Testbeds and Evaluation Frameworks for Anomaly Detection within Built Environments: A Systematic Review

MOHAMMED ALOSAIMI, Cardiff University, UK
OMER RANA, Cardiff University, UK
CHARITH PERERA, Cardiff University, UK

The Internet of Things (IoT) has revolutionized built environments by enabling seamless data exchange among devices such as sensors, actuators, and computers. However, IoT devices often lack robust security mechanisms, making them vulnerable to cyberattacks, privacy breaches, and operational anomalies caused by environmental factors or device faults. While anomaly detection techniques are critical for securing IoT systems, the role of testbeds in evaluating these techniques has been largely overlooked. This systematic review addresses this gap by treating testbeds as first-class entities essential for the standardized evaluation and validation of anomaly detection methods in built environments. We analyze testbed characteristics, including infrastructure configurations, device selection, user-interaction models, and methods for anomaly generation. We also examine evaluation frameworks, highlighting key metrics and integrating emerging technologies such as edge computing and 5G networks into testbed design. By providing a structured and comprehensive approach to testbed development and evaluation, this paper offers valuable guidance to researchers and practitioners in enhancing the reliability and effectiveness of anomaly detection systems. Our findings contribute to the development of more secure, adaptable, and scalable IoT systems, ultimately improving the security, resilience, and efficiency of built environments.

CCS Concepts: • **Security and privacy → Intrusion/anomaly detection and malware mitigation**; • **General and reference → Surveys and overviews**; **Metrics**; **Evaluation**.

Additional Key Words and Phrases: smart home, datasets, evaluation metric, network feature, smart building

ACM Reference Format:
Mohammed Alosaimi, Omer Rana, and Charith Perera. 2025. Testbeds and Evaluation Frameworks for Anomaly Detection within Built Environments: A Systematic Review . *ACM Comput. Surv.* 0, 0, Article 0 (March 2025), 35 pages. https://doi.org/10.1145/3722213

1 INTRODUCTION

The **Internet of Things (IoT)** enables devices, or 'things,' such as sensors, actuators, and computers, to autonomously exchange data while monitoring or interacting with their surroundings [1]; however, these IoT devices often face significant power constraints [2]. By 2025, IoT could impact the economy by up to $11 trillion, with an estimated 27 billion devices deployed [3]. However, with this growth, IoT devices face increasing cyber threats, such as unauthorized access and privacy

Authors' addresses: Mohammed Alosaimi, alosaimimm1@cardiff.ac.uk, Cardiff University, Senghennydd Rd, Cardiff CF24 4AX, Cardiff, UK; Omer Rana, Cardiff University, Senghennydd Rd, Cardiff CF24 4AX, Cardiff, UK, ranaof@cardiff.ac.uk; Charith Perera, Cardiff University, Senghennydd Rd, Cardiff CF24 4AX, Cardiff, UK, Pererac@cardiff.ac.uk.

Table 1. Summary of Existing Surveys

Paper	Method	Focus	AD-T	Testbed	Dataset	EM	CUT
Cook et al. [11]	Survey	Time-Series Data	●	○	○	○	○
Himeur et al. [12]	Review	Energy Consumption in Buildings	●	○	○	○	○
Erhan et al. [13]	Review	Sensor System	●	○	○	○	○
Moustafa et al. [14]	Survey	Network Anomaly Detection	●	○	●	●	●
Gaddam et al. [15]	Survey	Sensor Faults and Outliers	●	○	○	○	○
da Costa et al. [16]	Survey	Intrusion Detection in IoT	●	○	●	◐	○
Luo et al. [17]	Systematic Survey	Cyber-Physical Systems	●	●	●	●	○
Benkhelifa et al. [18]	Review	Intrusion Detection Systems	●	○	○	○	○
Fahim & Sillitti [19]	Systematic Review	Intelligent Inhabitant Environments	●	○	○	●	○
Taha & Hadi [20]	Review	Categorical Data	●	○	●	○	○
Fernando et al. [21]	Survey	Medical Data	●	○	●	●	○

○ No, it does not discuss this topic, ● Yes, it discusses this topic ◐ partially discusses this topic. "**AD-T**"=Anomaly Detection Techniques. "**EM**"=Evaluation Metrics. "**CUT**"=Commonly Used Tools in the context of testbed and anomaly detection.

breaches involving voice assistants or cameras [4]. Given that IoT devices are spread throughout the home, the risks of data privacy and security increase [5]. If compromised, attackers could alter data or control devices, potentially unlocking doors or assessing occupancy through network traffic [6]. Consequently, detecting anomalies is critical to protecting data integrity and privacy.

Detecting cyber or physical threats requires a system that monitors data for anomalies. According to Barnett and Lewis, “an outlier is an observation (or subset of observations) that appears to be inconsistent with the remainder of that set of data” [7], while Cook et al. define an anomaly in IoT as a measurable consequence of an unexpected change outside the norm [8]. Detecting anomalies is challenging because the concept of "normal" can change over time or due to environmental factors. Trends or seasonal shifts might appear abnormal even when they are not, while certain conditions, such as someone lying on the kitchen floor, are clear anomalies needing immediate action [9]. This complexity means that effective anomaly detection is often application-specific, designed to recognize unique features within a particular context [8].

Previous reviews on anomaly detection have been conducted, but our work focuses on anomaly detection algorithms specifically within built environments, emphasizing the importance of the testbed. We treat the testbed as a first-class citizen, essential for standardized and thoroughly evaluating detection techniques. Our review also covers evaluation metrics, types of anomalies, methods of anomaly creation, and smart home datasets, including data collection, duration of capture, and analysis tools.

This review is structured as follows: Section 2 provides background on anomaly types, detection techniques, and **Reinforcement Learning (RL)** for anomaly detection. Section 3 outlines the research methodology, including research questions and filtering criteria. Section 4 covers the creation of the testbed, the generation of anomalies, and the setup of the testbed for devices and users. Section 5 describes various anomaly detection mechanisms and their algorithms. Section 6 details the setup of the dataset, the network configuration, the data collection, the duration of the capture, and the analysis. Section 7 highlights tools for building testbeds or datasets, while Section 8 reviews commonly used evaluation metrics in the anomaly detection literature. Table 1 compares existing surveys, where a detailed discussion is provided in the supplementary material [10].

2 ANOMALY DETECTION

In this section, we highlight different anomaly detection types and techniques that we found in the literature. We also compare different anomaly detection techniques and highlight key points for each presented technique.

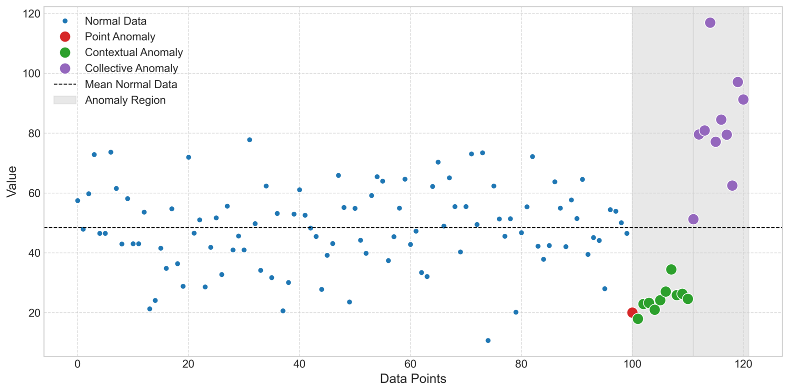

Fig. 1. A scatter plot illustrating the three types of anomalies: *point anomalies (red)*, *contextual anomalies (green)*, and *collective anomalies (purple)*. The x-axis represents data points, and the y-axis represents values. A shaded region marks an anomaly-prone area, and a dashed line represents the mean normal data. [22][23].

Table 2. Comparison Between Different Anomaly Detection Techniques

AD Techniques	Labeled	ML-model Example	Key Points
S	Yes	SGB [24]	Number of anomalies is much less than normal instances
SS	Partially	Ramp-OCSVM [25]	More common than supervised AD due to the fact that they do not need anomalous labels
US	No	GAN [26]	Produces high false positives if a dataset has more anomalous samples than normal ones.
RL	No	DQN [27]	Needs a thorough analysis of the anomaly and its relationship to the environment.
DL	Yes/No	Autoencoder [28]	Works well for complex anomalies, but needs a large amount of labeled data for training
En	Yes/No	Majority voting [29]	Uses multiple base detector to detect an anomaly
H	Yes/No	Autoencoder & SVM [30]	A combination of S, SS, or US to overcome the limitation of individual techniques
FL	Yes/No	LSTM [31]	Training is done locally, updates are only shared

"**AD**"=Anomaly Detection, "**S**"=Supervised, "**SS**"=Semi-supervised, "**US**"=Unsupervised, "**DL**"=Deep Learning, "**En**"=Ensemble, "**H**"=Hybrid, "**FL**"=Federated Learning, "**RL**"=Reinforcement Learning, "**SGB**"=Stochastic Gradient Boosting

2.1 Anomaly Types

Chandola et al. [22] describe three types of anomalies: point, contextual, and collective. A **point anomaly** is a single data point that stands out within a dataset. A **contextual anomaly** appears unusual only within its specific data context, while a **collective anomaly** involves a set of data points that seem normal individually but are abnormal as a group. For example, a temperature reading of 45°C in a dataset of 16–22°C values would be a point anomaly. Ten consecutive warm days in winter would represent a collective anomaly, while 45°C in summer could be normal in some regions but anomalous in others - making it a contextual anomaly. Figure 1 shows the difference between point, contextual, and collective anomalies.

2.2 Anomaly Detection Techniques

Anomaly detection identifies uncommon occurrences or data points that deviate significantly from normal behavior and fall into three types: supervised, semi-supervised, and unsupervised. **Supervised** detection requires labeled data to train a model to classify new data as normal or abnormal, but it faces challenges like data imbalance and accurate labeling [22]. **Semi-supervised** detection uses both labeled and unlabeled data, requiring less annotation [32] and working well for rare abnormal scenarios. **Unsupervised** detection identifies anomalies within unlabeled data based on inherent properties, assuming most data are normal, but it risks false alarms if this balance shifts [22]. Detection results are presented as either scores or labels, with data classified as either normal or anomalous.

Deep learning (DL)-based anomaly detection has gained popularity for its capacity to learn complex, high-dimensional patterns, recognize anomalies, and extract features directly from data. DL methods, including autoencoders, adapt to evolving anomalies and large datasets but require extensive labeled data and careful hyperparameter tuning. **Ensemble** anomaly detection techniques use multiple models to improve accuracy, aggregating results through methods like averaging or voting to create a robust detection system. **Hybrid** anomaly detection techniques combine supervised, semi-supervised, and unsupervised methods, blending statistical and **machine learning (ML)** algorithms for more accurate anomaly detection. **Federated Learning** (FL)-based anomaly detection techniques enable decentralized anomaly detection, where entities train models locally and share only model updates, preserving data privacy while collaborating to enhance model accuracy.

Feature selection is essential for preparing training data in anomaly detection, removing irrelevant features to enhance model accuracy, reduce storage needs, and lower computational costs [33]. In supervised feature selection, the goal is to choose a subset of features that aid in classifying data samples [33]. In contrast, unsupervised selection processes all data, with the algorithm refining feature choices iteratively [33]. These methods apply across supervised, unsupervised, and semi-supervised anomaly detection techniques [33].

RL, a recent addition to anomaly detection, involves an agent interacting with an environment and maximizing rewards through a trial-and-error approach [34]. Pang et al. [27] developed an RL framework that detects unknown anomalies in mostly unlabeled data, tested against semi-supervised and unsupervised methods like DeepSAD [35], DevNet [36], iForest [37], and REPEN [38] on datasets such as NB15 [39] and Thyroid [40][41]. However, Müller et al. [42] argue that RL in anomaly detection is inefficient due to its often simplified approach and detachment from real-world complexity. They propose guidelines for practical RL applications, like focusing on dynamic, environment-specific anomalies and eliminating rewards in the deployment stage to better simulate real conditions. Table 2 shows a comparison between the anomaly detection techniques we discussed.

3 RESEARCH METHOD

The study approach we used follows the guidelines of Barbara Kitchenham [43] and comprises the following consecutive phases: developing a set of research questions; eliciting keywords from the developed research questions to generate the search queries; selecting the databases where the search must be conducted; specifying filtering criteria such as language and year of publication; skimming through titles and abstracts to eliminate irrelevant and duplicate research papers; reviewing the papers in detail and developing extensive criteria throughout a comprehensive reading; adding different tags to relevant research papers so that it is easier to comprehend what each paper has covered; and assessing the remaining articles using the research questions that were developed at the start.

3.1 Research Questions

This systematic review aims to investigate the different anomaly detection techniques and testbed characteristics in the built environment context. For that reason, the following questions were developed:

RQ 1 What are the different types of anomalies and how are they generated?

RQ 2 What are the characteristics and properties of a smart home testbed that are useful in the context of developing and testing anomaly detection techniques?

Table 3. Digital Databases

No.	Digital Database
1	IEEE Xplore
2	ACM Digital Library
3	Science Direct
4	Research Gate
5	Elsevier

Table 4. Websites

No.	Websites
1	Research Rabbit
2	Connected Papers
3	LitMaps

Table 5. Keywords

No.	Keyword
1	smart home testbed
2	smart building testbed
3	testbed
4	smart home anomaly detection
5	smart building anomaly detection
6	testbed anomaly detection

RQ 3 What are the evaluation criteria for anomaly detection techniques within a built environment?

RQ 4 How are smart home and anomaly detection datasets created and analyzed?

