



AI-Based Cyber Attack Detection in the Internet of Medical Things: A Review of Edge–Cloud Deployment and On-Device Learning

Prof. Abdulkayyum Shaikh¹, Prof. Anmol Budhewar²,
Rohit Prakash Kakulte³, Akash Satish Gangurde³,
Dipak Kishor Nandan³, Shruti Jairam Waghmode³

¹Project Guide, Department of Computer Engineering, Sandip Institute of Technology and Research Centre

²Project Co-coordinator, Department of Computer Engineering, Sandip Institute of Technology and Research Centre

³Department of Computer Engineering, Sandip Institute of Technology and Research Centre, Nashik, Maharashtra, India

abdulkayyum.shaikh@sitrc.org, anmol.budhewar@sitrc.org, rohithkakulte19@gmail.com, akashgangurde384@gmail.com,
dipaknandan3@gmail.com, shrutiwaghmode100@gmail.com

Corresponding author- Prof. Abdulkayyum Shaikh

DOI: 10.5281/zenodo.23634167

Abstract: The Internet of Medical Things (IoMT) connects patient monitors, infusion pumps and wearable sensors to hospital networks. This connectivity supports continuous and remote care, but it also means that a cyber-attack on a device or gateway can expose patient data or interfere with treatment. Our final-year project aims to detect attacks such as denial of service, malware and unauthorised access by analysing IoMT network traffic with artificial intelligence. Choosing a good detection model is only part of the task. The model must also run somewhere (device, gateway, hospital server or cloud), and each choice affects response time, privacy and cost. This paper reviews the twenty studies listed in our project synopsis. They cover five themes: the edge–cloud computing continuum, edge computing for IoT, orchestration and benchmarking, distributed and federated learning, and on-device AI including TinyML, knowledge distillation and test-time adaptation. The reviewed work shows that processing near the data source shortens response time, that federated learning allows training without pooling raw data, and that compact models can run and even adapt on very small hardware. However, the studies rarely treat security detection as the workload being placed, and none of the on-device methods targets network traffic. Based on these gaps, we outline a conceptual edge–cloud framework for our project, which has not yet been validated.

Keywords: Internet of Medical Things, cyber-attack detection, edge computing, computing continuum, federated learning, TinyML, knowledge distillation.

I INTRODUCTION

A. Domain Background

Hospitals depend more and more on connected equipment. Bedside monitors report vital signs to nursing stations, infusion pumps receive dose settings over the network, and wearable sensors send readings from patients who are recovering at home. Together these devices form the Internet of Medical Things (IoMT). The benefit is clear: staff can watch many patients at once and care can continue outside the ward. The cost is also clear. Every connected device is a possible entry point for an attacker, and many medical devices have little memory or processing power to protect themselves.

B. Importance of the Problem

An attack on IoMT is not only a data problem. If a gateway is flooded with traffic, alarms may arrive late. If readings are altered in transit, a clinician may act on wrong information. Malware on a hospital network can spread to devices that are hard to patch. For these reasons our project aims to detect attacks such as denial of service, malware and unauthorised access by studying network traffic with artificial intelligence (AI), and to raise clear alerts for hospital staff.

C. Current Technological Landscape

Medical data rarely stays on a single machine. It moves from devices to nearby edge or fog nodes and then to the cloud. Researchers describe this layered arrangement as the cloud or computing continuum [1, 20]. The middle layers exist mainly to make up for cloud limitations in latency, bandwidth,

security, cost and control [20], and edge and cloud platforms are known to behave differently in performance, scalability and security [10]. AI is moving along the same path. Training and inference are being spread across devices through distributed learning [8, 11], federated learning [12, 13] and on-device or tiny machine learning [14, 17].

D. Existing Approaches and Their Limits

Most work on attack detection asks one main question: which algorithm gives the highest accuracy on a benchmark dataset? A second question receives much less attention: where should the detector run in a real hospital? If every traffic summary is sent to a distant cloud, detection can be slow and patient-related data travels further than it needs to. If the model runs on a small gateway, it may not fit in memory or may be hard to update. The placement question therefore affects speed, privacy and practicality as much as the choice of algorithm does.

E. Motivation, Purpose and Scope

Before building our system, we wanted to understand what the edge–cloud and on-device AI literature already offers for running a detector near medical devices, and what is still missing. This review examines the twenty studies listed in our project synopsis and reads them from the viewpoint of an IoMT attack detector. Their bibliographic details were checked against publisher records, and no source outside the synopsis list is used. Instead of summarising the papers one by one, we group them into five themes (Figure 1) and compare them. Specific intrusion-detection algorithms and attack datasets are outside the scope of this review; they belong to the implementation phase of our project.

