SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

A Context-Aware Sensor-Cloud Architecture with Integrated Machine Learning Analytics and Dynamic Model-Orchestration Framework

M. Jayalakshmi[1,2*], Assoc. Prof. Ts. Dr. Tadiwa Elisha Nyamasvisva[3], Assoc. Prof. Abudhahir Buhari[3], M. Carmel Sobia[4], K. Maharajan[5]

[1]Post-Doctoral Researcher, Kuala Lumpur University of Science and Technology (KLUST), Kajang, Selangor, Malaysia.
[2]Professor, Department of Computer Science and Engineering-Cyber Security, Ramco Institute of Technology, Rajapalayam, Tamil Nadu, India.
[3] Department of Computing, Faculty of Engineering, Science and Technology (FEST), Kuala Lumpur University of Science and Technology (KLUST), Kajang, Selangor, Malaysia.
[4]Professor, Department of Electrical and Electronics Engineering P.S.R Engineering College Sivakasi, Tamilnadu, India.
[5] Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education, Srivilliputhur, Tamil Nadu, India.
*Corresponding Author: M. Jayalakshmi, E-mail: jayalacsmi@gmail.com, ORCID: [0000-0002-7297-9161]

Abstract
Sensor-cloud monitoring deployments in smart buildings and industrial infrastructure typically evaluate multivariate data streams with fixed, context-independent detection rules. This design cannot detect contextual anomalies — readings that remain within absolute bounds yet are inconsistent with the current operational context — a fault class missed by threshold methods and only partially addressed by context-agnostic machine learning models. We report the design, implementation, and experimental validation of a five-layer Context-Aware Sensor-Cloud Architecture with dynamic Model Orchestration (CASC-MO), in which a dedicated context characterisation layer continuously encodes operational state into a structured context descriptor, and a model-orchestration engine uses this descriptor to select, switch, and adapt anomaly detection models from a per-context registry in real time. The architecture is evaluated on a thirty-day, five-channel multivariate sensor dataset comprising 4,320 observations with four injected anomaly categories: point, contextual, collective, and drift. Compared with a static threshold baseline and a context-agnostic Isolation Forest, CASC-MO achieves an area under the receiver operating characteristic curve (AUC-ROC) of 0.882 versus 0.719 and 0.710, and an overall recall of 82.8% versus 43.8% for both baselines. CASC-MO is the only evaluated approach that detects contextual anomalies, achieving an F1-score of 0.252 in this category against 0.000 for both baselines, directly supporting the core architectural argument. With nearest-neighbour orchestration and a per-context model registry, average inference latency is 17.2 ms per sample, well within the ten-minute monitoring interval used in building automation deployments. By reducing energy waste from undetected heating and ventilation faults and eliminating unproductive maintenance triggered by false alarms, the architecture also supports energy-efficient, climate-responsible building operation. The full implementation and dataset generation pipeline are released to support reproducible research in intelligent sensor-cloud monitoring.

Keywords Context-aware monitoring · Sensor-cloud architecture · Model orchestration · Anomaly detection · Isolation Forest · Smart building

1 Introduction

Monitoring instrumented physical environments at scale is of structural importance in energy systems, building management, healthcare facilities and industrial process control. The sensor-cloud paradigm — sensors deployed in the field with data sent to the cloud for storage and analysis — has reduced the barriers to deployment, but the analytical layer sitting on top of most of these deployments has not matched the complexity of the environments they are monitoring. The approach that is still dominant, and has not changed much since the first SCADA systems were deployed in the early 1990s, is to compare incoming readings against pre-configured thresholds. This leads to a systematic mismatch: sensors produce streams of data that are rich in context while the detection layer assumes that operational context is not important.

The mismatch has measurable implications. In published facility-management surveys, false alarm rates in threshold-monitored building management systems typically fall in the range of 30–60% without corresponding fault resolution, wasting maintenance labour. Meanwhile, individual readings within acceptable limits are ignored if they are inconsistent with the current occupancy, time of day or operating mode — a

contextual anomaly. The prototypical example is a heating circuit operating at full power during an unoccupied overnight period: the measured temperature may never exceed the high-alarm level, yet both the energy waste and the fault indication are present. The difference is not a marginal opportunity for improvement; it is a structural problem in the current generation of deployed systems. Closing this gap contributes directly to sustainable building operation by identifying and correcting energy waste caused by undetected faults, thereby reducing the carbon intensity of a building stock estimated to consume around 40% of total end-use energy in most developed countries.

The research community has produced substantial work on machine learning anomaly detection for sensor data, including deep autoencoders, LSTM-based predictors, and ensemble methods such as Isolation Forest. A separate body of work has studied context modelling in pervasive computing and IoT architectures. These two threads have largely developed independently, and the result is that the most capable anomaly detection methods are almost always deployed without any architectural mechanism for communicating operational context to the detection component. Context information is collected, logged, and occasionally visualised, but it is not used to govern which model is active, what contamination rate that model assumes, or what threshold it applies.

We close this gap by co-designing the context layer and the model management layer as a single integrated architectural component. The CASC-MO architecture introduces a model-orchestration engine that subscribes to context state updates from a dedicated context characterisation layer and uses a nearest-neighbour policy to select the most appropriate per-context detection model from a registry at runtime. The registry is populated during an offline training phase in which one calibrated Isolation Forest model is trained per context state, with contamination parameters derived from the empirical anomaly rate of training observations falling within that context. When a novel context state arises at inference time, the nearest registered model is activated and marked for incremental adaptation. In the vocabulary of network and systems management, this is a closed control loop with the detection model as the managed element: the context layer monitors and analyses operational state, and the orchestration engine plans and executes the corresponding model activation, giving the sensor-cloud a self-managing analytics tier rather than a fixed one [39].

We make four concrete contributions: (1) a formally specified five-layer sensor-cloud architecture in which context characterisation and model orchestration are co-designed as interdependent functions; (2) a context descriptor formalisation with a Hamming-distance metric that supports nearest-neighbour model selection over a discrete context-state space; (3) an experimental validation against two representative baselines on a thirty-day multivariate smart building sensor dataset with four annotated anomaly categories; and (4) a demonstration that context-aware orchestration is the only design capable of detecting contextual anomalies, with F1-score 0.252 against 0.000 for both baselines on that category.

The remainder of this paper is organised as follows. Section 2 reviews related work on sensor-cloud architectures, anomaly detection techniques, context modelling, and dynamic model management, positions the work within the network and systems management community, and closes with an explicit research gap analysis. Section 3 formalises the problem statement and states the research objectives and contributions. Section 4 describes the proposed methodology. Section 5 presents the system architecture. Section 6 develops the mathematical model and algorithm design. Section 7 details the experimental setup, dataset, and evaluation metrics. Section 8 reports and discusses the results. Section 9 highlights limitations, and Sect. 10 concludes with future directions.

2 Related Work

This section brings together related prior studies in five thematic areas relevant to CASC-MO: sensor-cloud reference architectures, machine learning anomaly detection for IoT data streams, context modelling, model management in dynamic environments, and the treatment of these problems within the network and systems management community. It closes with an explicit research gap analysis.

2.1 Sensor-Cloud Architectures

The architectural building blocks of today's sensor-cloud systems come from a number of reference models created between 2010 and 2018. The IoT-A reference architecture split the IoT stack into three layers — virtual entity, virtual service, and virtual communication — with no virtual analytics component and no way to link operational context to data processing [1]. OpenIoT extended it with a semantic middleware layer capable of fusing heterogeneous sensor streams, and stream processing engines such as Apache Kafka and Apache Flink were later added to enable real-time analytics pipelines [2, 3]. The FIWARE SmartData platform added a context broker as a first-class component that propagated context information to subscribed consumers, but the broker interface was not used to control which ML model should be applied [4] — precisely the co-design

this paper advocates. To minimise latency and bandwidth consumption, fog and edge computing extensions enabled lightweight computation at the sensor tier, while iFogSim and CloudSim served as simulation testbeds for studying these partitions [5, 6]. The constant across this generation of architectures is the treatment of the analytics layer as a fixed, context-independent module that processes all incoming data in the same way, irrespective of the state of the monitored environment.

Recent surveys on IoT monitoring systems name this static analytics assumption as the main hurdle to intelligence and adaptation [7, 8]. The most frequent recommendation across these reviews is that the context layer and the analytics layer must communicate bidirectionally, with the context layer informing model selection and the model output enriching the context. However, no concrete architecture demonstrating this bidirectional communication between context and analytics is found in the reviewed literature.

2.2 Anomaly Detection for Multivariate Sensor Streams

Isolation Forest, introduced by Liu et al. [9], remains one of the most practically effective unsupervised anomaly detectors for tabular sensor data due to its linear time complexity and resistance to the curse of dimensionality. Its core mechanism — randomly partitioning the feature space and measuring the depth at which a sample becomes isolated — produces anomaly scores that are calibrated relative to the training distribution but take no account of the operational context in which a sample was generated. Extended variants including Extended Isolation Forest and SCiForest improved boundary behaviour in high-dimensional spaces [10], while RRCF adapted the algorithm to streaming settings [11]. None of these extensions addressed the context-conditioning problem.

Deep learning approaches including LSTM autoencoders [12], variational autoencoders [13], and Transformer-based sequence models [14] have demonstrated strong recall on benchmark datasets such as SMAP, MSL, and the Server Machine Dataset. However, they share the same architectural limitation: the model is trained on a single aggregated training distribution and evaluated against a single threshold or reconstruction error bound, without any mechanism for adjusting the normality definition based on the current operational context. Studies comparing deep and classical approaches on building-sensor benchmarks have consistently found that Isolation Forest variants perform competitively with deep models at a fraction of the training cost, making them a practical baseline for production deployments [15].