3.2 Search Process

This systematic review has searched for related research papers from the following electronic databases shown in Table 3. Additionally, the literature review used three websites that help to find related research papers, as shown in Table 4. For each website, a related research paper (we used [44] as a seed) was fed to the website as a seed; then, each website generated a visual literature map that shows related references to the fed research paper after analyzing it. We chose the aforementioned research paper as a seed because we needed more research papers that discuss implementing a testbed in detail. This approach has introduced strongly related literature that covers the topic of developing and evaluating a testbed. Table 5 shows the list of keywords used during the search process. Additionally, we used the following search string along with the keywords:

```
- smart home OR building and testbed AND anomaly detection
 - smart home testbed OR smart building testbed OR testbed
```

3.3 Filtering Criteria and Quality Assessment

After conducting an initial search, we came across 429 research papers that must be filtered. We started reading titles, then abstracts, and conclusions to determine how related a certain paper is according to our literature review topic. Additionally, we created two labels within Google Scholar: one for research papers that discuss testbeds, and another for papers that discuss anomaly detection for a built environment.

4 TESTBEDS

The inherent connectivity of IoT devices has introduced a changing communication paradigm [45], emphasizing the critical need for robust testbeds designed to evaluate anomaly detection techniques in complex and dynamic environments. Having the testbed as a second-class citizen has always been the case when evaluating anomaly detection techniques. For that reason, we are introducing testbed specifications, infrastructure configurations, user and device selection criteria, and different approaches to creating anomalies as testbed characteristics that we summarize in Figure 2.

4.1 Testbed Design Specifications

Designing and implementing a testbed is not an arbitrary task. There should be guidelines to accomplish the testbed design; as such, we are trying to form testbed development guidelines

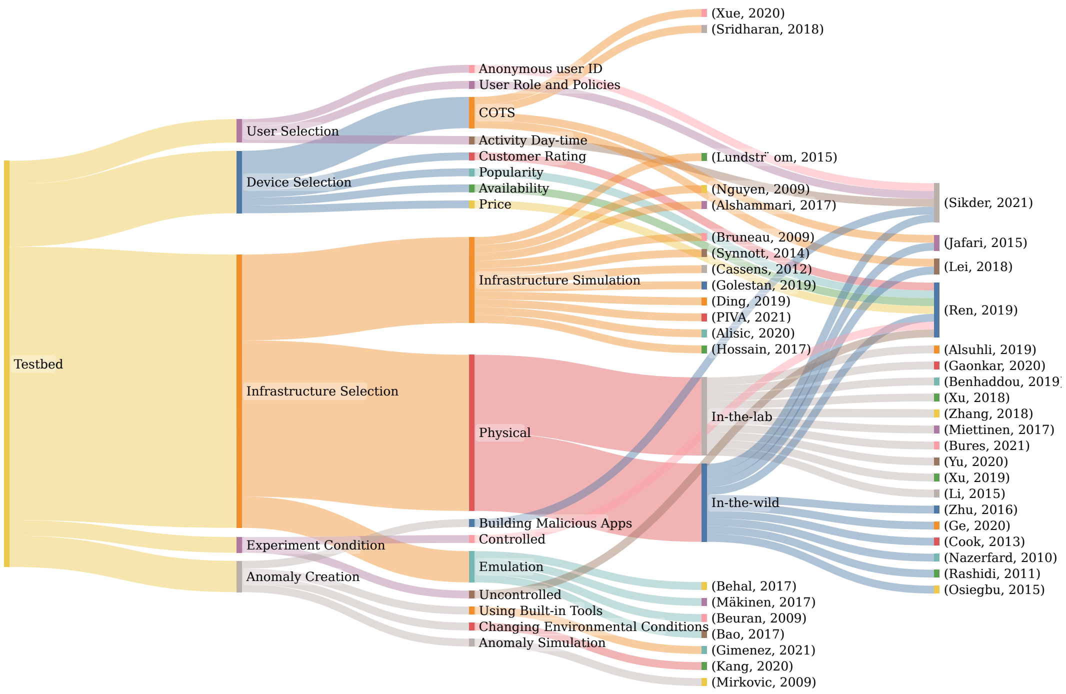

Fig. 2. A Sankey diagram categorizing testbed characteristics based on infrastructure selection, device selection, user selection, experimental conditions, and anomaly creation. Colored flows indicate relationships between categories and citations of related studies.

from the literature review we are conducting. Siboni et al. [46] illustrated several testbed design specifications:

- **Device under test:** The testbed should adopt different types of devices, such as smartwatches and fitness trackers. Even though Siboni et al. [46] were specific about wearable devices, this design requirement should also apply to smart home testbed devices.
- **Testing environments:** The testbed should be able to mimic and simulate different environments, whether static or dynamic, for instance.
- **Security testing:** There should be a variety of security tests conducted on the testbed, each targeting a different aspect of security. A vulnerability scan and penetration test should be conducted to assess and verify the security level of IoT devices.
- **Simulator array:** The testbed should be able to simulate different scenarios to generate real-time data. For instance, this design requirement could be accomplished using a simulation technique like a GPS simulator to try the testbed in different locations. Another simulation scenario is the use of a network simulator to try different wireless technologies such as Zigbee and Bluetooth.
- **Communication channels:** The testbed should support different communication technologies, wired and wireless, such as Zigbee, Wi-Fi, Bluetooth, USB, and Ethernet.
- **Protocol analysis:** The testbed should be able to analyze different types of protocols such as IPv4, TCP, and TLS.
- **Operating system compatibility:** The testbed should provide virtualization so that its devices can run their software, which supports different types of operating systems.

- **Data forensic analysis:** The testbed should maintain stored device data, including data from backup and log files, to conduct data forensic analysis.
- **Management and report mechanisms:** It is important that the testbed is equipped with management and reporting mechanisms in order to be able to control and manage the testing flow. Tools for exploring and analyzing data should be included in these report tools.
- **User intervention and automation:** The testbed should allow for user intervention and automation during all testing phases.
- **Testbed enhancement capability:** The entire testbed should be implemented as a plug-in framework to support future operations.

4.2 Infrastructure Deployment Configurations

This section discusses various testbed infrastructure deployment configurations that include physical (both in-the-wild and in-the-lab), simulation, and emulation. We also summarize the differences between each testbed deployment that Bhatia et al. [47] discussed in Table 6.

4.2.1 Physical: **In-The-Wild:**
There are several ways to implement a testbed within a built environment, with home layout influencing data generated from device interactions. Sikder et al. [48] demonstrated that different layouts—single-bedroom apartments, double-bedroom homes, and duplexes—affect user activities and, in turn, detection accuracy. They tested setups with a single **smart home system (SHS)** hub and multiple hubs (e.g., Amazon Alexa, Samsung SmartThings), finding that multi-hub setups improve accuracy across layouts, with single-bedroom homes achieving higher accuracy except in duplexes with adaptive training. Similarly, Lei et al. [49] evaluated a **virtual security button (VSButton)** across three layouts using a Wi-Fi router, motion detection module, and Echo Dot. Cook et al. [50] conducted a smart home-in-a-box study in a three-bedroom apartment with 20 participants, assessing installation times for each participant.

In-The-Lab/Prototype: Jabbar et al. [51] took a different approach to testbed deployment with their IoT@Home prototype, a smart home model built in NX10 software. It includes a master bedroom, two additional bedrooms, two bathrooms, a kitchen, a living room, and a porch. Key features include sliding windows with motorized doors, an RFID entry system, and a high-positioned water tank with an ultrasonic sensor to monitor water levels. This setup demonstrates the variety in smart home testbed layouts, from physical layouts to software-based prototypes.

4.2.2 Simulation: A smart home environment could also be simulated by monitoring smart home devices and then building a model that simulates each device's behavior. Pina et al. [52] followed this approach where they decided to use a temperature sensor, presence sensor, light bulb, simple TV, joystick, and remote control that would be installed in different rooms from E1 to E4 that have different layouts. Afterward, they used the Markov chain to simulate each device's behavior after monitoring its state and how it changes and communicates with other devices. Even though this approach might be suitable in cases where budget is an issue, there is no guarantee that the simulated states will truly reflect how devices communicate and behave in the real world.

4.2.3 Emulation: To test their DDoS detection mechanism, Behal et al. [53] used the **Common Open Research Emulator (CORE)** to increase the number of nodes in their testbed. In their emulated testbed, they deployed 16 virtual nodes and 4 soft routers using CORE, and they used 75 physical nodes in three different groups of 25 computers each, 3 D-Link physical routers, 4 L2 switches, 2 L3 switches, and 1 2-processor 8-core Linux server that acts as the victim Web server. Although this approach used software, CORE in this case, and hardware to build the testbed, the result of using such an approach is not widely adopted in the testbed community.

Table 6. Infrastructure Deployment Comparison

Characteristics	Physical	Simulation	Emulation
Realism	Optimal realism	Lack of Realism as it relies on software	More realistic as it uses real machines that run an OS, while part of the testbed is virtualized such as routers
Easy to Reconfigure	No	Yes	Moderate as part of the testbed depends on physical machines
Cost	High	Low	Moderate as part of the testbed is virtualized
Scalable	No	Yes	No
Examples	CASAS, Aegis+	OpNet, NS-2	Emulab, DETER

4.3 Selection Criteria

As we discuss testbed requirements and characteristics in the context of a built-in environment, we must also discuss users using the testbed, experiment conditions, and device selection.

4.3.1 User Criteria. In this section, we discuss user selection criteria in the context of the smart home environment so that the smart home testbed can generate realistic data that reflects the user's behavior. According to Sikder et al. [48], the following selection criteria should be considered:

- **Anonymous user ID:** Each residence or user should be assigned an ID, and this ID should be used for all communication with the user to maintain the user's privacy.
- **User role and policies:** Each residence can choose a rule to their liking. For example, a user can choose to turn on the light once a motion sensor detects a motion. Conversely, another user can choose to turn the light differently once the door is open, for example. Different rules can be assigned by different users.
- **Activity day-time:** Different time activities should be considered because an individual's activity differs in time. A working adult might have more activity on weekends than on weekdays. Hence, a change in daily routine might not be malicious behavior.

4.3.2 Device Criteria. One of the tasks to develop a testbed is to select devices for the testbed. As several devices come from the same category, there should be a way to prefer one over the other. During the selection of IoT devices, Ren et al. [54] selected IoT devices that fall into a broad range of categories based on popularity, customer rating, price, and availability for both the UK and the US because the study wanted to compare between the two testbeds in terms of IoT device information exposure in different locations. In addition, anomaly detection systems in smart home environments vary in effectiveness depending on the diversity of the device types used in training these systems. Research indicates that the training datasets characteristics significantly impact the performance of these systems [55].

4.3.3 Experiment Condition. Smart home experiments can occur in controlled or uncontrolled environments. Ren et al. [54] examined IoT device data exposure in a studio apartment testbed over six months with 36 participants, testing both uncontrolled (US only) and controlled settings (US and UK). In the uncontrolled setup, participants accessed devices freely, while controlled settings restricted access times. A privacy issue emerged with a doorbell recording and sending videos without consent. Exposed data were classified as stored, sensor, and activity, with data-sharing parties identified as first-party vendors, support services, and third-party advertisers.

4.4 Anomaly Creation

This section discusses several techniques to create different types of anomalies within the smart home environment.

4.4.1 Building Malicious Apps: One method for creating anomalies is building malicious apps that mimic cyberattacks to test detection algorithms. Sikder et al. [48] developed five threat scenarios: impersonation, false data injection, side-channel attack, denial-of-service, and app-triggered attacks. For **impersonation**, an app leaked unlock codes via SMS or replayed a captured voice command to mimic the owner. **False data** injection apps altered sensor data or issued commands to devices. **Side-channel** apps either flickered lights without user presence or triggered a speaker to perform a potentially harmful action. **Denial-of-service** apps disabled smart home tasks, while **app-triggered attacks** changed device states to provoke malicious outcomes.

4.4.2 Using Built-in Tools: Creating controlled cyber anomalies in a testbed is essential to prevent unintended spread. Gimenez et al. [56] used the open-source **Mirage** (github.com/conchyliculture/mirage) framework to generate wireless attacks, including Zigbee scans and injections, Wi-Fi deauthentication and rogue APs, and **Bluetooth (BLE)** scans, each lasting two minutes across various locations. Dadkhah et al. [57] used the **Low Orbit Ion Cannon (LOIC)** for DDoS attacks over HTTP, UDP, and TCP, while Nmap (nmap.org) and Hydra (github.com/vanhauser-thc/thc-hydra) were employed for brute force RTSP attacks on camera URLs.

4.4.3 Changing Environmental Conditions: Anomalies in built environments can be introduced by altering environmental factors such as temperature, humidity, and physical obstructions, which significantly impact IoT device performance and reliability [58]. Kang et al. [59] demonstrated this by using a Mi Air Purifier 2S and Mi Sterilization Humidifier to inject anomalies through air mist, simulating a polluted environment that affected the purifier's operation.

Environmental interference, including signal loss due to physical barriers like walls or nearby devices, can cause unexpected device behavior, leading to false-positive or false-negative anomaly detections. Misclassification occurs when detection models are not trained for such environment-specific anomalies, often mistaking them for security breaches. Power fluctuations or network congestion can also mimic cyberattacks or device failures, complicating anomaly detection further.

Testbeds simulate environmental factors like temperature, humidity, and signal interference, as seen in Kang et al.'s work [59], though replicating complex real-world conditions remains challenging. Network emulators, such as CORE, help mimic interference and communication delays, providing insights into anomaly detection resilience.

A hybrid approach, combining physical and virtual testbeds, enhances realistic replication of environmental conditions. Physical setups with devices like smart lights or thermostats, alongside network emulation, can improve the robustness of anomaly detection models by accounting for dynamic environmental variations. Advanced simulation tools are essential to emulate complex real-world factors, including multiple interference sources, signal degradation, and fluctuating environmental conditions.