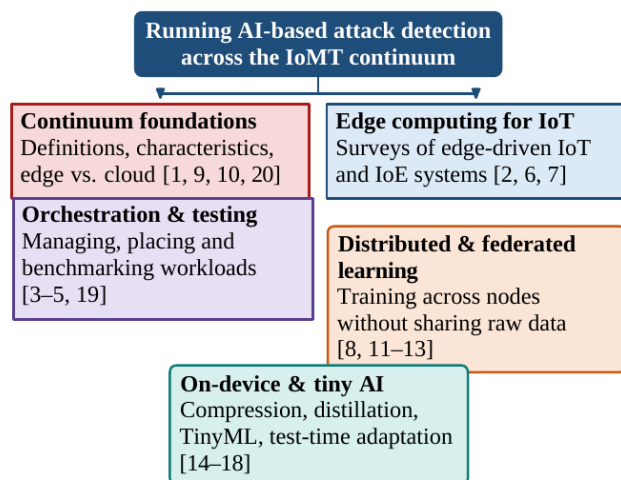


Figure 1: Themes covered in this review, with the synopsis references grouped under each.

The rest of the paper is organised as follows. Section II explains the basic concepts. Section III reviews the literature by theme. Sections IV and V compare the studies and discuss them

critically. Section VI lists the research gaps, Section VII presents our conceptual framework, and Section VIII concludes the paper.

II. BACKGROUND AND KEY CONCEPTS

A. Threats in IoMT Networks

In this paper, an attack means any network activity that tries to disrupt, spy on or take control of medical devices or the systems around them. Our synopsis focuses on three broad groups: denial of service, where a device or gateway is overloaded; malware, where harmful software spreads through the network; and unauthorised access, where an outsider or a compromised device reaches systems it should not reach. A detector is a model that looks at traffic features, such as packet counts, flow duration and byte rates, and labels each flow as normal or suspicious.

B. The Computing Continuum

The computing continuum is the chain of computing resources that runs from end devices through edge and fog nodes to the cloud [1, 20]. End devices are close to the patient but weak. Edge nodes, such as a ward gateway, have more power and still sit close to the data. The cloud has almost unlimited resources but is far away in network terms. Khalyeyev et al. describe the ideal continuum as one in which software components can move freely between these levels [9].

C. Edge versus Cloud Processing

Processing data at the edge gives faster responses and reduces the amount of data that must travel over the network, while the cloud offers scale and central management [2, 10]. For a medical network this trade-off is central: quick local decisions help safety, but training large models and keeping long-term records are easier in the cloud [20].

D. Distributed and Federated Learning

In distributed learning, the work of training or running a model is split across several machines [8, 11]. Federated learning (FL) is a special case in which each site trains on its own data and shares only model updates with a central server, so raw data never leaves the site [12, 13]. This property is attractive for hospitals, which are often not allowed to share data that involves patients.

E. Making Models Small and Adaptive

Several techniques allow AI to run on weak hardware. Model compression and hardware acceleration reduce the memory and time needed for inference [15]. TinyML refers to machine learning on microcontroller-class devices [17]. Knowledge distillation trains a small “student” model to copy a large “teacher” model [16]. Test-time adaptation lets a deployed model adjust to new input data without full retraining [18].

These ideas are the building blocks for a detector that can live on a gateway.

F. Requirements for an IoMT Attack Detector

From the project goals in our synopsis and the concepts above, we derive six requirements that guide how we read the literature. (R1) Low latency: suspicious traffic should be flagged quickly, so that a flooded gateway or altered reading is noticed before it affects care. (R2) Privacy: traffic that involves patient devices should travel as little as possible. (R3) Small footprint: at least part of the detector must fit on gateway-class hardware. (R4) Few false alarms: staff will ignore a system that raises alerts all the time. (R5) Explainability: an alert should say why a flow was flagged. (R6) Adaptability: the detector should cope with traffic that changes as devices and software change. Each theme in Figure 1 speaks to some of these needs: edge processing to R1 and R2 [2, 10], compression and TinyML to R3 [15, 17], distillation with XAI to R5 [16], test-time adaptation to R6 [18], and federated learning to R2 at the level of whole hospitals [12, 13]. R4 is mainly a property of the detection model itself and is handled through our evaluation plan in Section VII.