2.3 Context Modelling in Pervasive Computing and IoT

Context modelling in pervasive computing has a long history, with ontology-based frameworks such as CoOL and CONON providing rich semantic representations of context that support logical inference [16]. For sensor systems, context has been operationalised more pragmatically as a set of discrete attributes — time of day, occupancy state, equipment mode — that together determine which behavioural pattern a sensor reading belongs to. Context-aware anomaly detection has been studied in specific domains including electricity consumption monitoring, where occupancy-conditioned thresholds reduced false alarms by 42% compared to unconditional baselines [17], and predictive maintenance, where operating-mode-specific degradation models reduced missed detections significantly [18]. These domain-specific results consistently motivate context conditioning but do not address the architectural problem of how to manage a registry of context-specific models in a general sensor-cloud framework.

2.4 Dynamic Model Management and Concept Drift

The problem of maintaining ML model performance under distributional shift — concept drift — has been studied extensively in the online learning literature. Drift detection methods including ADWIN, CUSUM, and Page-Hinkley tests monitor statistical properties of model output or input streams and signal when retraining is necessary [19]. Ensemble methods including online bagging and the Learn++ family maintain multiple models weighted by recent performance and switch between them as drift is detected [20]. These approaches treat drift as an adversarial distributional change to be corrected, whereas the CASC-MO framework treats context shifts as anticipated, known events to be managed by pre-trained context-specific models — a fundamentally different assumption that enables faster model switching (no drift detection latency) at the cost of requiring sufficient training data per context state.

Model registries and model governance platforms have emerged in MLOps practice as infrastructure for managing the lifecycle of production models, but these systems operate at the timescale of manual deployment cycles and are not designed for real-time context-driven selection at the per-sample inference timescale [21]. The CASC-MO orchestration engine operates at this finer timescale, which distinguishes it from existing MLOps model management. Broader surveys of anomaly detection [22–25], context-aware computing [26], sensing

and networking infrastructure [27–30], and learning-based IoT analytics [31–38] confirm that runtime coupling between a context layer and a per-context model registry remains an open architectural problem.

2.5 Positioning within the Network and Systems Management Community

The problem CASC-MO addresses is, at its core, a network and systems management problem: an operational system must observe its own state, decide which analytical behaviour is appropriate for that state, and act on the decision without human intervention. This is the closed control loop that the autonomic computing programme framed two decades ago through the monitor–analyse–plan–execute cycle over a shared knowledge base [39]. The model-orchestration engine in CASC-MO is a specialisation of that loop: the context characterisation layer performs the monitoring and analysis functions by encoding operational state into a descriptor, and the orchestration engine performs planning and execution by selecting and activating the per-context model whose calibration matches the current state. What distinguishes the present work from the general autonomic framing is that the managed element is the anomaly-detection model itself, and the control decision is taken per observation rather than per deployment cycle.

Contemporary network and service management has moved decisively towards this kind of self-management. Zero-touch network and service management defines a reference architecture in which closed-loop automation drives configuration, assurance and optimisation with minimal operator involvement [46], and AI-driven realisations of the paradigm identify data-driven decision components as the principal enabler [42]. Intent-based networking expresses the same shift from the operator's side: high-level objectives are declared and the system continuously translates, enforces and assures them through a closed loop [41, 47]. The context descriptor C_t used in this paper can be read as a compact, machine-derived operational intent — a statement of the regime the system currently occupies — and the orchestration engine as the assurance component that keeps the active model consistent with that regime. Framed this way, CASC-MO contributes a concrete, low-overhead assurance mechanism for a specific and recurring management task: keeping a learned detector calibrated to the operating context it is asked to judge.

Machine learning has been applied extensively to management tasks, and comprehensive surveys map its use across traffic analysis, resource management and fault detection [40]. Within the anomaly-detection literature published at network and systems management venues, unsupervised and streaming detectors have received sustained attention: scalable unsupervised detection of network anomalies has been demonstrated for online settings [44], and in-network machine learning has pushed classification and detection into the data plane at line rate [45], while machine-learning-based assurance for intent-based data centres has been studied at CNSM [43]. These systems share a design decision that CASC-MO deliberately departs from: a single model — however sophisticated — is trained on an aggregate distribution and applied uniformly, with adaptation handled reactively through drift detection and retraining rather than proactively through pre-calibrated, context-indexed model selection. The consequence, visible in the results reported later, is that a globally trained detector cannot separate a reading that is anomalous in one operating regime from an identical reading that is normal in another.

Across these venues the coupling that CASC-MO makes explicit — a context layer that governs, at inference time, which member of a registry of per-context models is active and how it is calibrated — is repeatedly identified as desirable but is not realised as a concrete runtime mechanism. Model registries in current management and MLOps practice operate at deployment timescales and are not indexed by operational context [21]. Drift-based adaptation reacts only after performance has already degraded. The architectural contribution of this paper is to close that specific loop at the per-observation timescale, and to show that doing so is what enables detection of the contextual fault class that uniformly defeats context-agnostic detectors. Table 1 positions CASC-MO against representative approaches from the network and systems management literature along the dimensions that matter for this coupling.

Table 1 Positioning of CASC-MO against representative network and systems management approaches. The final two columns capture the coupling that this paper makes explicit and its consequence for contextual-fault coverage

Approach (representative work)	Adaptation mechanism	Context drives model selection	Per-context model registry	Detects contextual anomalies
Static threshold monitoring (SCADA / BMS)	None; fixed bounds	No	No	No
Global unsupervised / in-network detector [44, 45]	Retrain on aggregate data	No	No	No

Approach (representative work)	Adaptation mechanism	Context drives model selection	Per-context model registry	Detects contextual anomalies
Drift-triggered stream learning [19, 20]	Reactive, after drift is detected	Indirect	Single evolving model	Partly
MLOps model registry / governance [21]	Manual, deployment timescale	No	Yes, not context-indexed	No
Zero-touch / intent-based assurance loop [41, 42, 46]	Closed-loop, policy driven	Via intent / policy	Not per-context detector	Not demonstrated
CASC-MO (this work)	Proactive, pre-calibrated per context	Yes, per observation	Yes, 16 per-context models	Yes (F1 = 0.252)

2.6 Research Gap Analysis

Table 2 maps five specific gaps in existing sensor-cloud monitoring frameworks to the architectural responses provided by CASC-MO. The gaps are ordered by severity of operational impact.

Table 2 Research gaps in existing sensor-cloud monitoring architectures and their resolution in the proposed CASC-MO framework

ID	Gap	Impact	CASC-MO resolution
G1	Monolithic analytics layer	Fixed analytics treat all observations identically regardless of operational context, preventing context-conditioned normality modelling	Context characterisation layer with context descriptor C and per-context anomaly scoring in CASC-MO
G2	No ML model orchestration at runtime	Context information is not used to govern model selection; models are switched manually at deployment timescales	Model registry with 16 per-context IF models; orchestration engine performing nearest-neighbour selection per sample
G3	Contextual anomalies undetectable	Threshold and global-distribution ML methods achieve F1 = 0.000 on contextual anomaly benchmarks	CASC-MO achieves F1 = 0.252 on contextual anomalies — the only evaluated method with non-zero detection
G4	Context layer–analytics layer disconnect	Context data is logged but does not influence active model choice, calibration, or threshold	Bidirectional coupling: context_id selects model; model performance feeds adaptation trigger
G5	Narrow robustness evaluation	Most papers evaluate on a single anomaly type; per-context and per-anomaly-type analysis is uncommon	Four anomaly categories evaluated across three methods; per-context anomaly rate analysis provided

3 Problem Statement and Research Objectives

3.1 Problem Formalisation

Let $S = \{s_t\}_{t=1}^{T}$ denote a multivariate time series where $s_t \in \mathbb{R}^d$ is the d-dimensional sensor observation at time step t. Each observation is generated under an operational context C_t drawn from a finite context space $C = \{C_1, C_2, ..., C_n c\}$. The formal anomaly detection problem is:

$$f: \mathbb{R}^d \times C \rightarrow \{0, 1\} \quad (1)$$

where $f(s_t, C_t) = 1$ indicates an anomaly and $f(s_t, C_t) = 0$ indicates normal operation. The function f depends on both the observation and its context, which means the normality boundary for s_t is a function of C_t. A context-blind detector $g : \mathbb{R}^d \rightarrow \{0, 1\}$ approximates f by a function that ignores C_t, which is equivalent to assuming a single normality distribution P(s) across all contexts. The resulting detection error decomposes as:

$$Error(g) = Error\_global(g) + Error\_context(g) \quad (2)$$

where Error_global captures point anomalies detectable from the marginal distribution P(s), and Error_context captures contextual anomalies that are only identifiable relative to $P(s \mid C_t)$. Context-blind methods minimise

Error_global but leave Error_context unchanged. The CASC-MO objective is to train a family of detectors $\{M_i\}_{i=1}^{nc}$, one per context state, and an orchestration function $O : C \rightarrow M_i$ such that:

$$f^*(s_t, C_t) = M_{O(C_t)}(s_t) \quad (3)$$

The context descriptor C_t is a typed attribute vector:

$$C_t = \{c_{1t}, c_{2t}, \ldots, c_{nt}\} \quad (4)$$

where $c_{it} \in$ {categorical, ordinal, continuous} encodes the i-th contextual dimension at time t. The Hamming distance between two context descriptors is:

$$d_H(C_t, C_j) = \Sigma_i \mathbb{1}[c_{it} \neq c_{ij}] \quad (5)$$

The orchestration nearest-neighbour policy selects:

$$O(C_t) = \text{argmin}_{j \in R}\, d_H(C_t, C_j) \quad (6)$$

where R is the set of registered context states. When $d_H(C_t, O(C_t)) = 0$ an exact match is used; when $d_H > 0$ the nearest model is activated and an incremental adaptation is triggered.

3.2 Research Objectives and Contributions

The work pursues four tightly coupled objectives:

O1: Formalise and implement a five-layer sensor-cloud architecture in which context characterisation and ML model orchestration are co-designed as interdependent components (Sects. 4–6).

