Publishing Series in Civil and Structural Engineering, Elsevier Science, 2014, URL https://books.google.com.sg/books?id=TfTCAgAAQBAJ.

[28] X. Ge, Q.-L. Han, Distributed Formation Control of Networked Multi-Agent Systems Using a Dynamic Event-Triggered Communication Mechanism, IEEE Trans. Ind. Electron. 64 (10) (2017-10) 8118–8127, http://dx.doi.org/10.1109/TIE.2017.2701778.

[29] L. Ding, Q.-L. Han, X. Ge, X.-M. Zhang, An Overview of Recent Advances in Event-Triggered Consensus of Multiagent Systems, IEEE Trans. Cybern. 48 (4) (2018-04) 1110–1123, http://dx.doi.org/10.1109/TCYB.2017.2771560.

[30] K. Ogata, Modern Control Engineering, Prentice Hall, 2010, arXiv:Wu5GpNAelzkC.

[31] T. Agrawal, Bayesian Optimization, in: Hyperparameter Optimization in Machine Learning, Apress, Berkeley, CA, 2021, pp. 81–108, http://dx.doi.org/10.1007/978-1-4842-6579-6_4.

[32] R. Ganguli, S. Adhikari, S. Chakraborty, M. Ganguli, Digital Twin: A Dynamic System and Computing Perspective, Taylor & Francis Group, 2023, URL https://books.google.com.sg/books?id=0o9RzwEACAAJ.

[33] S. Chen, W. Li, P. Pace, L. He, G. Fortino, Many-Objective Computation Offloading in Vehicular Edge Computing Using Bayesian and Incremental Learning Methods, IEEE Internet Things J. (2025) http://dx.doi.org/10.1109/JIOT.2025.3595091, 1–1.

[34] N.M. Newmark, A Method of Computation for Structural Dynamics, J. Eng. Mech. Div. 85 (3) (1959-07-01) 67–94, http://dx.doi.org/10.1061/JMCEA3.0000098.

[35] AnalogDevices, ADXL362 Datasheet and Product Info | Analog Devices, 2024-12-25, URL https://www.analog.com/en/products/adxl362.html.

[36] G. Han, B. Lin, Z. Xu, Electrocardiogram signal denoising based on empirical mode decomposition technique: An overview, J. Instrum. 12 (03) (2017-03) P03010, http://dx.doi.org/10.1088/1748-0221/12/03/P03010.

[37] P.I. Frazier, A Tutorial on Bayesian Optimization, 2018-07-08, http://dx.doi.org/10.48550/arXiv.1807.02811, arXiv:1807.02811.

[38] Espressif, ESP-DSP, 2025, URL https://components.espressif.com/components/espressif/esp-dsp.

[39] Espressif, ESP-DL, 2025, URL https://components.espressif.com/components/espressif/esp-dl.

[40] D. Lee, J.-S. Kim, S. Hong, Dual Core Based Microcontrollers Inference Design and Performance Analysis, IEEE Access 12 (2024) 120326–120336, http://dx.doi.org/10.1109/ACCESS.2024.3443406.

[41] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, S. Han, MCUNet: Tiny Deep Learning on IoT Devices, 2020, http://dx.doi.org/10.48550/arXiv.2007.10319, arXiv:2007.10319.