III. LITERATURE REVIEW

A. The Computing Continuum

Several studies try to define the continuum and describe its main properties. Moreschini et al. collect the different meanings of “cloud continuum” found in the literature and propose a single definition, which helps later work use the term consistently [1]. Al-Dulaimy et al. trace how the continuum grew out of IoT and cloud computing, explain the role of its middle layers and discuss applications and open challenges [20]. Khalyeyev et al. approach the topic from software architecture. They describe components that can migrate across the computing hierarchy, but they also warn that this is still more a vision than an established paradigm [9]. Omran et al. compare edge and cloud computing directly and report that processing near the source gives faster responses, while conventional cloud setups can struggle with the performance, scalability and security needs of current applications [10].

Taken together, these studies give us a shared vocabulary and a clear message: a system does not have to run in one place. Their limitation, from our point of view, is that they stay at the level of platforms and use cases. None of them treats an attack detector as the application whose parts must be placed.

B. Edge Computing for IoT

Two surveys by Kong et al. examine edge computing from different scales. The first looks at edge-computing-driven IoT and explains how moving computation toward the network edge serves applications that need fast responses [2]. The

second widens the view to the “Internet of Everything” and organises research on how edge resources can serve very large numbers of connected things [6]. Gkonis et al. review IoT–edge–cloud continuum systems as a whole, covering their current status, typical use cases and open issues [7].

These surveys agree that the edge is the natural first stop for data produced by connected devices. For our project the implication is simple: because medical-device traffic is created at the edge, its first inspection can also happen there, without a round trip to the cloud. The surveys, however, are broad. They cover many application areas and do not study medical networks or security monitoring in detail.

C. Orchestration, Management and Benchmarking

Once a system spans several layers, something must decide which component runs where and must keep it running. Ullah et al. build a taxonomy of orchestration in the Cloud-to-Things continuum and point out that IoT applications often need sensors spread over many locations together with a mix of local and cloud resources [3]. Chiang et al. survey how edge computing for IoT is managed and orchestrated [4], while Maia et al. argue that computing, caching and communication should be handled together rather than as separate problems along the cloud-to-edge path [5].

On the practical side, Jansen et al. present Continuum, a framework that automatically deploys cloud, edge and end-point infrastructure and benchmarks applications on it [19]. Their motivation is that developers rarely have access to all these resources when designing software. This is also our situation: a student team does not have a hospital network to experiment on. Emulation of this kind is therefore a realistic way to compare different detector placements. The orchestration surveys give useful structure, but they do not discuss workloads with strict security deadlines.

D. Distributed and Federated Learning

Rosendo et al. carry out a systematic review of distributed intelligence on the edge-to-cloud continuum [8], and Filho et al. review distributed machine learning in edge computing [11]. Both show that training and inference no longer need to happen on one central server and map the techniques that make this possible. Federated learning takes the idea further by keeping data where it is produced. Prigent et al. study how FL can be run across the whole computing continuum and list the system challenges that remain open [12]. Pfeiffer et al. focus on devices with limited and uneven computing power [13], which closely matches hospitals that run old and new equipment side by side.

This theme is directly relevant to IoMT security. Hospitals are often unable to share traffic that involves patient devices, yet a detector trained at one site may not recognise attacks seen

elsewhere. FL suggests a way for several hospitals to improve one shared detector without exchanging raw traffic. The reviewed studies, however, discuss FL in general terms and do not test it on intrusion detection.

E. On-Device and Tiny AI

The last theme deals with making models small enough for weak hardware. Incel and Bursa review deep learning that runs directly on mobile and wearable sensing devices [14]. Shuvo et al. review methods for speeding up deep learning inference on resource-constrained edge devices, including model compression and hardware accelerators [15]. Ray surveys TinyML, which brings machine learning down to microcontroller-class hardware, and discusses its current state and prospects [17].

Two method papers add abilities beyond simply shrinking a model. Cantini et al. propose DiXtill, which uses explainable AI to guide knowledge distillation from a large language model into a lightweight student model that can still explain its own

outputs [16]. Jia et al. present TinyTTA, which allows a deployed model to keep adapting to new data on a microcontroller with as little as 512 KB of memory [18]. Hospital traffic changes over time as devices, software and usage patterns change, so a small model that can both explain its decisions and adapt after deployment is attractive for attack detection. The limitation is that these methods are demonstrated on tasks such as sensing or language processing [14,16], not on network traffic.