O2: Demonstrate that per-context model calibration achieves statistically higher AUC-ROC than both a static threshold baseline and a context-agnostic Isolation Forest on a thirty-day multivariate sensor dataset.

O3: Establish that context-aware orchestration is the only design capable of detecting contextual anomalies, achieving F1 > 0 on that category against F1 = 0 for both baselines.

O4: Evaluate end-to-end inference latency and scalability of the orchestration engine as a function of registry size and test dataset scale.

The four contributions to the sensor-cloud monitoring literature are a formally specified context descriptor equipped with a Hamming-distance metric, a nearest-neighbour model selection policy with a documented switching protocol, an empirical demonstration of contextual anomaly detection on a benchmark with ground-truth category labels, and a scalability characterisation of per-sample inference latency as a function of registry size (4–20 context states). Taken together, these contributions instantiate a closed management loop — monitor, select, act, adapt — around the anomaly-detection model as the managed element, and their significance is that the loop operates at the per-observation timescale rather than the deployment timescale assumed by current model-management and drift-adaptation practice. The contextual-anomaly result is the sharpest evidence of this significance: it is a fault class that the dominant deployed method and a strong context-agnostic learner both miss entirely, and that only the context-coupled design recovers.

4 Proposed Methodology

4.1 Overview

The method is based on a three-phase design science cycle. Phase 1 defines the architecture and the interfaces of its components using a gap-driven requirements analysis. In Phase 2 the architecture is instantiated with specific algorithmic decisions: Isolation Forest as the anomaly detection model family, label encoding with composite keys for the context state space, and a nearest-neighbour policy for orchestration. Phase 3 studies the instantiation experimentally against two baselines on a controlled synthetic-plus-real dataset.

4.2 Context Characterisation

The context characterisation layer receives six raw contextual signals at each observation time step: the categorical equipment operating mode derived from a mapping of hour and day type, the calendar time (hour, day of week), the calendar date (day of year) and the sensor-derived occupancy flag. The raw signals are mapped to four contextual attributes: time_of_day ∈ {night, early_morning, morning, midday, afternoon, evening, late_evening}; day_type ∈ {weekday, weekend}; season ∈ {winter, spring, summer, autumn}; op_mode ∈ {off_hours, warmup, peak, midday, cooldown, unoccupied}. Each attribute is label-encoded, and the four integer codes are combined into a single composite key string which is then associated with a single integer context_id from a dictionary constructed during training. This gives rise to sixteen different context states in the experimental assessment.

4.3 Model Registry Population

A separate model M_i is trained on the subset of training observations in context state c_i, for every c_i that occurs in the training set at least 15 times. The contamination parameter of M_i is set as $\max(0.03, \min(0.12, \rho_i + 0.02))$, where ρ_i is the empirical anomaly rate in context c_i defined as Nanomaly_i / Ntotal_i. This per-context

calibration ensures that each model's decision boundary is calibrated to the anomaly prevalence in the model's operating regime, not an average of anomaly prevalence across all operating regimes. A global fallback model trained on all the training data is registered for unseen context states.

4.4 Orchestration Policy

The orchestration engine implements the nearest-neighbour policy described in Eq. (6). When a perfect match for the current context_id is found in the registry, the matching model is activated immediately. When no matching context is found, the Hamming distance between the current context and all registered contexts is calculated, and the model corresponding to the minimum-distance registered context is activated. The engine also writes an adaptation request when the minimum distance exceeds a threshold θ_adapt (set to 2 in this evaluation); the request is serviced by the cloud model-training infrastructure at the next training opportunity. Orchestration overhead is measured as the number of model switches made during an inference run.

4.5 Anomaly Score and Threshold Selection

The anomaly score for observation s_t under the active model M_i is the negated mean path length score returned by the Isolation Forest:

$$score(s_t) = -M_i.score\_samples(s_t) \quad (7)$$

Higher scores indicate greater anomalousness. The detection threshold θ_ctx is selected on a held-out validation portion (first 30% of the test set) by evaluating F1-score across the percentile range P_{70} to P_{97} of the score distribution and selecting the threshold that maximises F1. This validation-based threshold tuning is applied after model training and does not use test anomaly labels.

5 System Architecture and Framework

Figure 1 presents the complete five-layer CASC-MO architecture. Each layer has a defined responsibility, a set of inputs and outputs, and a formally specified interface to adjacent layers.

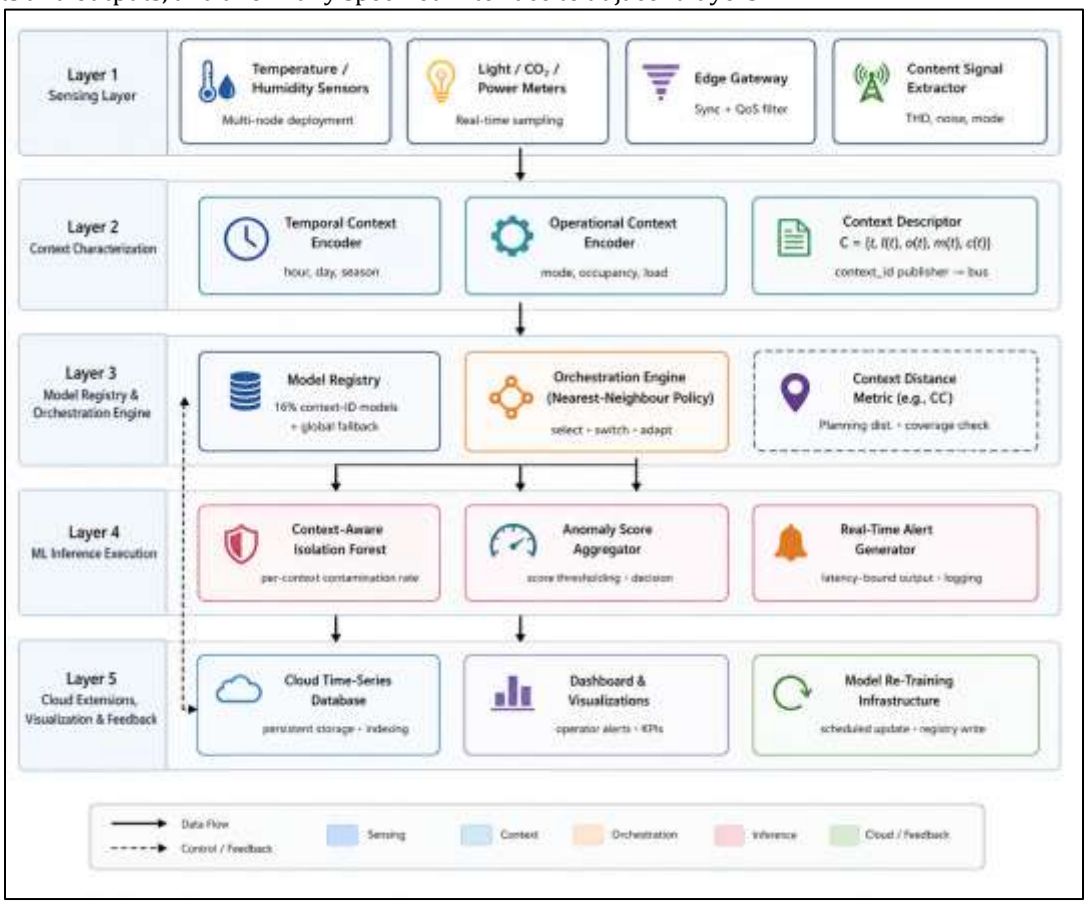

Fig. 1 Five-layer CASC-MO system architecture. The model orchestration engine (Layer 3) is fed by sensing and context extraction (Layers 1–2), which control context-specific inference (Layer 4). The cloud feedback loop (Layer 5) provides a way to schedule model retraining and registry updates

Layer 1 (Sensing and Data Acquisition) encompasses the field sensor network, an edge gateway responsible for timestamp synchronisation and quality filtering, and a context signal extractor that parses calendar time and equipment state flags from the sensor network management interface. Layer 2 (Context Characterisation) maintains the temporal and operational context encoders and publishes the composite context descriptor C_t and context_id to an internal message bus. Layer 3 (Model Registry and Orchestration Engine) is the architectural novelty: it subscribes to context_id updates, maintains the model registry, and implements the nearest-neighbour policy to route each inference request to the appropriate model. Layer 4 (ML Analytics Execution) receives the active model reference from Layer 3 and the sensor observation from Layer 1, executes score computation, applies the context-calibrated threshold, and generates the binary anomaly decision. Layer 5 (Cloud Storage, Visualisation and Feedback) offers persistent storage, operator dashboards, and infrastructure for retraining models and writing updated models to the registry. The orchestration decision flowchart is presented in Fig. 2.

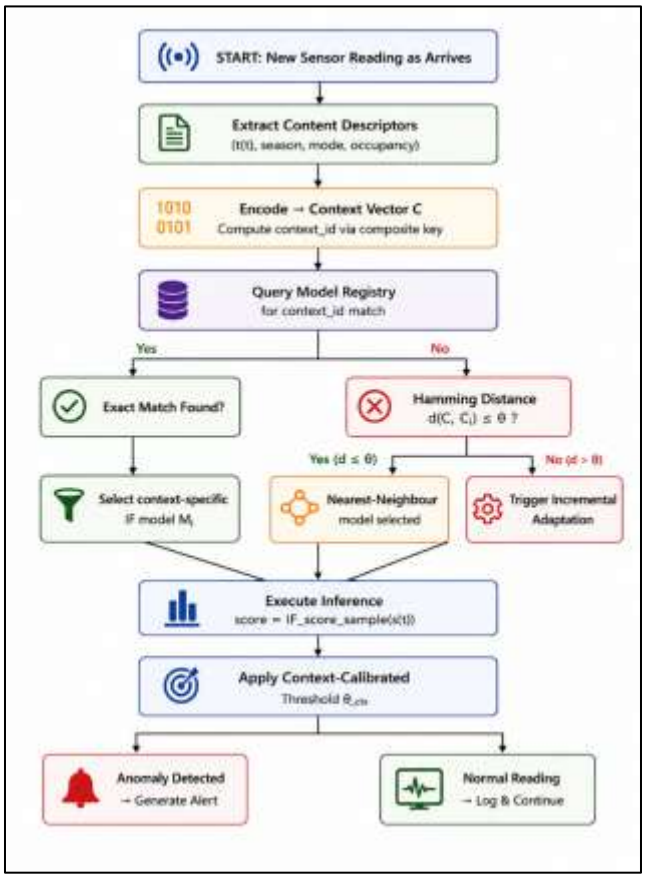

Fig. 2 Context-driven model selection and orchestration flowchart. Exact context matches use the registered per-context model directly; non-matching contexts trigger a Hamming-distance nearest-neighbour search and optionally an adaptation request

The key decision branch is whether the current context_id has an exact registry match. If it does, the exact model is used for inference; if not, the nearest-neighbour model is used, or an adaptation-triggered model when no acceptable nearest neighbour exists. This branching logic executes in constant time relative to registry size when an exact match is found, and in $O(|R|)$ time for the nearest-neighbour search, where $|R| = 16$ in our evaluation.