4.4.4 Simulation: Simulation, though sometimes less realistic, is a budget-friendly option for generating anomalies. Said et al. [60] designed an intrusion detection model for IoT systems in smart hospitals using the Contiki Cooja simulator (github.com/contiki-ng), with settings for nodes, topology, and transmission rates, achieving 93.4% accuracy for rank attacks but only 60.8% for flood attacks. Similarly, Mirkovic et al. [61] used the NS-2 simulator and the DETER testbed [62] for DoS attack evaluation, finding that the testbed handled significantly higher throughput. Lundström et al. [63] created a Markov-based anomaly detection model for smart homes, trained on simulated normal data, and achieved 85% accuracy. However, as Zuo et al. [64] emphasize, the rapid expansion of network traffic and its complex nature pose challenges for traditional ML methods, making it difficult to accurately capture traffic characteristics when working with large datasets.

Table 7. Anomaly Detection Mechanisms

Paper	Anomaly Detection Mechanism	Testbed	Dataset	CA	PA	DF
Sikder et al. [48]	Markov Chain–based machine learning model	●	○	○	●	●
Ullah & Mahmoud [65]	FFN	○	●	●	○	○
Kang et al. [59]	(Not mentioned in the paper)	●	○	○	●	●
Hosseini et al. [66]	Gaussian-based Kernel Density Estimation	●	○	○	●	●
Li et al. [67]	LSTM Autoencoder	○	●	●	○	○
Liu et al. [68]	AMCNN-LSTM (Attention Mechanism-Based CNN-LSTM Model)	○	●	●	○	○
Mudgerikar et al. [69]	Random Forest	○	●	●	○	○
Djenouri et al. [70]	The genetic algorithm and the bee swarm optimization	○	●	●	○	○
Mothukuri et al. [71]	LSTM & GRU	○	●	●	●	●
Shafi et al. [72]	E3ML & MLP &RNN &ADT	○	●	●	○	○
Yahyaoui et al. [73]	OCSVM & DL	●	●	●	●	○
Gimenez et al. [56]	Autoencoder neural network & temporal and spatial diagnosis	●	○	●	○	●
Heartfield et al. [4]	Isolation Forest	●	○	●	○	○
Teh et al. [74]	RFE & RFR for feature extraction and selection &PCA for AD	○	●	○	●	○
Sater & Hamza [75]	Three LSTM layers	○	●	●	○	○
Sarwar et al. [76]	Adaboost	○	●	●	○	○
Dissem et al. [77]	RLNAS & Autoencoder	○	●	○	●	○
Xiao et al. [78]	Autoencoder with (LDMS & TTPE & NWRL)	●	●	○	●	●
Lie et al. [79]	Knowledge graph-based bimodal	●	○	●	○	○

○ No, it does not discuss this topic, ● Yes, it does discuss this topic. "**CA**"=Cyber Anomaly. "**PA**"=Physical Anomaly. "**DF**"=Device Fault.

This highlights the importance of high-fidelity simulations to ensure that network anomalies are realistically represented, enhancing the performance of anomaly detection systems.

4.5 Scalability and Adaptability

Scalability and adaptability are essential for testbeds, especially as IoT evolves and anomalies grow more complex. Scalability depends on infrastructure type, communication protocols, and device diversity. Physical testbeds are less scalable due to financial and logistical demands for adding devices, while simulated or emulated testbeds, like CORE, scale more easily by expanding virtual nodes and handling high network traffic efficiently. Adaptability requires that testbeds integrate new devices and protocols, often achieved with modular frameworks where components like communication layers can be added or replaced without significantly changing the entire setup. Technologies like SDN and virtualization also support real-time updates, enabling adaptability. To facilitate these needs, a plug-in framework and layered design allow seamless updates to individual layers, reducing modification complexity. New devices or protocols can be tested via simulation or emulation before physical integration, simplifying the process.

4.6 New Technologies Integration

The complexity of IoT networks is advancing rapidly with the emergence of technologies like 5G and edge computing, making it essential to develop testbeds that incorporate these advancements. Such integration not only enhances real-time processing but also creates a more realistic environment for evaluating AD techniques. To address resource constraints in edge environments, testbeds should include devices like microcontrollers to evaluate AD performance under limited resources, ensuring AD techniques are lightweight and efficient. The testbed should also support decentralized learning, such as FL, allowing local ML model training on edge devices to preserve data privacy while sharing only model updates. Real-time responses are essential; the testbed should isolate compromised devices upon anomaly detection. With the emergence of 5G, which supports numerous IoT devices with low latency, testbeds must simulate high-density 5G networks to assess AD scalability and accuracy. Furthermore, to reflect 5G's low-latency environment, AD capabilities should be distributed across the network, and the testbed should simulate large-scale IoT networks

Table 8. Strengths, Weaknesses, and Suitability of Anomaly Detection Mechanisms

Anomaly Detection Mechanism	**Strength**	**Weakness**	**Suitability**
Markov Chain–based machine learning model	Effective for time-series data	Not suitable for non-linear relationship	Detecting simple point anomalies
FFN	Handles structured and unstructured data	Computationally intensive	Detecting complex anomalies that have labeled data
Gaussian-based Kernel Density Estimation	Effective in anomaly detection for continuous data	Produce false positives in changing environments	continuous datasets
LSTM Autoencoder	Captures long-term dependencies in the data	Requires extensive training	Detecting anomalies that developed gradually
AMCNN-LSTM (Attention Mechanism-Based CNN-LSTM Model)	Ability to extract features and perform temporal analysis	Computationally intensive	Detecting sophisticated attacks
Random Forest	Handles high-dimensional data well	Not suitable for time-series data	Detecting point anomalies and knows attacks
The genetic algorithm and the bee swarm optimization	Works well in complex environments	Computationally intensive	Works well when feature selection and optimization are important
LSTM & GRU	Works well for sequence prediction	Computationally intensive	Detecting anomalies in time-series data
E3ML & MLP & RNN & ADT	Using combination of ML models improve accuracy	Computationally intensive	Detecting anomalies in complex environments
OCSVM & DL	OCSVM works well for unsupervised AD & DL works well for high-dimensional data	OCSVM is sensitive to anomalies & DL requires large training data	Detecting novel anomalies
Autoencoder neural network & temporal and spatial diagnosis	Works well for unsupervised AD & it detects complex anomalies due to temporal and spatial diagnoses	Requires extensive training data	Detecting complex anomalies
Isolation Forest	Resource efficient & Scales well	Not suitable for sophisticated anomalies	Detecting simple anomalies
PCA	Works well for high-dimensional data	Not suitable for non-linear data	Detecting point anomalies
Three LSTM layers	Having three-layers allows LSTM to detect complex anomalies in time-series data	Computationally intensive	Detecting real-time sophisticated anomalies
Adaboost	Effective in combining weak classifiers to form a strong classifier	Sensitive to noisy data and outliers	Suitable high-dimensional datasets and unusual data patterns
RLNAS & Autorncoder	Capable of automatic feature learning and neural architecture search	High computational cost during training	Effective in discovering optimal neural architectures for anomaly detection
Autoencoder with (LDMS & TTPE & NWRL)	Integrates multiple techniques for enhanced performance	Complexity in implementation and tuning	Suitable for complex anomaly detection tasks requiring multiple methodologies
Knowledge graph-based bimodal	Leverages structured knowledge for improved context understanding	Limited by the quality and completeness of the knowledge graph	Effective in scenarios where relational context is crucial

and network slicing to evaluate AD effectiveness in isolated network segments. In summary, while this discussion has centered on edge computing and 5G networks, these are just two examples of emerging IoT technologies. As the IoT landscape continues to evolve, AD techniques must be designed to adapt effectively to the challenges posed by these advancements.

4.7 Ethical and Privacy Considerations

The use of testbeds is highly dependent on ethical and privacy considerations because data from testbeds are close to real life data. Studies should be more transparent on how they use the data, and who has access to it, and what are the measurements that the data has taken to secure sensitive data. Sikder et al. [48] implemented an approach to confidentiality by assigning anonymous IDs to participants, ensuring that personal information remained protected. Another method to enhance privacy involves **Differential Privacy (DP)**, which adds noise to the dataset, preserving the statistical integrity of the data while safeguarding user privacy. However, DP can introduce challenges, as the added noise may impact the accuracy of anomaly detection algorithms, requiring careful evaluation of its effects [80]. FL also offers a privacy-focused framework by ensuring data remains locally while only model updates are shared with a central node. Zhang et al. [81] presented FedGroup, an FL approach that aggregates updates from groups of IoT devices, maintaining individual data privacy throughout the process. Finally, obtaining informed consent from participants is essential; clear communication about data usage further reinforces trust and ethical standards in testbed development.

4.8 Challenges and Mitigation

Developing a comprehensive real-world testbed presents significant challenges, particularly due to the financial requirements for infrastructure, heterogeneous IoT devices, and ongoing maintenance. One way to overcome these challenges is through the use of virtualization and emulation, which enable the simulation of real-world devices, network configurations, and environments without the need for physical deployment. For example, Emulab is an emulator that can simulate both IoT and network devices, offering a cost-effective alternative to physical testbed development. By leveraging such tools, the overall cost of testbed creation can be significantly reduced. In addition, employing **commercial off-the-shelf (COTS)** devices instead of more sophisticated and expensive IoT hardware can help lower the financial burden. Low-cost alternatives like Raspberry Pi or microcontrollers can serve as substitutes for various smart home devices, offering functionality at a fraction of the cost. Another cost-saving strategy is the use of collaborative testbed environments, where organizations or universities share testbed resources. For instance, platforms like FIT IoT-LAB allow shared access to testbed infrastructure, reducing individual development costs and maximizing resource utilization.

Integrating heterogeneous devices into a testbed poses another critical challenge. IoT devices are often built by different manufacturers and use various communication protocols, making it difficult to seamlessly integrate them into a unified testbed environment. Middleware platforms such as Home Assistant and OpenHAB can help bridge this gap by facilitating the integration of devices that use different communication protocols, allowing them to be managed and controlled from a single platform. Moreover, using devices that comply with common standard communication protocols can further simplify integration efforts, reducing compatibility issues and mitigating the complexities associated with IoT heterogeneity.

Evaluating testbed characteristics is key to building reliable anomaly detection frameworks for built environments like smart homes. These testbeds simulate real-world conditions, enabling us to assess how well systems detect unusual behaviors. However, achieving realistic simulations within testbeds is challenging, as they often lack the complexity of real environments, where factors like device malfunctions, human errors, and network congestion occur. Hybrid testbeds, which combine physical IoT devices with simulated environments, can address this by better replicating these scenarios. Device diversity is another challenge, as real environments contain a range of devices from different manufacturers with various communication protocols. Focusing on widely-used devices can help address this. Testbeds also miss long-term environmental effects like seasonal changes or device wear, which impact anomaly detection. Extending data collection periods and simulating environmental factors can help reveal issues that emerge over time. Finally, human interaction varies across ages, routines, and emergency responses, making short-term testbeds inadequate for capturing real behavior. Long-term studies that gather data on human interaction could bridge this gap and enhance testbed realism.

Table 9. Overview of Anomaly Detection in Built Environments, Including Smart Homes, Smart Buildings, Smart Cities, and Industrial IoT. The Table Summarizes Anomaly Types, Detection Techniques, Testbed Characteristics, Evaluation Metrics, and Dataset Creation Challenges Across Different Applications

Application	**Anomaly Types**	**Detection Techniques Examples**	**Testbed Characteristics**	**Evaluation Metrics**	**Dataset Creation Challenges**
Smart Home	Unauthorized access, device malfunction, abnormal energy consumption, privacy breaches	Markov Chain models (state-based anomalies), DL (complex patterns), Ensemble methods (robustness)	High device diversity, realistic network topology, user behavior modeling, environmental control (temperature, lighting)	Recall (critical for security), Precision (minimize false alarms), F1-score (balanced performance)	Privacy-preserving techniques, realistic user behavior profiles, diverse attack scenarios
Smart Building	HVAC system failures, security breaches, occupancy anomalies, water leaks, fire detection	DL (complex systems), Hybrid approaches (adaptability), Statistical methods (energy consumption)	Scalable network topology, sensor diversity (temperature, humidity, CO2), energy consumption monitoring, physical security	Accuracy (overall performance), Precision (minimize false positives), F1-score (balanced)	Long-term monitoring (seasonal variations), diverse occupancy patterns, integration of building management systems (BMS) data
Smart City	Traffic congestion, water leaks, air quality anomalies, infrastructure failures, cyberattacks	Traffic flow prediction models, anomaly detection in sensor networks, DL (urban data analysis)	Large-scale network simulation, sensor network emulation, traffic flow modeling, environmental monitoring (air quality)	Balanced accuracy (imbalanced data), ROC AUC (overall performance), Runtime (real-time)	Real-world traffic data integration, sensor data fusion, simulation of urban dynamics, diverse environmental conditions
Industrial IoT	Equipment failures, process deviations, safety violations, cyber-physical attacks	Statistical process control, ML-based fault detection, DL (predictive maintenance)	Real-time data acquisition, industrial protocol support (e.g., Modbus), sensor integration (vibration, temperature), fault injection	Precision (minimize false alarms), Recall (capture critical failures), MAE, MSE, RMSE (prediction)	Integration of SCADA systems data, fault injection, simulation of industrial processes, long-term equipment performance data

5 ANOMALY DETECTION MECHANISM

In this section, we discuss several anomaly detection mechanisms applied to testbeds and various datasets, focusing on the anomaly detection algorithms used, dataset choices, devices employed in testbeds, types of anomalies detected, and evaluation metrics. A summary of anomaly detection mechanisms is presented in Table 7, and a comparison between the presented mechanisms based on their strengths, weaknesses, and suitability is presented in Table 8. An overview of anomaly detection in built environments across different applications is provided in Table 9.