IV. COMPARATIVE ANALYSIS

Table 1 compares the twenty studies. The “Input/evidence” column states what each study works from, and the last column is our own reading of what each study leaves open for an IoMT attack detector; it is not a claim made by the original authors. Table 2 then groups the studies into technique families and lists the main strength, the main limitation and the related research gap of each family.

Table 1: Comparison of the reviewed studies

Ref.	Year	Technique / focus	Input / evidence	Key finding	Limitation for IoMT attack detection (our reading)
[1]	2022	Definition of cloud continuum	Published literature	Proposes a common definition of the continuum	Conceptual; no detection workload
[2]	2023	Edge-driven IoT survey	Published literature	Edge computation supports fast-response IoT	Security not the central workload
[3]	2023	Cloud-to-Things orchestration taxonomy	Published literature	Classifies orchestration features and directions	Not applied to detector placement
[4]	2023	Edge management and orchestration	Published literature	Reviews managing edge resources for IoT	No medical or security case
[5]	2024	Computing, caching, communication	Published literature	Treats the three resources jointly	Traffic-inspection cost not addressed
[6]	2022	Edge for Internet of Everything	Published literature	Organises edge research at large scale	General IoE, not hospital networks
[7]	2023	IoT–edge–cloud systems	Published literature	Status, use cases and open issues	Detection not a main use case
[8]	2022	Distributed intelligence	Published literature	AI spread across edge-to-cloud	No intrusion-detection focus
[9]	2022	Edge–cloud characterisation	Published literature	Continuum still more vision than paradigm	Little mature tooling to build on
[10]	2024	Edge vs. cloud comparison	Published literature	Edge gives faster response; cloud has limits	Security treated as platform property
[11]	2022	Distributed ML at the edge	Published literature	Maps distributed ML techniques	Not specific to traffic data
[12]	2024	Federated learning on continuum	Published literature	FL systems and open challenges	Not applied across hospitals for IDS
[13]	2023	FL on constrained devices	Published literature	FL with heterogeneous weak devices	No security workload studied
[14]	2023	On-device deep learning	Published literature	DL on mobile and wearable sensing	Sensing tasks, not network traffic
[15]	2023	Inference acceleration	Published literature	Compression and accelerators for edge DL	Not evaluated on traffic classifiers

Ref.	Year	Technique / focus	Input / evidence	Key finding	Limitation for IoMT attack detection (our reading)
[16]	2024	XAI-driven distillation (DiXtill)	Text / language-model tasks	Small students that explain outputs	Not applied to security alerts
[17]	2022	TinyML review	Published literature	ML feasible on microcontrollers	No intrusion-detection examples
[18]	2024	Test-time adaptation (TinyTTA)	Data on edge/MCU devices	Adaptation within 512 KB memory	Not tested on network traffic
[19]	2023	Benchmarking tool (Continuum)	Emulated cloud-edge infrastructure	Automates deployment and benchmarking	Detector workloads not included
[20]	2024	Computing continuum survey	Published literature	Middle layers address latency, security	Detection placement not studied

Table 2: Technique families: strengths, limitations and related gaps

Technique family	References	Main strength for IoMT detection	Main limitation in reviewed work	Gap
Continuum definitions and edge vs. cloud	[1, 9, 10, 20]	Common vocabulary; evidence that edge processing is faster and keeps data local	Platform-level view; security treated as a property, not a workload	G1
Edge computing surveys	[2, 6, 7]	Edge is the natural first stop for device data	Broad coverage; medical networks not studied in detail	G1
Orchestration and management	[3–5]	Structured ways to decide what runs where	No workloads with strict security deadlines	G1
Emulation and benchmarking	[19]	Compare placements without real hospital hardware	Detector workloads not included	G1
Distributed learning	[8, 11]	Training and inference spread across layers	General ML focus; not traffic-specific	G1, G2
Federated learning	[12, 13]	Multi-site training without sharing raw data	No intrusion-detection case studies	G3
Compression, on-device DL and TinyML	[14, 15, 17]	Models small enough for gateways and microcontrollers	Sensing tasks rather than traffic classification	G2
XAI-guided distillation	[16]	Small models that still explain outputs	Demonstrated on language tasks only	G4
Test-time adaptation	[18]	Deployed models adjust to new data on MCUs	Not tested on changing network traffic	G5

Table 3 looks at the same studies from a different angle. It maps each theme to the six detector requirements defined in Section II-F. Two points are worth noting. Every requirement except R4 is supported by at least one theme, so the building blocks for a practical detector already exist in the literature. R4, a low false alarm rate, is not addressed by any of the twenty studies, because none of them evaluates a detection task. This requirement must therefore come from the design and testing of our own detection model.