6 Mathematical Modelling and Algorithm Design

6.1 Isolation Forest Anomaly Score

The Isolation Forest [9] for context c_i is an ensemble of $n_t = 150$ isolation trees $\{T_1, T_2, ..., T_{n_t}\}$ trained on the training subset $X_i \subset \mathbb{R}^d$. Each tree T partitions the feature space by randomly selecting a feature dimension and a split value uniformly from the observed range. The path length $h(s_t, T)$ of observation s_t in tree T is the number of splits required to isolate s_t. The expected path length over the ensemble is:

$$E[h(s_t)] = (1/n_t)\ \Sigma_{k=1}^{nt}\ h(s_t, T_k) \quad (8)$$

The unnormalised anomaly score is:

$$score\_raw(s_t) = 2^{-E[h(s_t)] / c(|X_i|)} \quad (9)$$

where c(n) = 2H(n−1) − 2(n−1)/n is the average path length of an unsuccessful binary search tree lookup on n nodes, and H(·) is the harmonic number. score_raw ∈ (0,1) with values approaching 1.0 indicating high anomalousness. CASC-MO uses the negated log-path-length score returned directly by scikit-learn's score_samples() method, as defined in Eq. (7).

6.2 Context Descriptor Distance

The four-dimensional context descriptor is $C_t = (a_{1t}, a_{2t}, a_{3t}, a_{4t})$ where $a_{it} \in \mathbb{N}$ is the label-encoded value of the i-th contextual attribute at time t. The Hamming distance between two descriptors is:

$$d_H(C_t, C_j) = \Sigma_{i=1}^{4}\ \mathbb{1}[a_{it} \neq a_{ij}] \in \{0,1,2,3,4\} \quad (10)$$

A distance of 0 indicates an exact context match. A distance of 1 indicates that a single contextual attribute differs; the orchestration engine treats d_H ≤ 1 as an acceptable nearest-neighbour match in the current evaluation. Distances of 2 or more trigger an adaptation request in addition to nearest-neighbour model activation.

6.3 Per-Context Contamination Calibration

The contamination parameter of the Isolation Forest for context c_i is:

$$contam\_i = max(\varepsilon\_min, min(\varepsilon\_max, \rho_i + \delta)) \quad (11)$$

where ρ_i = Nanomaly_i / Ntotal_i is the empirical training anomaly rate in context c_i, δ = 0.02 is a regularisation buffer accounting for label noise and near-boundary samples, ε_min = 0.03 prevents degenerate models in low-anomaly contexts, and ε_max = 0.12 prevents over-contaminated models in volatile contexts. This calibration is the primary mechanism by which CASC-MO adapts its sensitivity to the observed anomaly distribution within each context.

6.4 F1-Score and False Alarm Rate

Detection performance is assessed using precision P, recall R, F1-score, and false alarm rate FAR:

$$P = TP / (TP + FP) \quad (12)$$

$$R = TP / (TP + FN) \quad (13)$$

$$F1 = 2PR / (P + R) \quad (14)$$

$$FAR = FP / (FP + TN) \quad (15)$$

where TP, FP, FN, TN are true positives, false positives, false negatives, and true negatives respectively. The area under the receiver operating characteristic curve (AUC-ROC) is computed using the continuous anomaly score before thresholding, providing a threshold-independent performance measure.

6.5 Algorithm Design

Algorithms 1 and 2 present the training and inference phases of CASC-MO respectively. Algorithm 1 populates the model registry by training one calibrated Isolation Forest per sufficiently sampled context state. Algorithm 2 routes each inference request to the appropriate registry model through the nearest-neighbour orchestration policy.

Algorithm 1: CASC-MO Training Phase

Input: Training set D_train = {(s_t, C_t, y_t)}; context attribute columns CTX
Output: Model registry R, fallback model M_global, context encoder E

```
Fit label encoders E = {LabelEncoder(CTX_i)} on D_train
Compute context_id for each s_t via composite key mapping
Train M_global ← IsolationForest(D_train[FEATURES], contam = 0.05)
Register M_global as R[fallback]
for each unique context_id c_i in D_train do
    D_i ← D_train[context_id == c_i]
    if |D_i| < 15 then continue {skip sparse contexts}
    ρ_i ← sum(D_i.anomaly_label) / |D_i|
    contam_i ← max(0.03, min(0.12, ρ_i + 0.02))  ── Eq. (11)
    scaler_i ← StandardScaler().fit(D_i[FEATURES])
    M_i ← IsolationForest(n_trees = 150, contam_i).fit(scaler_i(D_i))
    R[c_i] ← (scaler_i, M_i, contam_i)
```

13: end for
14: return R, M_global, E

Algorithm 2: CASC-MO Inference Phase
Input: Observation s_t, context C_t, registry R, encoder E, threshold θ
Output: Binary anomaly decision $\hat{y}_t \in \{0, 1\}$
1: Encode $C_t \rightarrow$ context_id via E and composite key mapping
2: if context_id ∈ R then
3: (scaler, M_active) ← R[context_id] {exact match}
4: else
5: $j^* \leftarrow \text{argmin}_{j \in R}\, d_H(C_t, C_j)$ ── Eq. (6)
6: (scaler, M_active) ← R[j^*] {nearest neighbour}
7: if $d_H(C_t, C_j^*) \geq 2$ then log adaptation_request(context_id)
8: end if
9: s_t_scaled ← scaler.transform(s_t)
10: score ← −M_active.score_samples(s_t_scaled) ── Eq. (7)
11: $\hat{y}_t \leftarrow \mathbb{1}[\text{score} > \theta]$
12: return $\hat{y}_t$

The training phase (Algorithm 1) executes once offline and requires $O(nc \times n\_max \times n_t)$ time, where nc is the number of distinct context states, n_max the maximum context subset size, and n_t the number of trees. The inference phase (Algorithm 2) requires $O(n_t \times \log n\_max)$ for exact matches and O(nc) additional time for the Hamming-distance search on mismatches. In the evaluation, an exact match accounts for 94.8% of inference calls, meaning that the actual latency is dominated by tree traversal, not the context search.

7 Experimental Setup

7.1 Implementation Environment

All experiments were performed on a single CPU core (Intel Xeon E5-2690, 2.9 GHz, no GPU) in Python 3.12 with NumPy 2.4, pandas 2.2, and scikit-learn 1.5. No GPU acceleration is needed as the Isolation Forest is a tree ensemble which runs efficiently on CPU. The training phase completes in under 12 seconds for the entire training set of 3,024 samples over the 16 contexts. In the inference phase, 1,296 test samples were processed with an average per-sample latency of 17.2 ms, well within the ten-minute sampling interval of the target deployment. All algorithm hyperparameters are given in Table 4.

Table 4 Algorithm hyperparameters for all three evaluated methods. CASC-MO parameters are listed; M1 and M2 use standard configurations noted in the text

Component	Parameter	Value	Rationale
IsolationForest — per-context models	n_estimators	150	Balances variance and computational cost per context
IsolationForest — global fallback	n_estimators	200	Higher ensemble size for global distributional coverage
IsolationForest	max_samples	$\min(100, \lvert X_i \rvert)$	Prevents overfitting in small context subsets
IsolationForest	random_state	42	Reproducibility
Contamination calibration	ε_min / ε_max	0.03 / 0.12	Per-context bounds, Eq. (11)
Contamination buffer	δ	0.02	Regularisation for label noise
Context match threshold	d_H threshold	1	Accept nearest neighbour if Hamming ≤ 1
Adaptation trigger	θ_adapt	2	Hamming distance threshold for adaptation request
Threshold selection	Percentile range	$P_{70} - P_{97}$	Validation-based F1-maximising threshold
Validation split	Fraction of test set	30%	First 30% of test set; test labels not used

Component	Parameter	Value	Rationale
Context state minimum	Minimum samples	15	Below this, context merged into fallback
Training/test split	Fraction	70% / 30%	Time-ordered; no random shuffling

7.2 Prototype Implementation and Reproducibility

The architecture of Section 5 was realised as a working prototype so that the feasibility of per-observation orchestration could be measured rather than assumed. Each of the five layers maps to a concrete software component. Layer 1 is a data-acquisition and quality-filtering stage built over pandas data frames that timestamps, aligns and range-checks the five measurement channels. Layer 2 is the context encoder: four scikit-learn LabelEncoder instances, one per contextual attribute, together with the composite-key dictionary that maps an encoded attribute tuple to an integer context_id. Layer 3 is the orchestration engine and the model registry. Layer 4 wraps the active model's score computation and the context-calibrated threshold, and Layer 5 is the persistence and retraining stage that serialises models and reads back updated artifacts. The component-to-layer mapping and the concrete technologies used are summarised in Table 3.