5.1 Testbed-based Anomaly Detection

Sikder et al. [48] presented Aegis, a context-aware security framework for (SHSs) that detects malicious behavior by analyzing device interactions and user activities. The framework contributes by leveraging contextual models to understand sensor-device co-dependencies and user activity correlations, enhancing anomaly detection in SHSs. Its novelty lies in its Markov Chain-based approach, which captures sensor-device interactions and user behavior patterns to identify security threats while adapting to different home layouts with minimal overhead. Evaluated on a real-world smart home testbed with various devices and attack scenarios, Aegis achieved high detection accuracy, demonstrating its effectiveness in securing diverse smart home environments.

Yahyaoui et al. [73] introduced READ-IoT, a reliable event and anomaly detection framework for IoT systems that integrates **event detection (EDS)** and **anomaly detection (ADS)** into a unified system for improved efficiency. The framework contributes by combining rule-based and ML-based

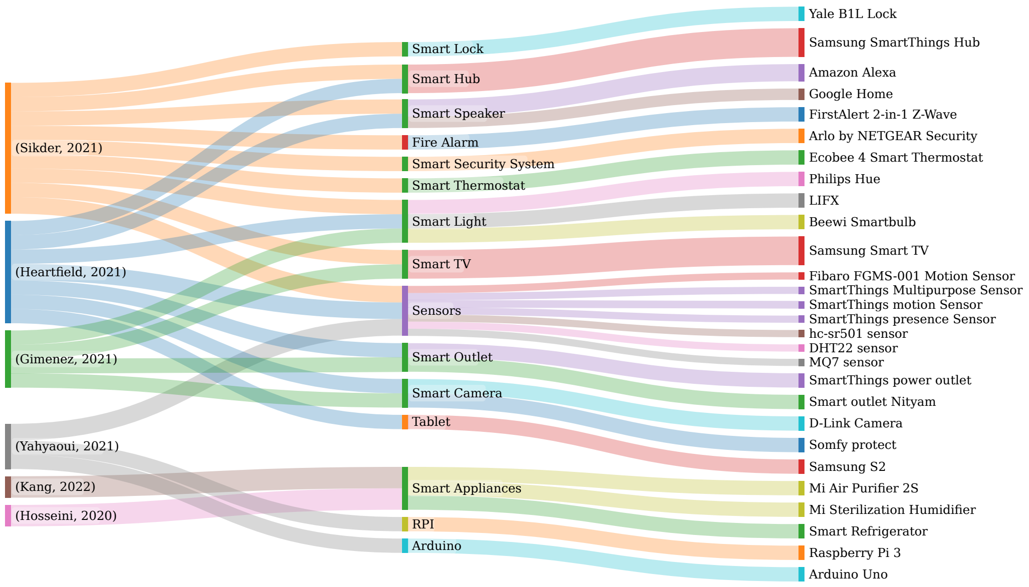

Fig. 3. A Sankey diagram showing IoT devices and sensors used in selected testbed studies. Devices such as smart locks, smart thermostats, and security cameras are mapped to their corresponding research sources. Common brands include SmartThings, Philips Hue, and Raspberry Pi.

detection within a hybrid cloud/fog deployment strategy, ensuring real-time responsiveness and reliability. Its novelty lies in its cascaded detection approach, where rule-based filtering precedes ML-based analysis, and a reputation-aware deployment mechanism enhances detection accuracy while addressing real-time constraints. Evaluated on real and simulated IoT environments, READ-IoT achieved over 90% accuracy with OCSVM and 98% with DL, demonstrating low latency and high reliability in detecting fire incidents and unauthorized access.

Kang et al. [59] proposed SmartHomeDetector, a hierarchical automata-based framework for detecting anomalies in IoT devices within smart homes. The framework consists of a model construction phase and a runtime anomaly detection phase, contributing by providing a structured approach to anomaly detection through automated behavior modeling. Its novelty lies in constructing a hierarchical automata-based behavior model using automatic testing, capturing both external user operations and internal device transformations, along with runtime metrics. By employing a comprehensive detection technique, SmartHomeDetector identifies various anomaly types based on deviations from this model. Evaluated on a smart home testbed, the approach effectively detected abrupt, accumulated, functional component, and human-induced anomalies, demonstrating its potential despite being tested on a limited number of devices.

Hosseini et al. [66] presented an on-line anomaly detection approach for household appliances, specifically standard and smart refrigerators, using energy consumption data to model normal operation and detect anomalies. The study contributes by offering insights into managing anomaly detection and diagnosis phases, crucial for correctly distinguishing faulty from abnormal behaviors. Its novelty lies in its appliance-level approach, leveraging sub-metered data to develop a low-intrusion, low-resolution, and efficient anomaly detection method, distinguishing it from aggregate-level techniques. Additionally, it introduces the concept of diagnosis decision, assessing operational conditions to differentiate faults from abnormal usage. Using a Gaussian-based kernel density estimation algorithm, the approach achieved 99% accuracy for smart refrigerators and 98% for standard refrigerators, demonstrating its practicality for real-world implementation.

Xiao et al. [78] contributed SmartGuard, an autoencoder-based anomaly detection framework for smart homes. SmartGuard addresses challenges in learning infrequent behaviors, temporal context, and noise handling. Its novelty lies in the integration of a **Loss-guided Dynamic Mask Strategy (LDMS)**, a **Three-level Time-aware Position Embedding (TTPE)**, and a **Noise-aware Weighted Reconstruction Loss (NWRL)** to enhance anomaly detection by improving learning efficiency, capturing temporal anomalies, and reducing misclassification. SmartGuard was evaluated using three datasets: a custom AN dataset collected through a smart home testbed with 36 devices and three volunteers living in an apartment for over two weeks, and two public datasets, FR and SP. Abnormal behaviors were artificially injected into normal sequences for testing. This comprehensive evaluation demonstrated SmartGuard's ability to detect ten types of anomalies, such as light flickering and unauthorized camera deactivation, outperforming methods like LOF, Isolation Forest, and TransformerAutoencoder in recall, precision, and F1-score, particularly for time-sensitive anomalies.

Li et al. [79] introduced SeIoT, a knowledge graph-based bimodal anomaly detection framework for smart homes that enhances IoT security by integrating interaction-related and time-related semantics extracted from network traffic. The framework contributes by leveraging a knowledge graph to represent smart home semantics and an FS-HAN-based anomaly detection mechanism to identify anomalous behaviors. Its novelty lies in using FS-HAN to model complex device-environment interactions, enabling the detection of both device-targeted and platform-based attacks solely through network traffic sniffing. Evaluated in a real-world testbed, SeIoT outperformed traditional NADSes, achieving a 97.7% **true positive rate (TPR)** and a 99.0% **true negative rate (TNR)**, demonstrating its effectiveness in securing smart home environments. A summary of the testbed devices is provided in Figure 3.

5.2 Dataset-based Anomaly Detection

Li et al. [67] presented ADRIoT, an edge-assisted anomaly detection framework for IoT networks, featuring an edge-based anomaly detection module with device-specific LSTM autoencoders and a multi-edge collaboration mechanism for resource sharing. The framework contributes by providing Detection-as-a-Service through a cloud-edge architecture, enhancing anomaly detection efficiency. Its novelty lies in leveraging unsupervised LSTM autoencoders for zero-day attack detection on individual IoT devices and integrating SSDP-based multi-edge collaboration for efficient resource utilization. Evaluated on a real-world IoT traffic dataset, ADRIoT outperformed other unsupervised anomaly detection methods, demonstrating its effectiveness in securing diverse IoT environments.

Mudgerikar et al. [82] introduced E-Spion, an edge-based **intrusion detection system (IDS)** designed for IoT devices, leveraging system-level profiling through process parameters and system calls to detect anomalies. The system contributes by integrating a device-edge split architecture and a three-layer anomaly detection engine, consisting of process whitelisting, process behavior monitoring, and system call behavior monitoring. Its novelty lies in its ability to detect both traditional IoT malware and sophisticated file-less attacks with minimal overhead. Evaluated on malware datasets, E-Spion demonstrated high detection accuracy, with Random Forest outperforming other classifiers in intrusion detection efficiency.

Ullah & Mahmoud [65] proposed a framework utilizing **conditional Generative Adversarial Networks (cGANs)** to address data imbalance and enhance anomaly detection in IoT networks. Their approach introduces **one-class cGAN (ocGAN)** and **binary-class cGAN (bcGAN)** models to generate synthetic data for minority classes, improving detection across various datasets. This study contributes by integrating cGAN-based data augmentation techniques for both normal and malicious profiles and introduces a novel multiclass classification approach using bcGAN. Evaluated across multiple IoT network datasets, the framework demonstrated superior performance, achieving high detection rates and outperforming traditional anomaly detection models and GAN-based approaches.

Mothukuri et al. [71] proposed an FL-based anomaly detection approach for intrusion detection in IoT networks, enabling GRU models to be trained on-device while preserving data privacy. Their approach integrates federated training rounds, where only model weights are shared instead of raw data, and an ensembler that aggregates updates to optimize a global model. This study contributes by leveraging FL to enhance intrusion detection without centralized data collection and introduces a novel FL-GRU framework that outperforms traditional centralized ML, achieving higher accuracy (90.255%) with fewer training epochs on a Modbus-based attack dataset [83].

Sarwar et al. [76] proposed a ML-based anomaly detection approach for securing smart home IoT networks. Using the UNSW BoT-IoT dataset, they evaluated six classifiers and demonstrated that **Random Forest (RF)**, **Decision Tree (DT)**, and AdaBoost achieved the highest accuracy (100% on training data, 99.9% on test data), outperforming ANN, LSTM, and Autoencoders. This study contributes by comparing multiple classifiers and highlighting the effectiveness of feature selection techniques in improving anomaly detection accuracy. Its novelty lies in demonstrating the robustness of tree-based models on a larger IoT dataset, offering a more scalable and effective approach than previous methods.

Dissem et al. [77] proposed an automated time-series anomaly detection framework for smart buildings, leveraging RL to optimize autoencoder architecture. Their **RL-based neural architecture search (RLNAS)** enables the automatic selection of optimal models without prior knowledge, outperforming random search. This study contributes by introducing an RL-driven approach for anomaly detection and demonstrates its novelty by achieving higher F1 scores and accuracy on smart building sensor data, particularly for temperature, humidity, and power consumption, while addressing the challenge of high-variability features like illuminance. Figure 4 presents the datasets and research papers discussed in Section 5.2.

6 DATASETS

Having IoT devices in a controlled environment facilitates monitoring and observing how devices communicate and behave under different test scenarios. In this section, we discuss different infrastructure setups, data creation, data collection, capture duration, and methods of dataset analysis.

6.1 Infrastructure Setup

When designing an infrastructure for a dataset, one must consider if the dataset infrastructure will be implemented physically or virtually. For example, Vigoya et al. [84] implemented a virtual infrastructure by building four client nodes, each having four processes where each process represents a temperature sensor. Each sensor (process) has an MQTT client, which sends its temperature data to a central MQTT broker using the MQTT publisher–subscriber approach. Similarly, Koroniotis et al. [85] also implemented a virtual infrastructure that follows the MQTT publisher–subscriber approach, which is implemented over TCP/IP using Node-Red to simulate IoT devices.

In contrast, Dadkhah et al. [57] followed another approach in terms of infrastructure setup for their dataset where they built a physical infrastructure using two network interface cards: one is connected to the gateway while the other is connected to a NETGEAR GS308 switch, which all IoT devices that require Ethernet are connected to it. Vera Plus smart hub is connected to the switch, acting as an Internet gateway for Wi-Fi IoT devices. Furthermore, Philips Hue Bridge, SmartThings Hub, and Fibaro Home Center Lite have been used to facilitate the communication of Zigbee and

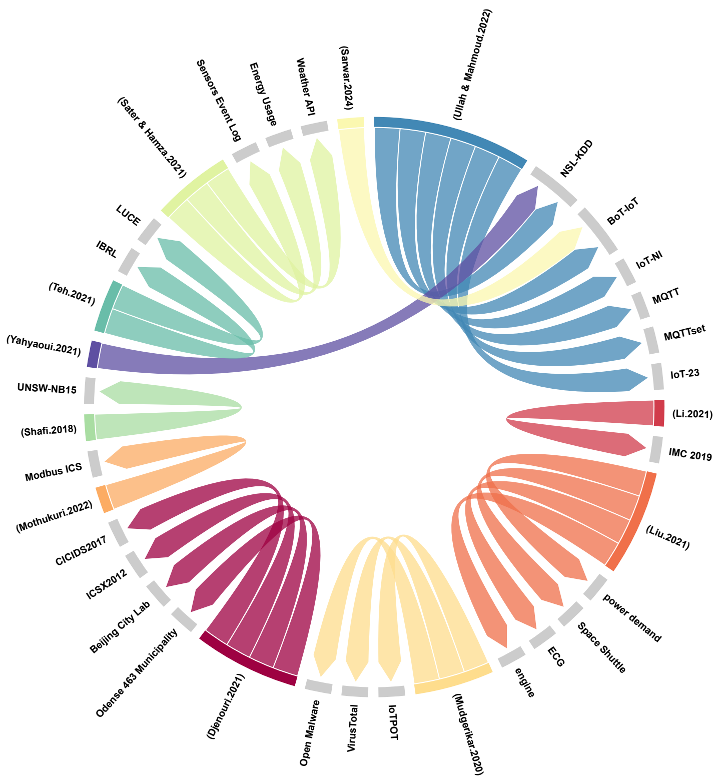

Fig. 4. A circular chord diagram linking research papers to the datasets they utilized. Each paper is positioned around the circle, with colored arcs connecting it to corresponding datasets.

Z-Wave devices and sensors. These two examples of infrastructure show how different dataset creation can be in terms of infrastructure. Additionally, it illustrates the possibility of creating a data structure with limited resources.