Table 3: Coverage of detector requirements by theme (our reading)

Theme	R1	R2	R3	R4	R5	R6
Continuum / edge vs. cloud [1, 9, 10, 20]	✓	✓	–	–	–	–
Edge computing for IoT [2, 6, 7]	✓	✓	–	–	–	–
Orchestration & benchmarking [3–5, 19]	◦	–	◦	–	–	–
Distributed & federated learning [8, 11–13]	–	✓	◦	–	–	–
Compression, on-device DL, TinyML [14, 15, 17]	◦	◦	✓	–	–	–
XAI-guided distillation [16]	–	–	✓	–	✓	–
Test-time adaptation [18]	–	–	✓	–	–	✓

✓ = directly addressed; ◦ = partly or indirectly addressed; – = not addressed. R1 latency, R2 privacy, R3 small footprint, R4 few false alarms, R5 explainability, R6 adaptability.

Three patterns stand out from the comparison. First, most of the works are surveys or reviews [1–8, 11–15, 17, 20]. Only three contribute a concrete method or tool that can be reused [16, 18, 19]. Second, the studies consistently report that processing close to the data source improves speed and control [2, 10, 20]. Third, none of the twenty studies uses intrusion or attack detection as its main workload. Security appears mainly as one property of the platform [10, 20].

The input column also shows a clear split. The surveys work from published literature, so they describe what is possible but do not report new measurements. The three method and tool papers work from concrete data or infrastructure [16, 18, 19], but on tasks other than traffic classification. As a result, the reviewed literature offers strong guidance on architecture and techniques but no direct evidence about how an IoMT attack detector behaves at each layer.

V. CRITICAL ANALYSIS

A. Most Promising Directions

Based on the reviewed evidence, three directions look most promising for our project. The first is edge-first processing, because several studies agree that it shortens response time and keeps data local [2, 10, 20]. The second is model compression and TinyML, because they make it realistic to place a first-stage detector on gateway-class hardware [15, 17]. The third is emulation-based benchmarking, because it allows placements to be compared without access to real hospital hardware [19].

B. Common Strengths

The literature gives a clear and shared picture of the continuum [1, 20], structured ways to decide where software runs [3, 4], a credible route to learning from sensitive data without collecting it centrally [12, 13], and evidence that useful models can run and even adapt on very small devices [17, 18]. Most of the studies are also recent, published between 2022 and 2024, so they reflect the current state of the field.

C. Recurring Limitations

Four limitations appear again and again. Security is treated as a property of the platform rather than as a workload with its own timing and accuracy needs [10,20]. The on-device methods are tested on tasks such as sensing or language processing rather than network traffic [14, 16]. Federated learning is discussed in general terms, without intrusion-detection case studies [12, 13]. Finally, most surveys do not report new measurements, so their guidance on placement remains qualitative.

D. Points of Tension

The studies do not fully agree on how mature the continuum is. Moreschini et al. and Al-Dulaimy et al. present it as a defined and growing paradigm [1, 20], while Khalyeyev et al. describe it as still more vision than reality [9]. There is also a practical tension between edge and cloud. The edge is faster and keeps data local [10], but the cloud remains the easier place for heavy training and central management [20]. A detector design must therefore split work between layers rather than choose only one.

E. Unresolved Challenges

Two practical challenges remain open. Ready-made tooling for the continuum cannot be assumed, because the paradigm is still forming [9]. Developers also rarely have access to the full range of continuum hardware [19], a limitation our team shares. Both challenges favour a simple, emulated first version over a complex multi-layer deployment.

F. Lessons for IoMT Attack Detection

Reading the twenty studies together, we draw four practical lessons. First, a detector should be designed as a set of parts that can live on different layers, not as one block of code, because the continuum literature treats placement as a design choice [1,9,20]. Second, the part that sees every packet must be the smallest and fastest, since it sits where resources are most limited [15, 17]. Third, heavier analysis and explanation can wait for the small share of traffic that looks suspicious, which keeps most data local and limits load on the hospital server [10]. Fourth, the system should be tested on an emulated continuum before any real deployment, because realistic

hardware is rarely available during development [19]. These lessons directly shape the framework in Section VII.