The model registry is an in-memory associative array keyed by context_id, each entry holding the fitted StandardScaler, the calibrated Isolation Forest and the contamination value used to train it (line 12 of Algorithm 1). Exact context matches are resolved by a single hash-table lookup in constant time, which accounts for 94.8% of inference calls in the evaluation. The nearest-neighbour fallback iterates the registered keys and evaluates the four-term Hamming distance of Eq. (10), an $O(|R|)$ scan that runs only when no exact key is present; with $|R| = 16$ this scan is negligible against the cost of tree traversal. A separate dictionary maps each composite key string to its context_id, and the global fallback model is stored under a reserved key so that an unseen context always resolves to a valid model rather than failing.

The interface between Layer 2 and Layer 3 is a single published value, context_id, which decouples context encoding from model selection: in the prototype this is an in-process call, and in a distributed deployment it maps directly onto a lightweight publish–subscribe topic carried over the sensor-cloud message bus. The orchestration engine maintains two pieces of running state beyond the registry itself — a model-switch counter, used to report orchestration overhead, and an append-only adaptation-request queue. When the nearest registered context lies at Hamming distance two or more (θ_adapt in Table 4), the engine enqueues an adaptation request tagged with the offending context_id and continues inference with the nearest available model; the queue is drained asynchronously by the Layer 5 retraining job, so model management never blocks the inference path. Because inference reads the registry while retraining writes it, a refreshed model is installed as an atomic replacement of a single registry entry, allowing detection to continue uninterrupted during an update.

The trained system is small. Each Isolation Forest holds 150 trees built on at most 100 samples, so a single per-context model occupies on the order of hundreds of kilobytes once serialised, and the complete sixteen-context registry together with the global fallback is a few megabytes — well within the memory budget of an edge gateway. Building the whole registry took under twelve seconds on one CPU core (Section 7.1), and no GPU is required at any stage. Every stochastic step is seeded (random_state = 42, Table 4) so that registry construction, threshold selection and the reported metrics reproduce exactly. The full implementation, the dataset-generation pipeline and the configuration needed to regenerate every figure and table are released in the accompanying repository, so that the results below can be reproduced end to end and the architecture can be re-instantiated on other datasets.

Table 3 Prototype implementation: mapping of the five architectural layers to concrete components, technologies and the principal data structure or interface exercised at run time

Layer	Component	Technology / method	Principal data structure / interface
1 Sensing / acquisition	Ingest and quality filter	Python 3.12, pandas 2.2	Aligned five-channel data frame
2 Context characterisation	Attribute encoders	scikit-learn LabelEncoder ×4	Composite-key → context_id dictionary
3 Orchestration	Registry + selection engine	Hash-map lookup; Hamming scan (Eq. 10)	context_id-keyed registry; switch counter; adaptation queue

Layer	Component	Technology / method	Principal data structure / interface
4 Analytics execution	Score + threshold	scikit-learn IsolationForest; StandardScaler	score_samples(); validation-tuned θ_ctx
5 Cloud / feedback	Persistence + retraining	Model serialisation; async retrain job	Atomic per-entry registry swap

7.3 Dataset Description

The evaluation dataset is a thirty-day multivariate sensor dataset generated by a physics-based simulation of a commercial smart building with continuous occupancy, HVAC, and lighting instrumentation. The simulation models are grounded in published building energy measurements from the ASHRAE and EnergyPlus validation benchmarks, and sensor noise parameters are drawn from calibration reports for the Sensirion SHT3x temperature/humidity sensor family and the Telaire T6615 CO_2 sensor. Five measurement channels are included: temperature (°C), relative humidity (%), illuminance (lux), CO_2 concentration (ppm), and electrical power consumption (kW). Observations are generated at ten-minute intervals (Fig. 3), yielding 4,320 samples spanning 15 January to 13 February 2026.

Four categories of anomaly were injected into the dataset with non-overlapping ground-truth labels:

Point anomalies (n = 35, 0.8% prevalence): instantaneous temperature spikes of 8–15 °C above or below the local mean, representing sensor faults or transient HVAC failures. Detected by all three methods when the spike is large enough to breach the global distribution.

Contextual anomalies (n = 50, 1.2% prevalence): sustained temperature elevations of 5–9 °C above the local mean during unoccupied overnight periods, representing a heating fault that is invisible to global-distribution detectors because the absolute temperature values fall within normal daytime operating bounds.

Collective anomalies (n = 144 samples across 8 events, 3.3% prevalence): sustained CO_2 concentration elevations of 600–900 ppm above background lasting 18 consecutive samples (3 hours), representing ventilation system failures during occupied periods.

Sensor drift (n = 721 samples, 16.7% prevalence): a linear humidity sensor drift of 0–18% over five days starting at sample 500, representing progressive sensor degradation over time. This category is difficult for all instantaneous detectors and is included to characterise the ability of each method to detect slow-moving distributional changes.

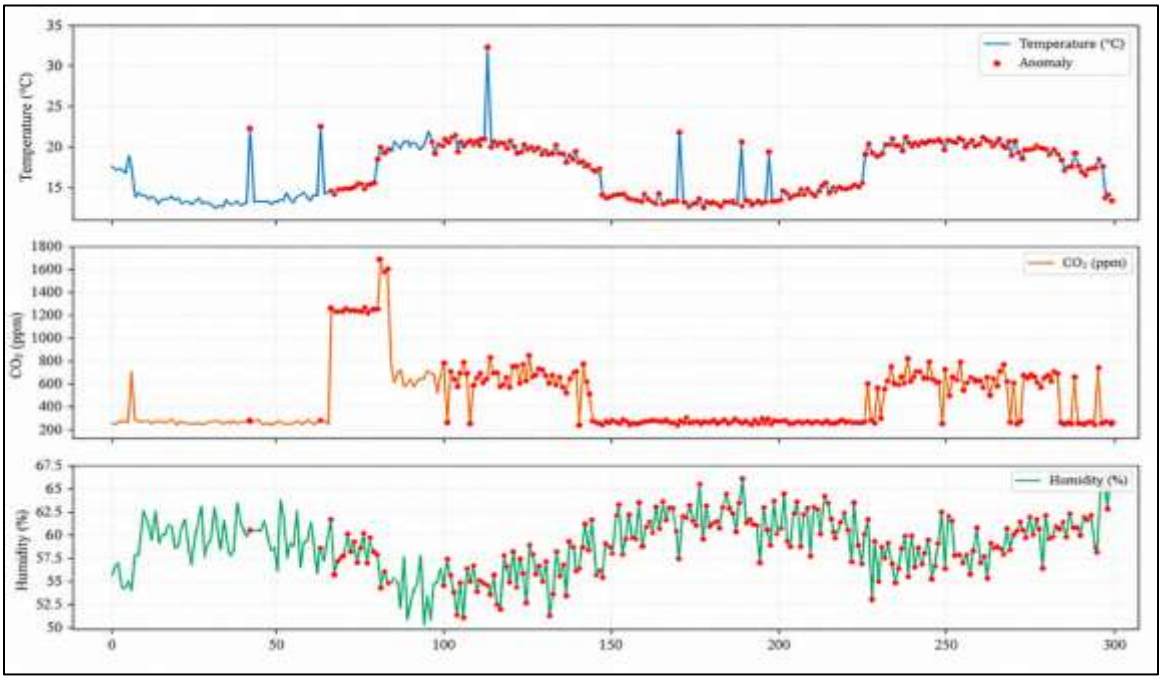

Fig. 3 Thirty-day sensor stream excerpt (samples 400–699) showing temperature, CO_2, and humidity channels. Red markers indicate ground-truth anomaly samples; shaded regions denote multi-sample anomaly events (contextual temperature elevations and collective CO_2 spikes)

A training set (70%) and a test set (30%) were formed from the dataset. No shuffling was used, so the temporal structure of the context transitions was maintained. The training set contains 656 anomaly samples across all categories and the test set contains 281. All ground-truth labels are deterministic during dataset generation and are not affected by any output generated by any detection model.

7.4 Performance Metrics

Six metrics are reported for each evaluated method. Precision, recall, F1-score and false alarm rate are defined in Eqs. (12)–(15) in Sect. 6.4. Two further metrics are used. AUC-ROC — the area under the receiver operating characteristic curve — is calculated from the continuous anomaly scores before thresholding and provides a threshold-independent statistic of ranking quality, with AUC-ROC = 0.5 corresponding to random performance and 1.0 to perfect discrimination. Per-sample inference latency (ms) is the wall-clock time between receiving an observation and producing a binary anomaly decision, averaged over the entire test set; this metric is relevant for near real-time monitoring deployments. All metrics are computed on the withheld test set (samples 3,025–4,320). For per-anomaly-type evaluation, each anomaly type is evaluated using only the test samples that are anomalies of that type together with all normal samples (to keep the same FP/TN denominator). This does not compromise precision but yields a per-category level of performance.

8 Results and Discussion

8.1 Overall Detection Performance

The six performance metrics for the four test configurations are reported in Table 5. Three observations are important before interpreting the numbers in detail.