6.2 Network Configuration

Different IoT device testing dictates different network configurations. Some devices only communicate through Zigbee protocol, and others communicate through Wi-Fi and voice commands. This heterogeneity directly contributes to the choice of network configurations. Even though having different wireless communication technologies is beneficial in terms of including a different range of IoT devices, it mostly comes with the burden of having a hub for each wireless communication technology, such as Zigbee. Moreover, some devices within the same wireless communication technology require different hubs, as does Bosch Twinguard, which requires a Bosch smart home controller. Therefore, types of devices and their communication settings must be considered when building the network configuration of a dataset environment.

6.3 Data Collection

The goal of the dataset has an important role in selecting what type of IoT data to collect, as some datasets are only interested in cyberattacks, and others extend their interests to different types of generated data. According to Dadkhah et al. [57], ideally and to observe how IoT devices behave under different scenarios, six statuses must be taken into consideration:

Power: During the power state, all devices must be unplugged from the network, then the network is rebooted. After that, each device is connected to the network on an individual basis, and the MAC address associated with each device is noted, and the network traffic for each device is captured individually as well. Moreover, the capture process should continue until no remaining packet from that device is received. This process would provide an idea of how each device behaves individually.

Idle: During the idle state, an assumption that devices will not communicate with each other during the idle stage must be established. As a result, this stage should usually take place during the night or during holidays. This process would help to identify if a device is making unwanted communication. Ideally, the data captured in the idle stage should match the data captured in the power state for each device.

Interactions: During the interactive state, each device's communication with other devices must be captured. It is also important to mention that some devices have more than one way of communication. For instance, a Philips Hue smart light can be turned on and off through voice command using a voice assistant device (such as Echo Dot) and the Philips Hue app. Thus, different control conditions must be considered as well.

Scenarios: During this state, the scenario intended for the dataset is performed, and network traffic is captured.

Active: During this stage. all IoT device network traffic is captured, including communications between devices, when there are people interacting with the devices either actively or passively.

Attacks: During attacks, a series of cyberattacks are performed on the devices, and the network is captured during attacks. Several tools can be used to generate cyberattacks, which we discussed in 4.4.2 Section.

Network traffic capture can be achieved using one of the network sniffer tools such as Wireshark and Tcpdump. However, some protocols might require additional hardware or software to be captured, such as the Zigbee protocol.

6.4 Data Creation

Data creation must start once the dataset environment is set up and configured with the required tools. During data creation, several scenarios are generated and followed to achieve the dataset's goal, whether for a smart home dataset or otherwise. For instance, Dadkhah et al. [57] generated six different scenarios that cover several activities.

Coming home: The initial setup for this scenario is that the Ring alarm is armed, and devices are off because no one is home. Then, once a participant enters the lab, the Ring alarm, the Netatmo camera, and ArloQ motion detection are alerted. The participant enters the passcode to disarm Ring and voice commands Google Nest Mini to turn on the Philips Hue lights and Globe lamp. Similarly, the participant voice-commands Alexa to turn on the Atomi Coffee Maker and to play music over the Sonos speaker. Finally, the participant starts the Roomba vacuum cleaner using the associated mobile app to start the cleaning process.

Leaving home: The initial setup for this scenario is that all devices are turned on to simulate the presence of people in the home. Then, the participant voice-commands Alexa to turn off the Atomi Coffee Maker and stop the music that was playing over the Sonos speaker. Similarly, voice commands Google Nest Mini to turn off the Philips Hue lights and Globe lamp. Finally, the Roomba vacuum cleaner to its home base to stop the cleaning process.

Home intrusion (during the day): The participant enters the lab; Ring alert, Netatmo camera, ArloQ, and HeimVision camera motion detection are triggered. The D-Link Water Sensor is also

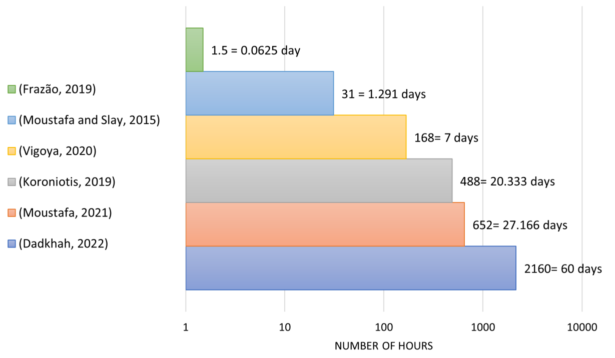

Fig. 5. A horizontal bar chart comparing the data capture duration of six datasets. The x-axis is showing hours, while dataset names and corresponding durations (e.g., 60 days, 27.166 days) are labeled within each bar.

triggered to indicate that the intruder has spilled some sort of liquid. Once the Ring alarm goes off, the participant leaves the lab, which will also trigger Netatmo and ArloQ motion detection. This attack scenario occurs with lights on.

Home intrusion (during the night): The participant enters the lab; Ring alert and Netatmo camera are triggered. ArloQ and HeimVision camera sound detection are triggered. The D-Link Water Sensor is also triggered to indicate that the intruder has spilled some sort of liquid. Once the Ring alarm goes off, the participant leaves the lab, which will also trigger Netatmo and ArloQ motion detection. This attack scenario occurs with lights off to evaluate the night vision capability of the Netatmo camera and the sound detection capability of ArloQ and HeimVision because they do not support night vision.

Owner error: This scenario is to simulate homeowner errors like entering the wrong password for a security lock. The participant enters the lab, which triggers a Netatmo camera and the ArloQ motion detection. Then, the participant enters an incorrect password for the Ring alarm thrice every 30 seconds, setting the alarm off.

IoT vs Non-IoT: This scenario is to test how well IoT devices integrate with Non-IoT devices. To do so, a participant asks Alexa to play music over a Sonos speaker, let LG TV play a YouTube video, surf social media using a phone, browse the Amazon website from a computer, and ask Google to turn the Philips Hue lights on.

Another approach to generating synthetic data is using simulation. Moustafa and Slay [39] used a network traffic simulator called IXIA PerfectStorm, which simulates different types of cyberattacks. Furthermore, Vigoya et al. [84] discussed three ways of generating anomalous traffic: interception, which could be achieved by deleting sent traffic packets; modification, which could be achieved by changing sensor readings that are sent randomly (for instance, Vigoya et al. [84] have changed temperature samples sent to the MQTT broker); and duplication, which could be achieved by sending more data than planned in the first place (for instance, Vigoya et al. [84] sent more tokens than planned).

6.5 Data Capture Duration

Data capture duration differs from one dataset to another. In this section, we try to estimate the minimum requirement for dataset capture duration depending on well-known different datasets that were published. (Dadkhah, 2022) [57] stated that the experiments conducted on the IoT devices

Table 10. Formulas of Statistical-based Methods

Statistical-based Methods	Formula	No.
Low variance	$p(1-p)$	(1)
T-score	$\lvert\mu_1-\mu_2\rvert/\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}$	(2)
Chi-square score	$\sum_{j=1}^{r}\sum_{s=1}^{c}\frac{(n_{js}-\mu_{js})^2}{\mu_{js}}$	(3)
Gini index	$\underset{\mathcal{W}}{min}\left(p(\mathcal{W})(1-\sum_{s=1}^{c}p(C_s\mid\mathcal{W})^2)+p(\overline{\mathcal{W}})(1-\sum_{s=1}^{c}p(C_s\mid\overline{\mathcal{W}})^2)\right)$	(4)
CFS	$CFS\_score(\mathcal{S})=\frac{k\overline{r_{cf}}}{\sqrt{k+k(k-1)\overline{r_{ff}}}}$	(5)

took 3 months to create the CIC IoT 2022 dataset (www.unb.ca/cic/datasets/iotdatas-et-2022.html). He specifically stated that he captured 30 days of idle network traffic and 30 days of active network traffic (see section 6.3 for more illustrations about different statuses). Furthermore, (Moustafa, 2021) [86] created an IoT dataset called TON IoT (research.unsw.edu.au/projects/toniot-datasets) that has a data capture of approximately 27 days. Even though (Moustafa, 2021) [86] did not specify the exact capture duration, we were able to estimate a data capture duration by calculating the difference between the first and second capture timestamps. Similarly, (Koroniotis, 2019) [85] published Bot-IoT (research.unsw.edu.au/projects/bot-iot-dataset) dataset that has a capture duration of approximately 20 days, where we investigated the first and last capture timestamps to estimate the capture duration.

(Vigoya, 2020) [84] created a dataset (DAD) (github.com/dad-repository/dad) that captured network traffic for 7 days. During those 7 days, different types of anomalies were generated within 5 days, which leaves 2 days with normal network activity. Additionally, each day of the capture duration has a summary of the attack performed, if any; otherwise, it will state that no attack has taken place.

On a shorter duration period, (Moustafa and Slay, 2015) [39] captured traffic for 2 days, where they simulated traffic, using IXIA PerfectStorm, for 16 hours on the first day and 15 hours on the second day, which generated 100 GB of traffic in total for their UNSW-NB15 dataset (research.unsw.edu.au/projects/unsw-nb15-dataset). (Frazão, 2019) [83] also captured network traffic for a short period. His experiment captured network traffic for 1 hour and 30 minutes with 1, 5, and 15 minutes of attack times. The reason behind the varying capture duration is that he wanted to see how each ML classifier would behave with different capture duration. Figure 5 presents a summary of capture duration.

6.6 Data Analysis

After creating the dataset, the data captured must be analyzed so that the dataset can be used in different applications that are tailored to user needs. In this section, we discuss two techniques of dataset analysis.

6.6.1 Feature Analysis. Analyzing network features helps in giving insight into the traffic pattern and whether the traffic is malicious or not. Tavallaee et al. [87] analyzed the well-known KDD CUP 99 dataset features into three categories:

Basic feature: This category has all fields that are extracted from the TCP/IP connection, which has information such as IP and MAC addresses.

Traffic feature: A feature computed on the basis of window intervals is included in this category, which can be further divided into two subcategories: same-host features, which measure protocol behavior, service, and other statistics about connections with the same destination host as the current connection over the past 2 seconds, and same-service features, which check only connections that have been connected to the same server in the past 2 seconds.

Content feature: For cyberattacks, R2L and U2R attacks do not have frequent sequential intrusion patterns like DoS and probing attacks. Because DoS and probe attacks involve many connections in a short period, R2L and U2R attacks typically involve only one connection. Thus, more in-depth features are needed to detect these kinds of attacks, e.g., the number of unsuccessful login attempts, which is in the payload part of the packet. Such features are referred to as content features.

Biglar Beigi et al. [88] also discussed distinct flow-based features thoroughly:

- **Source and destination IP addresses:** One of the primary advantages of their use is the possibility of quantifying the number of distinct connections. However, it does not provide a definitive conclusion as a feature.
- **Source and destination port:** Botnet traffic was often recognized by source and destination ports, but this is not the case anymore due to the frequent change of source and destination ports.
- **Protocol:** It reduced the volume of flows that needed to be processed by filtering out non-related traffic. Furthermore, a certain protocol might be an indication of malicious traffic, such as Internet Relay Chat (IRC), because it is rarely used for non-malicious purposes.
- **Duration:** This feature could help identify malicious traffic. For example, most botnets initiate a single-way connection and maintain very short communication sessions before attempting a multidirectional connection.
- **First packet length:** In addition to revealing the underlying protocol's characteristics, the first packet in the flow is useful in detecting malicious traffic.
- **Flow size features:** Communication patterns are mainly intended to be represented by such features.
- **Ratio between the number of incoming packets over the number of outgoing packets:** It has been noted in studies examining the breakdown of Internet traffic that there is an even distribution between inbound and outbound traffic for various protocols. This would help identify further traffic characterization [89].
- **Packet exchange:** The number of network packets exchanged between two entities could help to identify malicious traffic. For instance, bots are always trying to keep communication alive by sending many packets.
- **Reconnect:** Some malicious entities might try to disconnect and reconnect again to break the detection process using feature extraction. To overcome this problem, a certain number of reconnections are only allowed within a certain interval of time.
- **Number and percentage of small or null packets exchanged:** Studies have shown that certain malicious traffic might exchange small network packets. For instance, IRC hosts exchange small messages with the C&C server [90][91].
- **Average packet or bits per second, average inter-arrival time of packets:** This feature helps identify similar communication patterns.

Different network features are useful for certain applications. We have seen some useful network features in the context of malicious traffic. In the context of IoT device classification, Sivanathan et al. [92] extracted DNS queries, NTP queries, port numbers, and cipher suits for each device within the testbed they developed and used these features to identify the devices in the network. This shows how different features can be used for different applications. As a final thought, extracting more features does not always guarantee a high accuracy anomaly detection rate, according to Revathi

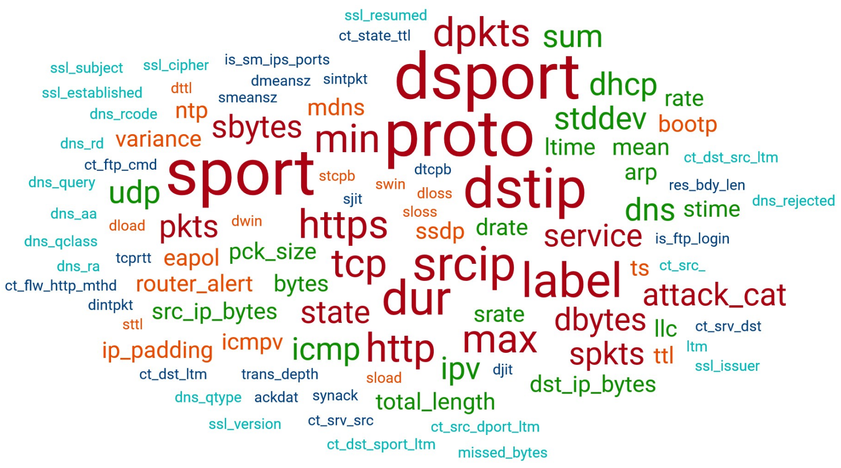

Fig. 6. A word cloud visualizing 207 extracted features from 10 datasets. Larger words indicate more frequently extracted features (e.g., ’dsport,’ ’proto,’ ’sport’), while smaller words represent less common features.