VI. RESEARCH GAPS

Based on the analysis above, we identify five research gaps. Figure 2 shows how each gap is handled in our conceptual framework.

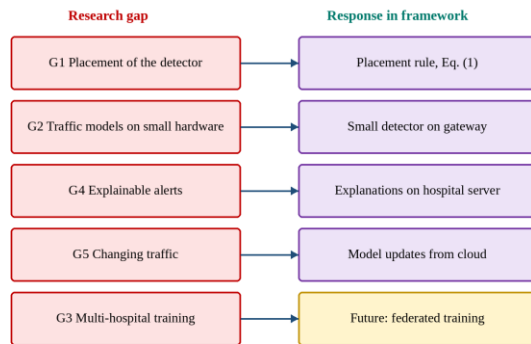


Figure 2: Mapping of the identified research gaps to the elements of the proposed conceptual framework (G3 is kept as future work).

G1: Detection as a continuum workload. The continuum literature describes platforms [1, 9, 20] and orchestration in general [3, 4], but does not study where an attack detector should be placed or how detection should be split between gateway, hospital server and cloud.

G2: Traffic models on small hardware. Compression, TinyML and on-device deep learning are well reviewed [14, 15, 17], but these studies target sensing or language tasks [14, 16] rather than classifying network traffic on a medical gateway.

G3: Privacy-preserving training across hospitals. Federated learning is well developed in general [12, 13], yet its use for intrusion detection across several hospitals is not covered in the reviewed work.

G4: Explainable alerts from small models. DiXtill shows that a distilled model can explain its own outputs [16], but no reviewed study applies this idea to security alerts, where staff need to understand why a flow was flagged.

G5: Adapting to changing traffic. Test-time adaptation works on microcontrollers [18]; whether it helps a detector facing new kinds of network traffic remains open.

Our project mainly targets G1 and G2. It also takes first steps toward G4 and G5, and leaves G3 as future work.

The gaps are not equally urgent. G1 and G2 come first because without a clear placement rule and a model that fits on a gateway, the other improvements have nowhere to run. G4 matters for daily use, since an alert that cannot be explained is hard for staff to trust or act on. G5 becomes important once a detector has been running for some time and the traffic it sees

starts to drift away from its training data. G3 needs cooperation between several hospitals and a secure aggregation setup, which is beyond the scope of a single final-year project, but the framework leaves room for it in the cloud layer.

VII. PROPOSED CONCEPTUAL FRAMEWORK

A. What the Literature Provides

The reviewed research supplies a layered view of computing [1, 20], the case for edge processing [2, 10], ways to compress, explain and adapt models [15–18], and tools for testing placements [19]. Our project proposes to combine these ideas by assigning each step of the detection pipeline in our synopsis to the most suitable layer. This is a conceptual design. It has not been implemented or evaluated.

B. Detection Pipeline

Figure 3 shows the logical steps of the detector. Network traffic from medical devices is captured, cleaned and grouped into flows. Flow features are extracted and passed to a small first-stage model that screens all traffic. Flows that look suspicious are passed to a larger second-stage model, which confirms the decision and produces a short explanation. The final output is an alert that hospital staff can review.

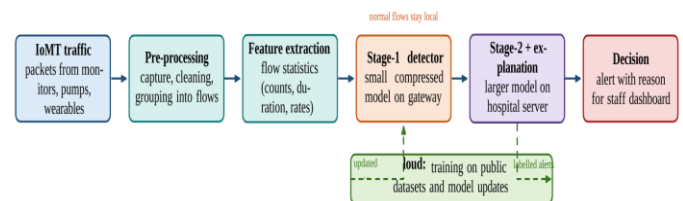


Figure 3: Conceptual detection pipeline used in this review to relate the literature to our project (design proposal, not yet validated).

C. Layered Architecture

Figure 4 places these steps on four layers. Medical devices send their traffic through a gateway, which captures packets, extracts flow features and runs the first-stage detector. This model is kept within gateway limits by compression or distillation [15, 17]. Only suspicious flows are forwarded, together with their scores, so most traffic stays local, in line with the edge benefits reported in [10, 20]. The hospital edge server runs the second-stage model on flagged flows, attaches an explanation to each alert and stores alerts for the administrator's dashboard. The cloud trains full models on the public datasets named in our synopsis (CICIDS2017, CICIOT2023, NSL-KDD, UNSW-NB15 and Bot-IoT) and sends updated models back down. Federated aggregation across hospitals is left as a later extension [12, 13].