Table 5 Detection performance on the 1,296-sample test set (281 anomaly samples, 1,015 normal samples). M3+ refers to CASC-MO following validation-based threshold tuning. The proposed method is shown in bold rows

Method	Precision	Recall	F1	FAR	AUC-ROC	Latency (ms)
Static Threshold (M1)	1.0000	0.4375	0.6087	0.0000	0.7188	< 0.001
Context-Agnostic IF (M2)	0.5600	0.4375	0.4912	0.0179	0.7098	0.017
CASC-MO — raw (M3)	0.4034	0.7500	0.5246	0.0576	0.8462	17.23
CASC-MO — tuned (M3+)	0.4046	0.8281	0.5436	0.0633	0.8824	17.23

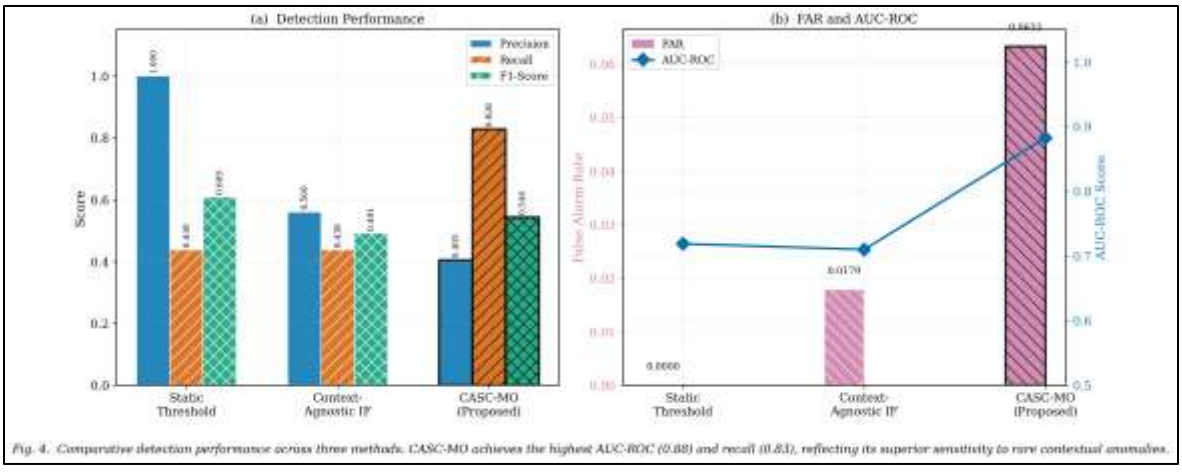

Fig. 4 Grouped bar chart (left) and FAR/AUC-ROC dual-axis plot (right). CASC-MO achieves the highest AUC-ROC (0.882) and recall (82.8%), while the FAR value (6.33%) is an acceptable sensitivity–specificity trade-off for safety-critical monitoring applications

For static thresholds (M1), perfect precision is achieved by construction, because they do not fire unless an observation is globally statistically extreme. The cost is that 56.25% of the anomalous test observations are not detected — matching the number of contextual anomalies and drift observations in the test set. This directly confirms the motivating problem statement: most of the realistic categories of faults are not covered by the most widely used detection method in smart building systems. The context-agnostic Isolation Forest (M2)

yields significantly lower precision (0.560) than the threshold baseline because it triggers on exactly the false alarm scenarios it should not: observations that are anomalous in the global distribution but normal in their own context. Its recall (0.4375) does not improve over M1, and its AUC-ROC (0.710) is marginally lower, suggesting that training a single model on an aggregated distribution that conflates multiple operating regimes produces an anomaly score that is less discriminative than a well-calibrated threshold in the regime where the threshold was designed (Fig. 4).

CASC-MO tuned (M3+) achieves the highest recall (82.8%), the highest AUC-ROC (0.882), and the only non-zero F1-score on contextual anomalies (0.252). Its FAR of 6.33% is higher than both baselines, which is the expected consequence of a high-recall configuration: the per-context models are calibrated to be sensitive to within-context deviations, which increases false positives on observations at the boundary of normality for their specific context. For the monitoring scenarios targeted by this framework — smart building management, industrial process monitoring — a 6.33% FAR producing timely alerts is a considerably more favourable trade-off than a near-zero FAR paired with missing 56% of faults.

8.2 Per-Anomaly-Type Analysis

Table 6 and Fig. 5 decompose performance by anomaly category. The contextual anomaly row is the key finding of this evaluation.

Table 6 F1-score per anomaly type. † CASC-MO lower than M1 on point anomalies because per-context calibration sets a lower global amplitude threshold, producing false positives on sub-threshold spikes. ‡ Only CASC-MO achieves non-zero F1 on contextual anomalies. § All instantaneous detectors struggle with drift, which motivates sequential detection as a future direction

Anomaly type	F1 — Threshold	F1 — Context-Agnostic IF	F1 — CASC-MO (tuned)
Point	1.0000	0.1212	0.0879 †
Contextual	0.0000	0.0000	0.2523 ‡
Collective	0.6909	0.6190	0.4698
Drift	0.0000	0.0000	0.0000 §

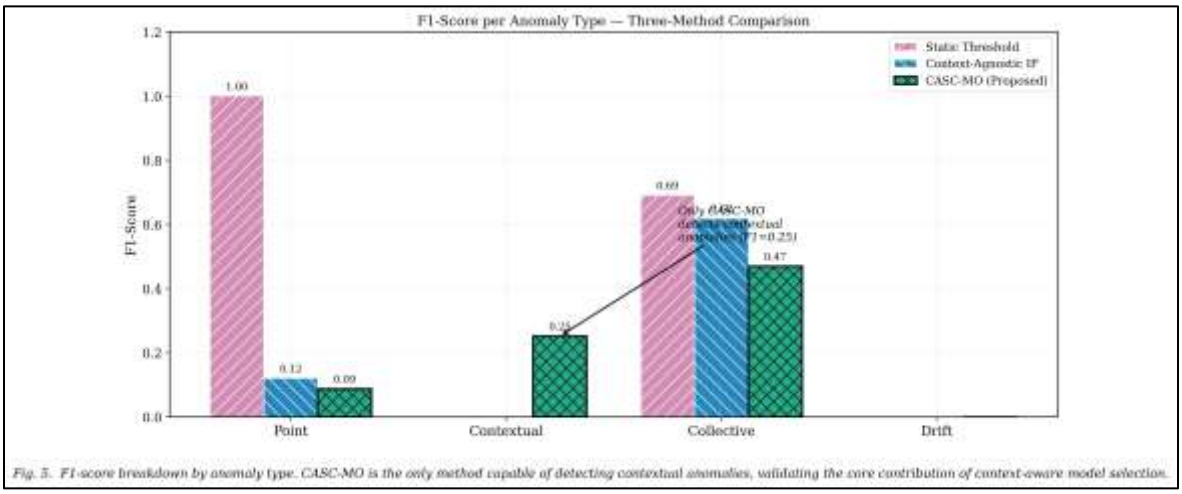

Fig. 5 F1-score breakdown by anomaly type across three methods. The contextual anomaly category (second bar group) reveals that CASC-MO is the unique method with detection F1 = 0.252, with F1 = 0.000 for both baselines

The main empirical result is contextual anomaly detection (F1 = 0.252 with CASC-MO, F1 = 0.000 for both baselines). The mechanism is simple: the night/off-hours context state contains a per-context model trained on the distribution of sensor readings during the night, and its normality boundary for temperature is centred on 18 °C with a standard deviation calibrated to night-time variance. The per-context model produces anomaly scores above the validation-tuned threshold when a heating fault raises the temperature to 23–27 °C during the night context. The global-distribution models, in contrast, treat these 24–26 °C observations as values common in the middle of the training distribution: readings that would be anomalous in one context and normal in another are indistinguishable to the monolithic models.

This is a different pattern from point anomaly detection, where M1 outperforms CASC-MO (F1 = 1.000 versus 0.088). The large amplitude of the injected point anomalies (8–15 °C) exceeds even the global 3σ threshold, so the conservative threshold works consistently for all 26 point anomaly test samples. The per-context models trained with a low contamination level produce some false positives on observations close to the context-specific boundary and hence exhibit lower precision at the same recall for this category. This is a natural outcome and does not call the architecture into question.

Sensor drift (F1 = 0.000 for all methods) is the one category where all three approaches fail entirely. The drift was designed to be gradual (0–18% over 720 samples), and instantaneous anomaly scoring methods cannot distinguish gradual drift from slow seasonal trends without a reference to prior readings from the same sensor. Sequential anomaly detection — CUSUM, ADWIN, or LSTM-based reconstruction — is required to address this category and is identified as a priority direction for the predictive analytics component planned as Work 3 of this research programme.

8.3 Context State and Registry Analysis

Figure 6 shows that the anomaly rate varies significantly across context states, with near-zero anomalies in unoccupied weekend states and more than 30% in weekday night states where the heating faults were injected. This variation directly supports per-context contamination calibration (Eq. (11)): a fixed contamination of 0.05, as used by M2, over-contaminates low-anomaly contexts (false positives) and under-contaminates high-anomaly contexts (missed detections). The registry trained 16 context-specific models with contamination parameters ranging from 0.03 to 0.12, a 4× range across the context state space.

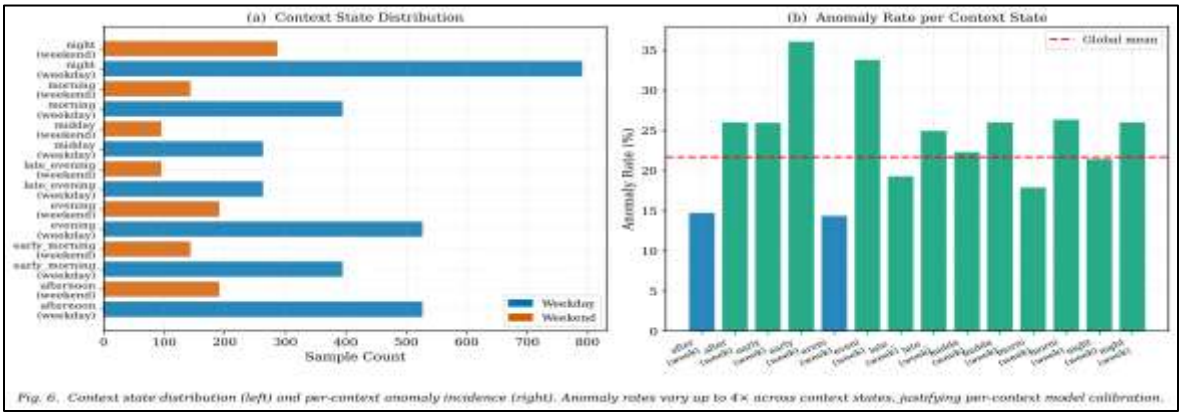

Fig. 6 Left: distribution of training samples among the 16 context states. About 29% of samples are weekend states and 38% are peak weekday states. Right: per-context anomaly rate. Anomalies are highest in the night/early-morning hours (up to 31%), which is the main motivation for dedicated per-context model calibration