& Malathi [93]. In their paper, they applied RF, naive Bayes, and SVM, among other classifiers, one with 41 features and another with only 13 features, where they used a **correlation feature selection (CFS)** subset to filter down the features. The result was that some classifiers had a high accuracy detection rate with fewer features. For instance, the RF classifier had an accuracy of 98.9% with only 13 features, whereas it achieved an accuracy of 97.6% with 41 features in detecting probe attacks. Figure 6 shows the most common features extracted from the following datasets: UNSW-NB15 [39], CIC IoT 2022 [57], Bot-IoT [85], MQTT-IoT-IDS2020 [94], DAD [84], TON_IoT [86], CIC IoT 2023 [95], IoT-deNAT [96], IOT Sentinel [97], IoTSense [98].

6.6.2 Statistical Data Analysis. An efficient way to predict and understand IoT device behavior is through mathematical modeling. Using proven approach techniques, it is essential to mathematically model the real environment to generate a dataset as realistic as possible [84]. The benefit of using a statistical-based analysis method is that it analyzes features individually, which eliminates the feature redundancy issue. Li et al. [99] discussed several statistical-based analysis methods that are summarized along with their formulas in Table 10:

- **Low variance:** Using this method, features with low variance are filtered out.
- **T-score [100]:** It is used for binary classification to determine whether a feature causes a statistically significant difference between two classes.
- **Chi-square score [101]:** This method helps to determine how independent a feature is of the class label.
- **Gini index [102]:** Different classes of samples can be separated using this method.
- **CFS [103]:** It evaluates how a certain feature fits in relation to the data.

7 COMMONLY USED TOOLS AND TECHNIQUES

In this section, we discuss several commonly used tools in the context of developing a testbed and creating a dataset that are summarized in Table 11.

7.1 Simulation

Moustafa & Slay [39] used a hardware network simulation called IXIA PerfectStorm (keysight.com/gb/en/products/network-test/network-test-hardware/perfectstorm.html) for the simulation of

Table 11. Summary of Commonly Used Tools and Techniques

Name	Purpose	References
Argus	Network Feature Extraction	[104][105][39][14][85]
Bro-IDS	Network Analyzer	[14][39]
Tcpdump	Packet Capture & Analyzer	[85][39][106][92]
Dpkt	Network Feature Extraction	[57][85][107][39]
IXIA PerfectStorm	Hardware Network Simulator	[14][108][39]
MATLAB	Programming & Numeric Computing	[109][110][111][112]
Nmap	Network Scanner	[46][57][85][113][9]
Hydra	Password Cracker	[85][57][114]
Mirage	Security Analysis	[56]
OpenWRT	OS for Routers	[115][116][106][92]
CICFlowmeter	Network Feature Extraction	[65][117][118][71]
LOIC	Network Stress Tool	[57] [113]
TSfresh	Time-Series Feature Extraction	[119][120][121]
Wireshark	Packet Capture & Analyzer	[122][46][107] [9]
Tshark	Packet Capture & Analyzer	[4][104] [85][123]
PyTorch	Implementing AD Algorithm	[71][124][75][68][34]
PySyft	Implementing AD Algorithm	[71][124][75][68]

network traffic and cyberattacks. According to their documentation, combining multiple simultaneous wired and wireless users with both application traffic and current attacks is possible with PerfectStorm Fusion. Furthermore, thousands of real-life end-user environments are simulated and tested using stateful applications and malicious traffic such as DoS and botnets. The workloads include data, video, voice, storage, and network applications.

7.2 Network Feature Extraction

We have previously discussed how important extracting features is for a dataset. Similarly, Frazão et al. [83] extracted network features using MATLAB (mathworks.com), where they extracted 68 network features, including packet timestamps, inter-packet arrival times, binary features defining which protocols were involved, and every field of the Ethernet, ARP, IP, ICMP, UDP, TCP and MODBUS over TCP headers. MATLAB can also be used to simulate an IoT environment using MATLAB Simulink features as presented by Al Farooq et al. [125]. According to MATLAB's website, both the model-based design and multi-domain simulation are supported by Simulink. Argus (openargus.org) is another network extraction tool that was used by Koroniotis et al. [85] to extract the record start time, record last time, standard deviation of aggregated records, class label: 0 for normal traffic, 1 for attack traffic, and traffic category among other network features. Likewise, Moustafa & Slay [39] extracted three categories of network features: basic, content, and time using Argus. In addition to auditing IP traffic, Argus provides network activity audit technology for all network traffic, according to their website. Along with Argus, Moustafa & Slay [39] used a network extraction tool called Bro-IDS (zeek.org) to extract network features and match the extracted features with the extracted features from Argus. Their website states that Bro-IDS can be configured to act as an IDS and detect well-known attacks. It is also important to know that Bro-IDS has been renamed Zeek. Dpkt (dpkt.readthedocs.io) is another feature extraction tool written in Python for creating and parsing network packets, which supports basic TCP/IP protocols. Dadkhah et al. [57] extracted 48 features from each device in their experiment that include packet size, protocol, epoch timestamp, and Ethernet frame size, among other features, using dpkt. According to Teh et al. [74], Tsfresh (tsfresh.readthedocs.io/en/latest/) is a widely used ML library for time series feature extraction that is written in Python. Ouyang et al. [120] also used Tsfresh in the context of power consumption to extract features from the power consumption data of 54,174 customers.

Furthermore, Liu et al. [119] used Tsfresh along with LSTM to present a sensor fault classification method that is based on feature extraction.

7.3 Network Capture and Analyzing

Wireshark (wireshark.org) is a well-known useful tool that is intended to capture and analyze network traffic. It supports hundreds of protocols to be analyzed and provides online and offline analyzing capabilities. Its interface is easy to read, and network capture can be filtered using port numbers, protocols, and IP addresses, among other attributes. Ren et al. [126] used Wireshark to capture network traffic in their project. According to their paper, Wireshark does not support all encrypted protocols. Tshark (wireshark.org/docs/man-pages/tshark.html) is the terminal version of Wireshark, which becomes handy in case the use of GUI is not feasible. Similarly, TCPDump (tcpdump.org) is another packet analyzer and network capture tool that is widely used nowadays. Hamza et al. [106] captured local and remote network traffic consisting of malicious and benign traffic for 16 days using TCPDump and stored pcap files on a hard disk that is attached to the network gateway. Similarly, CICFlowmeter (unb.ca/cic) is a network analyzer and generator that the Canadian Institute for Cybersecurity developed. It can generate bidirectional traffic and can capture up to 80 different features, such as duration and length of packets. Additionally, a user can select what features to be captured and can add new features. CICFlowmeter has been used by Ullah & Mahmoud [127] to extract network features from pcap files of five different datasets.

7.4 Router Management

To maximize the experience of having full control of routers and access points, Bhatt & Morais [116] configured the router they used in their research using OpenWRT (openwrt.org). OpenWRT is an embedded Linux operating system whose file system is fully writable and has a package management feature rather than creating a single, static firmware, as mentioned on their website. Additionally, Pessoa & Duarte-Figueiredo [128] used OpenWRT to change the firmware of the TPLink1043nd v2.1 access point to enable OpenFlow connections.

7.5 Cyber Anomalies

To perform cyber anomalies and test the security of their testbed, Dadkhah et al. [57] performed DoS attack using LOIC, which is a network stress tool that performs cyberattacks. Likewise, Bhuyan et al. [113] used LOIC to generate a distributed DoS attack. Security analysis of wireless communications is made easier with Mirage, a powerful and modular framework that is based on Romain Cayre's research on IoT security. The tool provides hackable wireless protocol stacks such as Zigbee, BLE, and Wi-Fi. Using Mirage, Gimenez et al. [56] performed different types of attacks such as DoS, injection, and probe. Hydra is another cyber anomaly tool that helps crack passwords by performing dictionary attacks. Similarly, Koroniotis et al. [85] performed a dictionary attack on their SSH service. Furthermore, Rose et al. [114] created malicious pcap files performing different attacks, using Hydra to perform brute force attacks to test their framework. Another well-known security tool is Nmap, an open-source security auditing and network discovery tool. It can find hosts within a network, services each host is running, and ports available. Furthermore, Nmap can be used to perform cyberattacks by running malicious scripts through the Nmap scripting engine. As an example of using Nmap, Siboni et al. [46] used Nmap to perform a network mapping attack. Even though Nmap can work on different operating systems, they modified Nmap to work on Android as the experiment was conducted on a Sony smartwatch. For vulnerability scanning, Tekeoglu & Tosun [107] used Nmap to scan for vulnerability on a FireTV stick, which shows severe threats. First, a vulnerability was found in the FireTV stick server running on port 49986 with a 10 (out of 10) severity score against `Format String Attack on URI`. Another vulnerability was also found with

Table 12. Anomaly Detection Evaluation Metrics Summary

Paper	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-Score	Threshold	Runtime	ROC	AUC	EER	MCC	PPV	MAE	MSE	RMSE	BA
Sikder et al. [48]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Ullah & Mahmoud [65]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Kang et al. [59]	○	○	○	○	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○
Hosseini et al. [66]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Li et al. [67]	●	●	●	●	○	○	○	○	●	○	●	●	●	○	○	○	○	○	○
Liu et al. [68]	●	●	●	●	●	○	○	○	●	○	○	○	○	○	○	○	○	○	○
Mudgerikar et al. [69]	●	●	●	●	●	●	●	●	○	○	●	○	○	○	○	○	○	○	○
Djenouri et al. [70]	●	●	●	●	●	●	●	●	○	●	○	○	○	○	○	○	○	○	○
Mothukuri et al. [71]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Shafi et al. [72]	●	●	●	●	●	●	●	●	○	○	●	○	○	○	○	○	○	○	○
Yahyaoui et al. [73]	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Gimenez et al. [56]	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○
Heartfield et al. [4]	●	●	●	●	●	●	●	●	○	○	●	●	○	○	●	○	○	○	○
Teh et al. [74]	●	●	●	●	○	○	○	○	●	○	○	○	○	●	○	○	○	○	○
Sater & Hamza [75]	●	●	●	●	○	●	●	●	○	○	●	●	○	●	○	●	●	●	●
Sarwar et al. [76]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Dissem et al. [77]	●	●	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
Xiao et al. [78]	●	●	●	●	○	○	●	●	○	○	○	○	○	○	○	○	○	○	○
Li et al. [79]	●	●	●	●	○	●	●	○	○	●	○	○	○	○	○	○	○	○	○

○ Has not been used, ● Has been used.

a 7.5 (out of 10) score of severity on the same port called `CERN httpd CGI name heap overflow`, which allows an attacker to stop a code on the server or even run a malicious code on the server.

7.6 Anomaly Detection Algorithms Implementation

Applying anomaly detection models is a part of the anomaly detection process; thus, we also list tools that help implement an anomaly detection model. Sater & Hamza [75] used PyTorch (pytorch.org) to implement the stacked LSTM as an anomaly detection algorithm. PyTorch is an open-source ML framework that is used for DL utilizing GPUs and CPUs. Similarly, PySyft (openmined.github.io/PySyft/index.html) is another open-source tool that is used to provide secure and private DL in Python. Mothukuri et al. [71] used PySyft along with PyTorch to implement their FL approach.

8 EVALUATION METRICS

For every anomaly detection technique presented, there must be an evaluation criterion that gives an indication of how efficient an anomaly detection technique is. We emphasize a range of evaluation metrics, each serving unique purposes in anomaly detection. Accuracy, a foundational metric, calculates the proportion of correct predictions but may mislead in imbalanced datasets where anomalies are rare [129]. Precision measures true positives among predicted positives, crucial when false positives affect system performance, such as in IoT scenarios. Recall, focusing on true positives among actual positives, is vital in critical contexts like cybersecurity. The F1-score, balancing precision and recall, suits imbalanced datasets by addressing trade-offs between false positives and missed detections. **Receiver Operating Characteristic (ROC)** curves and **Area Under the Curve (AUC)** illustrate the trade-off between sensitivity and specificity, providing valuable insights for reducing missed anomalies and false alarms. Metrics like TPR and **false positive rate (FPR)** provide detailed views of detection performance, especially in real-time systems where frequent alerts can lead to alert fatigue [130]. Selecting the right metric is key to evaluating anomaly detection systems effectively in specific applications.