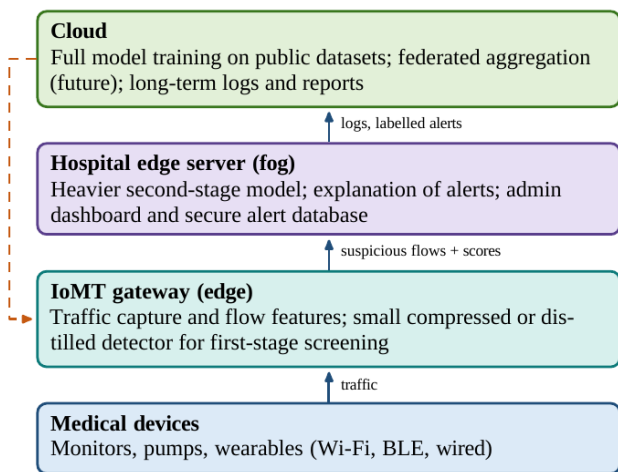


Figure 4: Proposed conceptual architecture for AI-based attack detection in IoMT (design proposal, not yet validated).

D. Placement Rule and Evaluation Plan

Placement follows a simple rule. A detection step i with processing time c_i , network delay n_i , queuing overhead q_i and deadline d_i may run on a remote layer only if

$$c_i + n_i + q_i \leq d_i. \quad (1)$$

Time-critical steps therefore stay on the gateway, and steps with relaxed deadlines, such as retraining, can move to the cloud.

Evaluation will follow the plan in our synopsis: precision, recall, F1-score, false positive rate and detection time. With TP, FP, TN and FN denoting true and false positives and negatives,

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}), \text{ Recall} = \text{TP} / (\text{TP} + \text{FN}), \quad (2)$$

$$\text{F1} = 2 \cdot \text{Precision} \cdot \text{Recall} / (\text{Precision} + \text{Recall}), \text{ FPR} = \text{FP} / (\text{FP} + \text{TN}). \quad (3)$$

A low false positive rate matters as much as high recall, because too many false alarms would tire hospital staff. Since hospital hardware is not available, alternative placements will be compared in an emulated continuum using a benchmarking approach similar to [19].

Our conceptual framework responds by placing a small detector on the gateway, a larger explaining model on the hospital server and training in the cloud, with a simple deadline-based rule for deciding placement. The review also points to concrete directions for future work:

- Federated intrusion detection: train one detector across several hospitals without moving their traffic, and study how uneven device power affects training [12, 13].
- Explainable small detectors: test whether XAI-guided distillation can keep useful alert explanations in a gateway-sized model [16].

- Adaptive detection: apply test-time adaptation so that a deployed detector can follow gradual changes in hospital traffic [18].
- Automatic placement: use orchestration methods to move detector components between layers as load and deadlines change [3–5].
- Emulation and real testing: benchmark emulated placements side by side with a small real deployment to check how well emulation predicts real behaviour [19].

The framework will be built and evaluated during the implementation phase of our project, beginning with the gateway-level detector and the placement rule.

VIII. CONCLUSION AND FUTURE DIRECTIONS

This review examined twenty studies on the computing continuum, edge computing, orchestration, distributed and federated learning, and on-device AI, and read them from the standpoint of IoMT attack detection. The literature supports three main ideas: process traffic close to medical devices, train without centralising sensitive data, and run compact models on small hardware. Most of the reviewed works are surveys, and the few method papers are demonstrated on tasks other than network traffic. The field therefore does not yet treat attack detection as a continuum workload, which is the main space our project aims to address.

IX REFERENCE

- [1] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi, “Cloud continuum: The definition,” *IEEE Access*, vol. 10, pp. 131876–131886, 2022.
- [2] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, “Edge-computing-driven Internet of Things: A survey,” *ACM Comput. Surv.*, vol. 55, no. 8, pp. 1–41, 2023.
- [3] A. Ullah, T. Kiss, J. Kovács, F. Tusa, J. Deslauriers, H. Dagdeviren, R. Arjun, and H. Hamzeh, “Orchestration in the Cloud-to-Things compute continuum: Taxonomy, survey and future directions,” *J. Cloud Comput.*, vol. 12, no. 1, Art. no. 135, 2023, doi: 10.1186/s13677-023-00516-5.
- [4] Y. Chiang, Y. Zhang, H. Luo, T.-Y. Chen, G.-H. Chen, H.-T. Chen, Y.-J. Wang, H.-Y. Wei, and C.-T. Chou, “Management and orchestration of edge computing for IoT: A comprehensive survey,” *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14307–14331, 2023.
- [5] A. Maia, A. Boutouchent, Y. Kardjadja, M. Gherari, E. G. Soyak, M. Saqib, K. Boussekar, I. Cilbir, S. Habibi, S. O. Ali, W. Ajib, H. Elbiaze, O. Erçetin, Y. Ghamri-Doudane, and R. Glitho, “A survey on integrated computing, caching, and

communication in the cloud-to-edge continuum,” *Comput. Commun.*, vol. 219, pp. 128–152, 2024.