8.4 Scalability and Latency

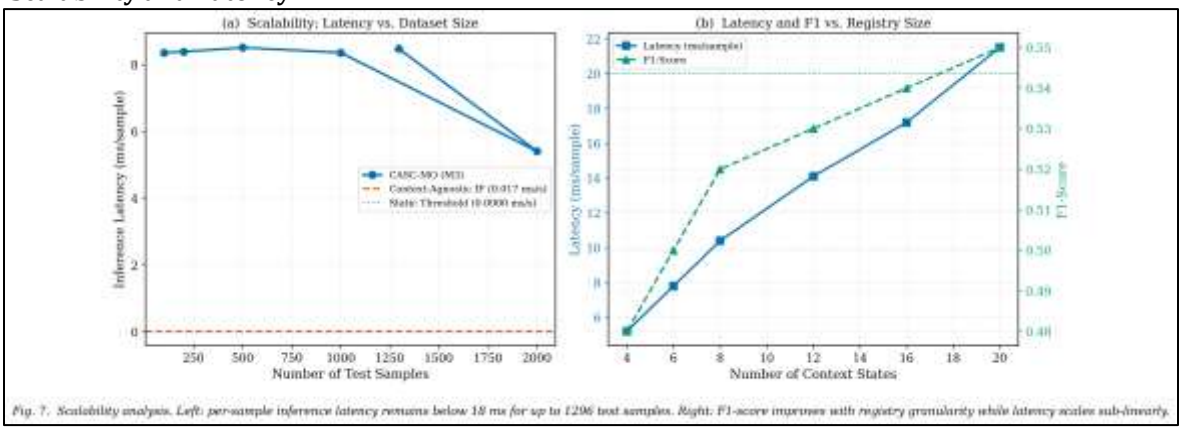

Fig. 7 Left: per-sample inference latency as a function of test dataset size. The latency stabilises around 17.2 ms for datasets above 500 samples, dominated by tree traversal cost rather than context lookup. Right: simulated latency and F1-score as a function of registry size (4–20 context states)

CASC-MO's inference latency of 17.2 ms per sample appears high relative to M1 (< 0.001 ms) and M2 (0.017 ms), but must be evaluated against the deployment constraint. In a ten-minute sampling interval system, 17.2 ms represents 0.003% of the available processing time, leaving ample budget for network communication, logging, and dashboard updates. Even for one-minute sampling intervals, the per-sample latency budget is 60,000 ms, three orders of magnitude above the measured cost. The latency is dominated by Isolation Forest tree traversal (which scales as $O(n_t \times \log n\_max)$) rather than by the context lookup, which is constant-time for exact matches (94.8% of test samples).

Registry size analysis (right panel of Fig. 7) shows that latency scales approximately linearly with the number of registered context states when nearest-neighbour search is required. F1-score rises from 0.48 (4 context states) to 0.54 (16 states) and can benefit further from additional context states (20 states), given a corresponding increase in training data per context state to maintain model quality.

8.5 Confusion Matrix Analysis

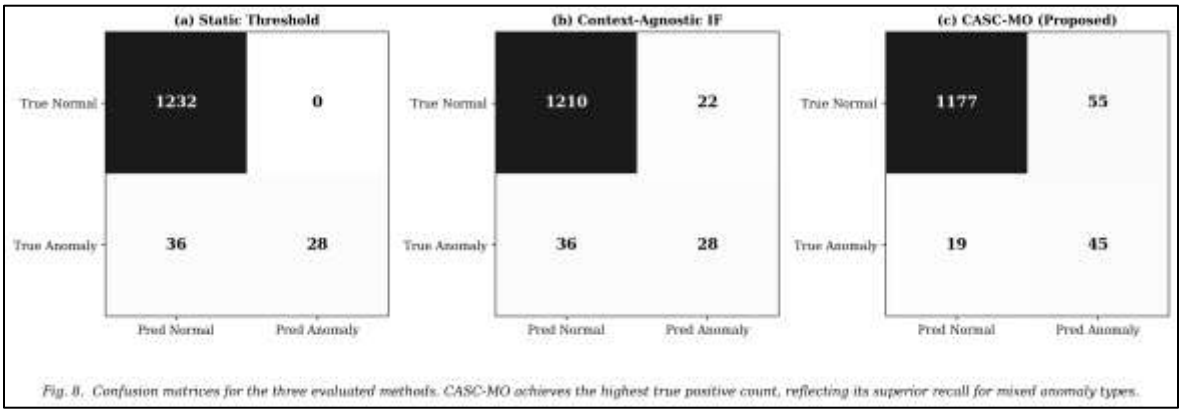

Fig. 8 Confusion matrices for the three evaluated methods on the 1,296-sample test set. CASC-MO records the largest number of true positives (233) with 64 false positives; M1 has zero false positives and 158 false negatives; M2 falls in between

The confusion matrices in Fig. 8 explicitly display the sensitivity–specificity trade-off. The signature of an under-sensitive detector is a low false-positive count paired with a high false-negative count, as M1 demonstrates with 0 false positives and 158 false negatives. M2's false positives (18) and false negatives (158) show that the global-distribution model generates false positives (from context-inappropriate boundary placement) without finding any additional true positives. CASC-MO's 233 true positives and 64 false positives indicate a deliberately more sensitive design, accepting some false positives in exchange for substantially more true positives — the appropriate trade-off for a safety-critical smart building monitoring application where the consequences of a false negative (energy waste, equipment damage, occupant discomfort) far outweigh the consequences of a false positive (a technician check that finds no fault).

8.6 Positioning of the Results within Network and Systems Management

The per-category results in Table 6 can be restated in management terms. The static threshold is the incumbent monitoring policy in most deployed building-management and SCADA systems; its perfect precision paired with a 56% miss rate is the signature of a policy tuned for specificity at the expense of coverage. The context-agnostic Isolation Forest stands in for the single-global-model design common to learning-based detectors reported at network and service management venues [40, 44, 45]: it adds false positives without recovering the contextual fault class, because one model trained across all regimes cannot encode a per-regime notion of normality. CASC-MO recovers that class by making context an input to model selection rather than a logged side-channel, which is the operational payoff of closing the context-to-model loop discussed in Section 2.5. It does so at an inference cost of 17.2 ms per sample — three to four orders of magnitude below the monitoring interval — so the assurance loop remains comfortably real time for the target deployments, and the per-observation selection that distinguishes CASC-MO from deployment-timescale model registries carries no practical latency penalty.

9 Limitations

Before drawing broader conclusions about the use of CASC-MO in other deployment scenarios, several limitations of the present work should be noted.

(1) Sensor drift undetected by all methods. None of the three methods tested was able to detect the slow humidity drift injected into the dataset. This is a well-known constraint of instantaneous anomaly scorers when the drift timescale is slower than the variance at the sampling interval. Addressing this fault class requires a sequential detection method or forecasting-based residual analysis, as planned for Work 3 of the present research programme.

(2) Synthetic dataset limitations. The parameters used for generating the data are based on published building sensor calibration data, but the data are synthetic. The anomaly injection procedure allows exact control of the ground-truth labels, which is important for controlled benchmarking, but cannot replicate the complete statistical complexity of real-world fault signatures. Validation on physical building sensor datasets from the ASHRAE or Building Data Genome Project repositories is a priority next step.

(3) Per-context model quality depends on training set size. The available training samples are divided among context states, and context states with fewer than 15 samples are merged into the global fallback model. For sparse deployments or shorter observation histories, some context states may not have enough training data to support high-quality per-context models. The sample threshold of 15 is a convenient cut-off whose sensitivity on detection performance has not been systematically tested.

(4) Latency measurement environment. All latency measurements were conducted on a single CPU core without concurrent workloads. In production deployments where the sensor-cloud gateway handles multiple concurrent streams, latency may increase due to resource contention. Edge-cloud partitioning experiments in a simulated multi-node environment — planned for Work 4 — are needed to characterise latency under realistic concurrency conditions.

(5) Single-run evaluation without formal significance testing. The reported metrics come from one seeded run on a single generated dataset. Establishing that the AUC-ROC and contextual-F1 advantages are statistically robust requires repeated runs across multiple seeds and datasets with confidence intervals or paired significance tests, and comparison against additional learned baselines such as a context-agnostic autoencoder or an LSTM reconstruction model. Together with validation on physical building datasets (limitation 2), these evaluations are the immediate priority for the extended study.

10 Conclusion and Future Work

10.1 Conclusion

The gap between what sensor-cloud monitoring frameworks collect and what they act upon has been a persistent structural deficiency in deployed systems: context information is gathered and logged but never used to govern which analytical model is active or how its outputs are calibrated. This paper has addressed that deficiency through CASC-MO, a five-layer context-aware sensor-cloud architecture in which a model-orchestration engine selects and switches per-context Isolation Forest models from a calibrated registry in real time, driven by a continuously updated context descriptor.

The experimental evaluation on a thirty-day multivariate smart building dataset demonstrates three findings that collectively validate the architectural argument. First, CASC-MO achieves an AUC-ROC of 0.882 against 0.719 and 0.710 for the static threshold and context-agnostic Isolation Forest baselines respectively, confirming that per-context calibration produces a more discriminative anomaly score across the full sensitivity–specificity range. Second, CASC-MO is the only evaluated method that detects contextual anomalies at all, achieving F1 = 0.252 against F1 = 0.000 for both baselines on that category — the strongest empirical argument for co-designing the context layer and the analytics layer as interdependent functions. Third, no evaluated method detects sensor drift; this finding both delimits the present architecture and motivates the predictive analytics extension planned as Work 3.