In this section, we present the most common evaluation metrics that are used to evaluate different anomaly detection techniques as summarized in Table 12. Furthermore, we summarize value interpretations that yield a quality anomaly detection technique in Figure 7 and formulas of discussed evaluation metrics in Table 13

- **True positive (TP)** is when an anomaly is correctly predicted by a model.
- **True negative (TN)** is when the absence of an anomaly is correctly predicted by a model.
- **False positive (FP)** is when an anomaly is incorrectly predicted by a model.
- **False negative (FN)** is when the absence of an anomaly is incorrectly predicted by a model.
- **Accuracy** measures how accurate a model's prediction is. Accuracy is inefficient if a dataset is imbalanced [129].
- **Precision** measures the ratio of predicted anomalies to the total data samples [129].
- **Recall** measures how many anomalies (TP) were classified by the model [129].
- **F1-Score** measures the accuracy of a model, using precision and recall, especially when the dataset is imbalanced [66].
- **Threshold** is a value that distinguishes between two binary classes, in our case, normal or abnormal. Less than a threshold is a normal instance, whereas any value over it is an anomaly.
- **Runtime** is the time it takes a model to completely run and finish, which is also known as computational time.
- **ROC** provides a graphical representation of a model's performance using the TPR, represented as a y-axis in a graph, and (FPR), represented as an x-axis in a graph [129].
- **AUC** measures the area under the ROC curve. Essentially, it measures how much area under the ROC has been covered [129].
- **Equal Error Rate (EER)** is an intersection point between the TPR and FPR on the ROC graph where FPR and FN rate (FNR) values are equal [129].
- **Matthews correlation coefficient (MCC)** measures how accurate a model is in terms of anomaly prediction taking into consideration all four evaluation metrics: TP, TF, FP, and FN. Furthermore, MCC yields a high score if most of both negative and positive data samples are correctly predicted [131].
- **Predictive positive value (PPV)** measures the proportion of positive anomalies that are actually positive [132]
- **Mean absolute error (MAE)** evaluates how closely the predicted anomalies match the actual anomalies [129].
- **Mean squared error (MSE)** is similar to MAE, but it is more sensitive to anomalies because errors are squared [129].
- **Root mean squared error (RMSE)** is a measure of the average of squared errors squared [129].
- **Balanced accuracy (BA)** measures how accurate the prediction of a model is when the dataset is imbalanced [75].

9 CONCLUSION

We reviewed and presented guidelines based on scientific background to design a testbed, choose a testbed layout, select users and devices, and define the experimental condition for the testbed. Additionally, different anomaly-creation methods were presented. We have also reviewed multiple anomaly detection mechanisms in terms of the detection algorithm used, whether was it tested on a testbed or a dataset, and the types of anomalies generated.

Furthermore, we discussed the type of infrastructure to create a dataset and what network configuration to choose because different wireless communication protocols might need different hubs to manage them; hence, the budget of the implementation will go up. In terms of data collection,

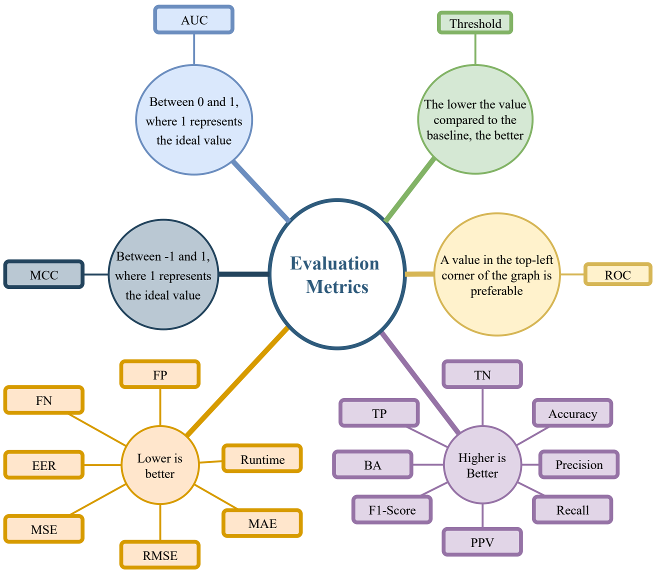
AUC
Between 0 and 1, where 1 represents the ideal value
Threshold
The lower the value compared to the baseline, the better
Evaluation Metrics
MCC
Between -1 and 1, where 1 represents the ideal value
A value in the top-left corner of the graph is preferable
ROC
FP
FN
EER
Lower is better
Runtime
MSE
RMSE
MAE
TN
TP
Accuracy
BA
Higher is Better
Precision
F1-Score
Recall
PPV

Fig. 7. A mind map of evaluation metrics used in anomaly detection. Nodes categorize metrics into groups like accuracy, recall, precision, and error rates, with arrows indicating whether higher or lower values are preferred.

Table 13. Anomaly Detection Evaluation Metrics Formulas

Evaluation Metric	Formula	No.
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	(1)
Precision	$\frac{TP}{TP+FP}$	(2)
Recall	$\frac{TP}{TP+FN}$	(3)
F1-Score	$2 \times \frac{Precision \times Recall}{Precision+Recall}$	(4)
ROC	$TPR = \frac{TP}{TP+FN} \quad FPR = \frac{FP}{FP+TN}$	(5)
MCC	$\frac{(TP \times TN)-(FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$	(6)
PPV	$\frac{TP}{TP+FP}$	(7)
MAE	$\frac{1}{n}\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$	(8)
MSE	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	(9)
RMSE	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	(10)
BA	$\frac{TPR+TNR}{2}$	(11)

we provided six different statuses in which data are collected during power, idle, interaction, scenarios, active, and attacks. To generate the data, different traffic scenarios must be considered and followed. We provided examples of smart home scenarios that might differ depending on the dataset's goal. Dataset analysis is an important aspect of the dataset creation process, which

Table 14. Applicability of Anomaly Detection Methodologies Across Smart Environments and Applications

Application/Domain	Smart Home	Smart Building	Smart City	Industrial IoT
Anomaly Types	●	●	●	●
Detection Technique	●	●	●	●
Testbed Characteristics	●	●	Partially addressed by the use of emulation(CORE). Requires further scalability testing	Limited coverage of industrial devices
Evaluation Metrics	●	●	●	●
Dataset Creation	●	●	Requires real-world traffic data	Limited coverage of industrial protocols

● Fully covered

consists of feature extraction and statistical analysis. Several network feature types were presented with examples of some distinct network features and their description. Statistical-based analysis methods with low variance, T-score, chi-squared score, Gini index, and CFS were discussed in detail.

A set of useful tools, in terms of capturing and analyzing network traffic such as Wireshark, extracting network features such as CICFlowmeter, and performing cyberattacks such as Mirage, was presented along with their websites and references. We also highlighted the most common anomaly detection evaluation metrics used in the literature.

To broaden the applicability of our study, which encompasses both urban-focused and industrial applications to some extent, we have included a diverse range of anomaly types, anomaly detection techniques, comprehensive testbed characteristics, and evaluation metrics. While our findings are validated against specific testing scenarios, further research is necessary to assess the framework's scalability in network-heavy environments, particularly those deploying a large volume of devices. This would provide valuable insights into its efficiency in managing challenges faced in smart cities and industrial systems. Table 14 illustrates the applicability and limitation of our study.

ACKNOWLEDGMENTS

This work has been partially supported by the PETRAS National Centre of Excellence for IoT Systems Cybersecurity (Resilient Built Environments (ResBE) Project) EP/S035362/1, Digital Catapult Researcher-in-Residence program (EP/T517203/1), Innovate UK CyberASAP, GCHQ National Resilience Fellowship, and EPSRC National Edge Artificial Intelligence Hub (EP/Y028813/1).

REFERENCES

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

[2] Ahmad al Qerem, Mohammad Alauthman, Ammar Almomani, and Brij B Gupta. Iot transaction processing through cooperative concurrency control on fog–cloud computing environment. *Soft Computing*, 24, 04 2020.

[3] James Manyika, Michael Chui, Peter Bisson, Jonathan Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. The internet of things mapping the value beyond the hype, Jun 2015.

[4] Ryan Heartfield, George Loukas, Anatolij Bezemskij, and Emmanouil Panaousis. Self-Configurable Cyber-Physical Intrusion Detection for Smart Homes Using Reinforcement Learning. *IEEE Transactions on Information Forensics and Security*, 16:1720–1735, 2021. Conference Name: IEEE Transactions on Information Forensics and Security.

[5] Sowmya Ramapatruni, Sandeep Nair Narayanan, Sudip Mittal, Anupam Joshi, and Karuna Joshi. Anomaly detection models for smart home security. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 19–24, 2019.

[6] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. Is anybody home? inferring activity from smart home network traffic. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 245–251, 2016.

[7] Vic Barnett and Toby Lewis. *Outliers in statistical data.* Wiley, 1974.

[8] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, 2020.

[9] Chun Zhu, Weihua Sheng, and Meiqin Liu. Wearable sensor-based behavioral anomaly detection in smart assisted living systems. *IEEE Transactions on Automation Science and Engineering*, 12(4):1225–1234, 2015.

[10] Mohammed Alosaimi, Omer Rana, and Charith Perera. Testbeds and evaluation frameworks for anomaly detection within built environments: A systematic review (supplementary material). https://iotgarage.net/publications/pdfs/testbed_survey_supp_material.pdf, 2025.

[11] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, July 2020. Conference Name: IEEE Internet of Things Journal.

[12] Yassine Himeur, Khalida Ghanem, Abdullah Alsalemi, Faycal Bensaali, and Abbes Amira. Artificial intelligence based anomaly detection of energy consumption in buildings: A review, current trends and new perspectives. *Applied Energy*, 287:116601, April 2021.

[13] L. Erhan, M. Ndubuaku, M. Di Mauro, W. Song, M. Chen, G. Fortino, O. Bagdasar, and A. Liotta. Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*, 67:64–79, March 2021.

[14] Nour Moustafa, Jiankun Hu, and Jill Slay. A holistic review of Network Anomaly Detection Systems: A comprehensive survey. *Journal of Network and Computer Applications*, 128:33–55, February 2019.

[15] Anuroop Gaddam, Tim Wilkin, and Maia Angelova. Anomaly Detection Models for Detecting Sensor Faults and Outliers in the IoT - A Survey. In *2019 13th International Conference on Sensing Technology (ICST)*, pages 1–6, December 2019. ISSN: 2156-8073.

[16] Kelton A. P. da Costa, João P. Papa, Celso O. Lisboa, Roberto Munoz, and Victor Hugo C. de Albuquerque. Internet of Things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, 151:147–157, March 2019.

[17] Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng (Daphne) Yao. Deep Learning-based Anomaly Detection in Cyber-physical Systems: Progress and Opportunities. *ACM Computing Surveys*, 54(5):1–36, June 2022.

[18] Elhadj Benkhelifa, Thomas Welsh, and Walaa Hamouda. A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems. *IEEE Communications Surveys & Tutorials*, 20(4):3496–3509, 2018. Conference Name: IEEE Communications Surveys & Tutorials.

[19] Muhammad Fahim and Alberto Sillitti. Anomaly Detection, Analysis and Prediction Techniques in IoT Environment: A Systematic Literature Review. *IEEE Access*, 7:81664–81681, 2019. Conference Name: IEEE Access.

[20] Ayman Taha and Ali S. Hadi. Anomaly Detection Methods for Categorical Data: A Review. *ACM Computing Surveys*, 52(2):38:1–38:35, May 2019.

[21] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. Deep Learning for Medical Anomaly Detection – A Survey. *ACM Computing Surveys*, 54(7):141:1–141:37, July 2021.

[22] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[23] Kinza Arshad, Rao Faizan Ali, Amgad Muneer, Izzatdin Abdul Aziz, Sheraz Naseer, Nabeel Sabir Khan, and Shakirah Mohd Taib. Deep reinforcement learning for anomaly detection: A systematic review. *IEEE Access*, 10:124017–124035, 2022.

[24] Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In Michelangelo Ceci, Jaakko Hollmén, Ljupčo Todorovski, Celine Vens, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 20–35, Cham, 2017. Springer International Publishing.

[25] Yingjie Tian, Mahboubeh Mirzabagheri, Seyed Mojtaba Hosseini Bamakan, Huadong Wang, and Qiang Qu. Ramp loss one-class support vector machine; a robust and effective approach to anomaly detection problems. *Neurocomputing*, 310:223–235, 2018.

[26] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Marc Niethammer, Martin Styner, Stephen Aylward, Hongtu Zhu, Ipek Oguz, Pew-Thian Yap, and Dinggang Shen, editors, *Information Processing in Medical Imaging*, pages 146–157, Cham, 2017. Springer International Publishing.

[27] Guansong Pang, Anton Hengel, Chunhua Shen, and Longbing Cao. *Deep Reinforcement Learning for Unknown Anomaly Detection.* arXiv, September 2020.

[28] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 665–674, New York, NY, USA, 2017. Association for Computing Machinery.

[29] Larry Shoemaker and Lawrence O. Hall. Anomaly detection using ensembles. In Carlo Sansone, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems*, pages 6–15, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[30] Xin Zhang, Pingping Wei, and Qingling Wang. A hybrid anomaly detection method for high dimensional data. *PeerJ Computer Science*, 9:e1199, January 2023.

[31] Raed Abdel Sater and A. Ben Hamza. A federated learning approach to anomaly detection in smart buildings. *ACM Trans. Internet Things*, 2(4), August 2021.

[32] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[33] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6), dec 2017.

[34] Jiuqi Elise Zhang, Di Wu, and Benoit Boulet. Time Series Anomaly Detection via Reinforcement Learning-Based Model Selection, July 2022. arXiv:2205.09884 [cs].

[35] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. Deep Semi-Supervised Anomaly Detection, February 2020. arXiv:1906.02694 [cs, stat].

[36] Guansong Pang, Chunhua Shen, and Anton van den Hengel. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 353–362, New York, NY, USA, 2019. Association for Computing Machinery.

[37] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1), mar 2012.

[38] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. Learning Representations of Ultrahigh-dimensional Data for Random Distance-based Outlier Detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2041–2050, July 2018. arXiv:1806.04808 [cs, stat].

[39] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, 2015.

[40] W. Schiffmann, M. Joost, and R. Werner. Synthesis and Performance Analysis of Multilayer Neural Network Architectures, 1992.

[41] W Schiffmann, M Joost, and R Werner. Optimization of the Backpropagation Algorithm for Training Multilayer Perceptrons. page 36, 1994.

[42] Robert Müller. Towards Anomaly Detection in Reinforcement Learning. *AAMAS '22: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, page 5, 2022.

[43] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.

[44] Jie Jiang, Riccardo Pozza, Nigel Gilbert, and Klaus Moessner. Makesense: an iot testbed for social research of indoor activities. *ACM Transactions on Internet of Things*, 1(3):1–25, 2020.