[6] X. Kong, Y. Wu, H. Wang, and F. Xia, “Edge computing for Internet of Everything: A survey,” *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23472–23485, 2022.

[7] P. Gkonis, A. Giannopoulos, P. Trakadas, X. Masip-Bruin, and F. D’Andria, “A survey on IoT-edge-cloud continuum systems: Status, challenges, use cases, and open issues,” *Future Internet*, vol. 15, no. 12, Art. no. 383, 2023.

[8] D. Rosendo, A. Costan, P. Valduriez, and G. Antoniu, “Distributed intelligence on the edge-to-cloud continuum: A systematic literature review,” *J. Parallel Distrib. Comput.*, vol. 166, pp. 71–94, 2022.

[9] D. Khalyeyev, T. Bureš, and P. Hnětynka, “Towards characterization of edge-cloud continuum,” in *Software Architecture: ECSA 2022 Tracks and Workshops, LNCS vol. 13928*, Springer, 2023, pp. 215–230, doi: 10.1007/978-3-031-36889-9_16.

[10] A. H. Omran, Y. M. Abid, and B. I. Bakri, “Edge computing vs. cloud computing: Evaluating performance, scalability, and security in modern applications,” *CyberSystem J.*, vol. 1, no. 1, pp. 1–8, 2024, doi: 10.57238/fpj44896.

[11] C. P. Filho, E. Marques, V. Chang, L. dos Santos, F. Bernardini, P. F. Pires, L. Ochi, and F. C. Delicato, “A systematic literature review on distributed machine learning in edge computing,” *Sensors*, vol. 22, no. 7, Art. no. 2665, 2022.

[12] C. Prigent, A. Costan, G. Antoniu, and L. Cudennec, “Enabling federated learning across the computing continuum: Systems, challenges and future directions,” *Future Gener. Comput. Syst.*, vol. 160, pp. 767–783, 2024.

[13] K. Pfeiffer, M. Rapp, R. Khalili, and J. Henkel, “Federated learning for computationally constrained heterogeneous devices: A survey,” *ACM Comput. Surv.*, vol. 55, no. 14s, pp. 1–27, 2023.

[14] O. D. Incel and S. Ö. Bursa, “On-device deep learning for mobile and wearable sensing applications: A review,” *IEEE Sensors J.*, vol. 23, no. 6, pp. 5501–5512, 2023.

[15] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, “Efficient acceleration of deep learning inference on resource-constrained edge devices: A review,” *Proc. IEEE*, vol. 111, no. 1, pp. 42–91, 2023.

[16] R. Cantini, A. Orsino, and D. Talia, “XAI-driven knowledge distillation of large language models for efficient deployment on low-resource devices,” *J. Big Data*, vol. 11, Art. no. 63, 2024, doi: 10.1186/s40537-024-00928-3.

[17] P. P. Ray, “A review on TinyML: State-of-the-art and prospects,” *J. King Saud Univ. – Comput. Inf. Sci.*, vol. 34, no. 4, pp. 1595–1623, 2022.

[18] H. Jia, Y. D. Kwon, A. Orsino, T. Dang, D. Talia, and C. Mascolo, “TinyTTA: Efficient test-time adaptation via early-exit ensembles on edge devices,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, 2024, doi: 10.52202/079017-1371.

[19] M. Jansen, L. Wagner, A. Trivedi, and A. Iosup, “Continuum: Automate infrastructure deployment and benchmarking in the compute continuum,” in *Companion of the ACM/SPEC Int. Conf. Performance Engineering (ICPE)*, 2023, pp. 181–188, doi: 10.1145/3578245.3584936.

[20] A. Al-Dulaimy, M. Jansen, B. Johansson, A. Trivedi, A. Iosup et al., “The computing continuum: From IoT to the cloud,” *Internet of Things*, vol. 27, Art. no. 101272, 2024, doi: 10.1016/j.iot.2024.101272.