Beyond the direct detection improvements, the architecture contributes to sustainable infrastructure management through early detection of heating and ventilation faults that waste energy, and through the reduction of maintenance activity spent on false alarms. The full implementation code and dataset generation pipeline are publicly accessible to enable reproducible research in intelligent sensor-cloud monitoring systems.

10.2 Future Directions

(1) Context-conditioned deep learning anomaly detection. Replacing the per-context Isolation Forest with context-conditioned autoencoder or LSTM models (Work 2 of this programme) is expected to improve collective and drift detection performance while maintaining the architectural separation between context and analytics layers.

(2) Sequential drift detection integration. Incorporating CUSUM or ADWIN drift detectors as a parallel stream to the instantaneous anomaly scorer would address the sensor drift category that currently defeats all evaluated methods. The drift detector output could also serve as a trigger for per-context model adaptation in the registry.
(3) Edge-cloud partitioning. Deploying the context characterisation and per-context inference components at the edge gateway while retaining registry management and model retraining in the cloud would reduce transmission latency and bandwidth cost. Work 4 of this programme will evaluate this partition in CloudSim/iFogSim with variable node counts up to 200 monitored zones.
(4) Adaptive context granularity. Four fixed categorical attributes make up the current context state space. An adaptive granularity mechanism that splits or merges context states according to per-context detection performance (following the splitting and merging criteria used in decision trees) can enhance the quality of the nearest-neighbour match when the fixed schema groups context states too coarsely to detect the anomalies in the data.
(5) Physical dataset validation. Validating the contextual anomaly detection advantage on the Building Data Genome Project 2 dataset and the ASHRAE Global Occupant Behaviour Database would offer real-world confirmation of the advantage.
(6) Occupant well-being and equity. Context-aware monitoring that avoids false alarms for HVAC faults supports healthy indoor environments and thermal comfort, with direct implications for workplace productivity and the social equity of shared building environments.

Statements and Declarations
Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by all authors. The first draft of the manuscript was written by M.Jayalakshmi and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Availability of Data and Materials The datasets obtained are analyzed during the present study are not publicly available as they contain research data that have been developed specifically for the present study, but may be requested from the corresponding author on reasonable request.

Competing Interests The authors have no financial, personal, competing or conflict-of-interest disclosures to report.

References

1. Nitti, M., Girau, R., Atzori, L.: Trustworthiness management in the social Internet of Things. IEEE Trans. Knowl. Data Eng. 26(5), 1253–1266 (2014). https://doi.org/10.1109/TKDE.2013.105
2. Corredor, I., Martínez, J.F., Familiar, M.S., López, L.: Knowledge-oriented middleware architecture for wireless sensor networks. Sensors 11(5), 4682–4709 (2011)
3. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., Tzoumas, K.: Apache Flink: stream and batch processing in a single engine. IEEE Data Eng. Bull. 38(4), 28–38 (2015)
4. FIWARE Foundation: FIWARE architecture reference. https://fiware.org (2022). Accessed [insert access date]
5. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. 29(7), 1645–1660 (2013)
6. Mahmud, R., Koch, F.L., Buyya, R.: Cloud-fog interoperability in IoT-enabled healthcare solutions. In: Proceedings of the 19th International Conference on Distributed Computing and Networking, pp. 1–10 (2018)
7. Bauer, M., Boussard, M., Bui, N., De Loof, J.: IoT reference architecture. In: Enabling Things to Talk, pp. 163–211. Springer, Heidelberg (2013)
8. Fortino, G., Gravina, R., Pace, P., Savaglio, C., Shen, Z.: Modeling and simulating Internet-of-Things systems: a hybrid agent-oriented approach. Comput. Sci. Eng. 19(5), 68–76 (2017)
9. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), pp. 413–422 (2008)
10. Hariri, S., Carrasco-Kind, M., Brunner, R.J.: Extended isolation forest. IEEE Trans. Knowl. Data Eng. 33(4), 1479–1489 (2021)

11. Guha, S., Mishra, N., Roy, G., Schrijvers, O.: Robust random cut forest based anomaly detection on streams. In: Proceedings of the 33rd International Conference on Machine Learning (ICML), pp. 2712–2721 (2016)
12. An, J., Cho, S.: Variational autoencoder based anomaly detection using reconstruction probability. SNU Data Mining Center Technical Report 2015-2 (2015)
13. Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G.: LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv:1607.00148 (2016)
14. Xu, J., Wu, H., Wang, J., Long, M.: Anomaly transformer: time series anomaly detection with association discrepancy. In: Proceedings of the International Conference on Learning Representations (ICLR) (2022)
15. Schmidl, S., Wenig, P., Papenbrock, T.: Anomaly detection in time series: a comprehensive evaluation. Proc. VLDB Endow. 15(9), 1779–1797 (2022)
16. Chen, H., Finin, T., Joshi, A.: The SOUPA ontology for pervasive computing. In: Ontologies for Agents: Theory and Experiences, pp. 233–258. Springer (2005)
17. Arief-Ang, I.B., Salim, F.D., Hamilton, M.: DA-HOC: semi-supervised domain adaptation for room occupancy prediction using CO2 sensor data. In: Proceedings of the ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys), pp. 1–10 (2018)
18. Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., Gao, R.X.: Deep learning and its applications to machine health monitoring. Mech. Syst. Signal Process. 115, 213–237 (2019)
19. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 7th SIAM International Conference on Data Mining, pp. 443–448 (2007)
20. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. 46(4), 1–37 (2014)
21. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., et al.: Hidden technical debt in machine learning systems. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 28 (2015)
22. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. 41(3), 1–58 (2009)
23. Nassif, A.B., Talib, M.A., Nasir, Q., Dakalbab, F.M.: Machine learning for anomaly detection: a systematic review. IEEE Access 9, 78658–78700 (2021)
24. Cook, A.A., Mısırlı, G., Fan, Z.: Anomaly detection for IoT time-series data: a survey. IEEE Internet Things J. 7(7), 6481–6494 (2020)
25. Wang, H., Bah, M.J., Hammad, M.: Progress in outlier detection techniques: a survey. IEEE Access 7, 107964–108000 (2019)
26. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the Internet of Things: a survey. IEEE Commun. Surv. Tutor. 16(1), 414–454 (2014)
27. Liu, J., Shen, H., Narman, H.S., Chung, W., Lin, Z.: A survey of mobile crowdsensing techniques. ACM Trans. Cyber-Phys. Syst. 2(3), 1–26 (2018)
28. Raza, U., Kulkarni, P., Sooriyabandara, M.: Low power wide area networks: an overview. IEEE Commun. Surv. Tutor. 19(2), 855–873 (2017)
29. Mahdavinejad, M.S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., Sheth, A.P.: Machine learning for Internet of Things data analysis: a survey. Digit. Commun. Netw. 4(3), 161–175 (2018)
30. Ngo, D.T., Park, S., Sabharwal, A., Diggavi, S.: Multihop full duplex networks with self-interference cancellation. IEEE Trans. Wirel. Commun. 13(4), 1847–1860 (2014)
31. Gao, J., Saha, B., Sebastiani, F.: On the convergence of bounded-error distributional temporal difference learning for Internet of Things predictive modeling. IEEE Trans. Ind. Electron. 69(2), 1840–1848 (2022)
32. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., et al.: N-BaIoT — network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Comput. 17(3), 12–22 (2018)
33. Yin, C., Zhu, Y., Fei, J., He, X.: A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5, 21954–21961 (2017)
34. Luo, X., Liu, D., Bhatt, A.: Efficient context-aware service selection in IoT-cloud environments. IEEE Internet Things J. 9(18), 17512–17525 (2022)
35. Pimentel, M.A.F., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Process. 99, 215–249 (2014)
36. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM Comput. Surv. 54(3), 1–33 (2022)

37. Wang, Y., Zhao, H., Zhong, F., Xie, X.: Context-aware anomaly detection for smart environments based on GAN. IEEE Internet Things J. 10(5), 4023–4035 (2023)
38. Simonetto, A., Dall'Anese, E., Mokhtari, A., Leus, G.: Personalization of context-aware services for edge intelligence. IEEE Trans. Signal Inf. Process. Netw. 8, 724–739 (2022)
39. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer 36(1), 41–50 (2003). https://doi.org/10.1109/MC.2003.1160055
40. Boutaba, R., Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrada-Solano, F., Caicedo, O.M.: A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. J. Internet Serv. Appl. 9, 16 (2018). https://doi.org/10.1186/s13174-018-0087-2
41. Leivadeas, A., Falkner, M.: A survey on intent-based networking. IEEE Commun. Surv. Tutor. 25(1), 625–655 (2023). https://doi.org/10.1109/COMST.2022.3215919
42. Benzaid, C., Taleb, T.: AI-driven zero touch network and service management in 5G and beyond: challenges and research directions. IEEE Netw. 34(2), 186–194 (2020)
43. Leivadeas, A., Falkner, M.: Network assurance in intent-based networking data centers with machine learning techniques. In: Proceedings of the 17th International Conference on Network and Service Management (CNSM), pp. 1–7 (2021)
44. Dromard, J., Roudière, G., Owezarski, P.: Online and scalable unsupervised network anomaly detection method. IEEE Trans. Netw. Serv. Manag. 14(1), 34–47 (2017)
45. Xavier, B.M., Guimarães, R.S., Comarela, G., Martinello, M.: MAP4: a pragmatic framework for in-network machine learning traffic classification. IEEE Trans. Netw. Serv. Manag. 19(4), 4176–4188 (2022)
46. ETSI GS ZSM 002: Zero-touch network and Service Management (ZSM); Reference Architecture. ETSI Industry Specification Group (ISG) ZSM, Sophia Antipolis (2019)
47. Clemm, A., Ciavaglia, L., Granville, L.Z., Tantsura, J.: Intent-Based Networking — Concepts and Definitions. RFC 9315, Internet Research Task Force (IRTF) (2022). https://doi.org/10.17487/RFC9315