[45] Ryan Wen Liu, Yu Guo, Yuxu Lu, Kwok Tai Chui, and Brij B. Gupta. Deep network-enabled haze visibility enhancement for visual iot-driven intelligent transportation systems. *IEEE Transactions on Industrial Informatics*, 19(2):1581–1591, 2023.

[46] Shachar Siboni, Asaf Shabtai, Nils O. Tippenhauer, Jemin Lee, and Yuval Elovici. Advanced Security Testbed Framework for Wearable IoT Devices. *ACM Transactions on Internet Technology*, 16(4):1–25, December 2016.

[47] Sajal Bhatia, Desmond Schmidt, George Mohay, and Alan Tickle. A framework for generating realistic traffic for Distributed Denial-of-Service attacks and Flash Events. *Computers & Security*, 40:95–107, February 2014.

[48] Amit Kumar Sikder, Leonardo Babun, and A Selcuk Uluagac. Aegis+ a context-aware platform-independent security framework for smart home systems. *Digital Threats: Research and Practice*, 2(1):1–33, 2021.

[49] Xinyu Lei, Guan-Hua Tu, Alex X. Liu, Chi-Yu Li, and Tian Xie. The Insecurity of Home Digital Voice Assistants - Vulnerabilities, Attacks and Countermeasures. In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9, May 2018.

[50] Diane J. Cook, Aaron S. Crandall, Brian L. Thomas, and Narayanan C. Krishnan. CASAS: A Smart Home in a Box. *Computer*, 46(7):62–69, July 2013. Conference Name: Computer.

[51] Waheb A. Jabbar, Tee Kok Kian, Roshahliza M. Ramli, Siti Nabila Zubir, Nurthaqifah S. M. Zamrizaman, Mohammed Balfaqih, Vladimir Shepelev, and Soltan Alharbi. Design and Fabrication of Smart Home With Internet of Things Enabled Automation System. *IEEE Access*, 7:144059–144074, 2019. Conference Name: IEEE Access.

[52] Mauro Piva, Andrea Coletta, Gaia Maselli, and John A. Stankovic. Environment-driven Communication in Battery-free Smart Buildings. *ACM Transactions on Internet of Things*, 2(2):1–30, May 2021.

[53] Sunny Behal and Krishan Kumar. Detection of DDoS attacks and flash events using information theory metrics–An empirical investigation. *Computer Communications*, 103:18–28, May 2017.

[54] Jingjing Ren, Daniel J Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Proceedings of the Internet Measurement Conference*, pages 267–279, 2019.

[55] Juan Ignacio Iturbe Araya and Helena Rifà-Pous. Anomaly-based cyberattacks detection for smart homes: A systematic literature review. *Internet of Things*, 22:100792, 2023.

[56] Pierre-Francois Gimenez, Jonathan Roux, Eric Alata, Guillaume Auriol, Mohamed Kaaniche, and Vincent Nicomette. RIDS: Radio Intrusion Detection and Diagnosis System for Wireless Communications in Smart Environment. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–1, July 2021.

[57] Sajjad Dadkhah, Hassan Mahdikhani, Priscilla Kyei Danso, Alireza Zohourian, Kevin Anh Truong, and Ali A. Ghorbani. Towards the Development of a Realistic Multidimensional IoT Profiling Dataset. In *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, pages 1–11, August 2022.

[58] Matt Webster, Michael Breza, Clare Dixon, Michael Fisher, and Julie McCann. Exploring the effects of environmental conditions and design choices on iot systems using formal methods. *Journal of Computational Science*, 45:101183, 2020.

[59] Kai Kang, Lijie Xu, Wei Wang, Guoquan Wu, Jun Wei, Wei Shi, and Jizhong Li. A hierarchical automata based approach for anomaly detection in smart home devices. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 1–8. IEEE, 2020.

[60] Abdel Mlak Said, Aymen Yahyaoui, and Takoua Abdellatif. Efficient Anomaly Detection for Smart Hospital IoT Systems. *Sensors*, 21(4):1026, January 2021. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[61] Jelena Mirkovic, Alefiya Hussain, Sonia Fahmy, Peter Reiher, and Roshan K. Thomas. Accurately Measuring Denial of Service in Simulation and Testbed Experiments. *IEEE Transactions on Dependable and Secure Computing*, 6(2):81–95, April 2009. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[62] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with deter: a testbed for security research. In *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006.*, pages 10 pp.–388, 2006.

[63] J. Lundström, W. Ourique De Morais, and M. Cooney. A holistic smart home demonstrator for anomaly detection and response. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 330–335, March 2015.

[64] Fengqin Zuo, Damin Zhang, Lun Li, Qing He, and Jiaxin Deng. Gsooa-1ddrsn: Network traffic anomaly detection based on deep residual shrinkage networks. *Heliyon*, 10(11):e32087, 2024.

[65] Imtiaz Ullah and Qusay H. Mahmoud. A Framework for Anomaly Detection in IoT Networks Using Conditional Generative Adversarial Networks. *IEEE Access*, 9:165907–165931, 2021. Conference Name: IEEE Access.

[66] Sayed Saeed Hosseini, Kodjo Agbossou, Sousso Kelouwani, Alben Cardenas, and Nilson Henao. A practical approach to residential appliances on-line anomaly detection: A case study of standard and smart refrigerators. *IEEE Access*, 8:57905–57922, 2020.

[67] Ruoyu Li, Qing Li, Jianer Zhou, and Yong Jiang. Adriot: An edge-assisted anomaly detection framework against iot-based network attacks. *IEEE Internet of Things Journal*, 9(13):10576–10587, 2022.

[68] Yi Liu, Sahil Garg, Jiangtian Nie, Yang Zhang, Zehui Xiong, Jiawen Kang, and M. Shamim Hossain. Deep Anomaly Detection for Time-Series Data in Industrial IoT: A Communication-Efficient On-Device Federated Learning Approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, April 2021. Conference Name: IEEE Internet of Things Journal.

[69] Anand Mudgerikar, Puneet Sharma, and Elisa Bertino. Edge-Based Intrusion Detection for IoT devices. *ACM Transactions on Management Information Systems*, 11(4):1–21, December 2020.

[70] Youcef Djenouri, Djamel Djenouri, Asma Belhadi, Gautam Srivastava, and Jerry Chun-Wei Lin. Emergent Deep Learning for Anomaly Detection in Internet of Everything. *IEEE Internet of Things Journal*, pages 1–1, 2021. Conference Name: IEEE Internet of Things Journal.

[71] Viraaji Mothukuri, Prachi Khare, Reza M. Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. Federated-Learning-Based Anomaly Detection for IoT Security Attacks. *IEEE Internet of Things Journal*, 9(4):2545–2554, February 2022. Conference Name: IEEE Internet of Things Journal.

[72] Qaisar Shafi, Abdul Basit, Saad Qaisar, Abigail Koay, and Ian Welch. Fog-Assisted SDN Controlled Framework for Enduring Anomaly Detection in an IoT Network. *IEEE Access*, 6:73713–73723, 2018. Conference Name: IEEE Access.

[73] Aymen Yahyaoui, Takoua Abdellatif, Sami Yangui, and Rabah Attia. READ-IoT: Reliable Event and Anomaly Detection Framework for the Internet of Things. *IEEE Access*, 9:24168–24186, 2021. Conference Name: IEEE Access.

[74] Hui Yie Teh, Kevin I-Kai Wang, and Andreas W. Kempa-Liehr. Expect the Unexpected: Unsupervised Feature Selection for Automated Sensor Anomaly Detection. *IEEE Sensors Journal*, 21(16):18033–18046, August 2021. Conference Name: IEEE Sensors Journal.

[75] Raed Abdel Sater and A. Ben Hamza. A Federated Learning Approach to Anomaly Detection in Smart Buildings. *ACM Transactions on Internet of Things*, 2(4):1–23, November 2021.

[76] Nadeem Sarwar, Imran Sarwar Bajwa, Muhammad Zunnurain Hussain, Muhammad Ibrahim, and Khizra Saleem. Iot network anomaly detection in smart homes using machine learning. *IEEE Access*, 11:119462–119480, 2023.

[77] Maher Dissem, Manar Amayri, and Nizar Bouguila. Neural architecture search for anomaly detection in time-series data of smart buildings: A reinforcement learning approach for optimal autoencoder design. *IEEE Internet of Things Journal*, 11(10):18059–18073, 2024.

[78] Jingyu Xiao, Zhiyao Xu, Qingsong Zou, Qing Li, Dan Zhao, Dong Fang, Ruoyu Li, Wenxin Tang, Kang Li, Xudong Zuo, Penghui Hu, Yong Jiang, Zixuan Weng, and Michael R. Lyu. Make your home safe: Time-aware unsupervised user behavior anomaly detection in smart homes via loss-guided mask. KDD '24, page 3551–3562, New York, NY, USA, 2024. Association for Computing Machinery.

[79] Ruoyu Li, Qing Li, Yucheng Huang, Qingsong Zou, Dan Zhao, Zhengxin Zhang, Yong Jiang, Fa Zhu, and Athanasios V. Vasilakos. Seiot: Detecting anomalous semantics in smart homes via knowledge graph. *IEEE Transactions on Information Forensics and Security*, 19:7005–7018, 2024.

[80] Fatima Ezzeddine, Mirna Saad, Omran Ayoub, Davide Andreoletti, Martin Gjoreski, Ihab Sbeity, Marc Langheinrich, and Silvia Giordano. Differential privacy for anomaly detection: Analyzing the trade-off between privacy and explainability, 2024.

[81] Yixuan Zhang, Basem Suleiman, Muhammad Johan Alibasa, and Farnaz Farid. Privacy-aware anomaly detection in iot environments using fedgroup: A group-based federated learning approach. *J. Netw. Syst. Manage.*, 32(1), January 2024.

[82] Anand Mudgerikar, Puneet Sharma, and Elisa Bertino. Edge-based intrusion detection for iot devices. *ACM Trans. Manage. Inf. Syst.*, 11(4), oct 2020.

[83] Ivo Frazão, Pedro Henriques Abreu, Tiago Cruz, Hélder Araújo, and Paulo Simões. Denial of Service Attacks: Detecting the Frailties of Machine Learning Algorithms in the Classification Process. In Eric Luiijf, Inga Žutautaitė, and Bernhard M. Hämmerli, editors, *Critical Information Infrastructures Security*, Lecture Notes in Computer Science, pages 230–235, Cham, 2019. Springer International Publishing.

[84] Laura Vigoya, Diego Fernandez, Victor Carneiro, and Fidel Cacheda. Annotated Dataset for Anomaly Detection in a Data Center with IoT Sensors. *Sensors*, 20(13):3745, January 2020. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

[85] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100:779–796, November 2019.

[86] Nour Moustafa. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_iot datasets. *Sustainable Cities and Society*, 72:102994, 2021.

[87] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, July 2009. ISSN: 2329-6275.

[88] Elaheh Biglar Beigi, Hossein Hadian Jazi, Natalia Stakhanova, and Ali A. Ghorbani. Towards effective feature selection in machine learning-based botnet detection approaches. In *2014 IEEE Conference on Communications and Network Security*, pages 247–255, October 2014.

[89] Wolfgang John and S. Tafvelin. Differences between In- and Outbound Internet Backbone Traffic. 2007.

[90] David Zhao, Issa Traore, Ali Ghorbani, Bassam Sayed, Sherif Saad, and Wei Lu. Peer to Peer Botnet Detection Based on Flow Intervals. volume 376, June 2012.

[91] Wen-Hwa Liao and Chia-Ching Chang. Peer to Peer Botnet Detection Using Data Mining Scheme. In *2010 International Conference on Internet Technology and Applications*, pages 1–4, August 2010.

[92] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, August 2019. Conference Name: IEEE Transactions on Mobile Computing.

[93] S Revathi and Dr A Malathi. A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection. *International Journal of Engineering Research*, 2(12), 2013.

[94] Hanan Hindy, Ethan Bayne, Miroslav Bures, Robert Atkinson, Christos Tachtatzis, and Xavier Bellekens. Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset), November 2020. arXiv:2006.15340 [cs].

[95] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13), 2023.

[96] Yair Meidan, Vinay Sachidananda, Hongyi Peng, Racheli Sagron, Yuval Elovici, and Asaf Shabtai. IoT-deNAT: Outbound flow-based network traffic data of IoT and non-IoT devices behind a home NAT, July 2020.

[97] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. IoT Sentinel: Automated Device-Type Identification for Security Enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184, June 2017. arXiv:1611.04880 [cs].

[98] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. IoTSense: Behavioral Fingerprinting of IoT Devices, April 2018. arXiv:1804.03852 [cs].

[99] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature Selection: A Data Perspective. *ACM Computing Surveys*, 50(6):1–45, November 2018.

[100] John C. Davis. *Statistics and data analysis in geology.* New York Wiley & Sons, 1973.

[101] Huan Liu and R. Setiono. Chi2: feature selection and discretization of numeric attributes. pages 388–391, November 1995. ISSN: 1082-3409.

[102] GINI C. W. Variability and mutability, contribution to the study of statistical distributions and relations. studi cconomico-giuridici della r. universita de cagliari (1912). reviewed in : Light, r. j., margolin, b. h. : An analysis of variance for categorical data. *J. American Statistical Association*, 66:534–544, 1971.

[103] Mark A. Hall and Lloyd A. Smith. Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, page 235–239. AAAI Press, 1999.

[104] Nikolaos Vakakis, Odysseas Nikolis, Dimosthenis Ioannidis, Konstantinos Votis, and Dimitrios Tzovaras. Cybersecurity in SMEs: The Smart-Home/Office Use Case. In *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–7, September 2019. ISSN: 2378-4873.

[105] Khalid Albulayhi and Frederick T. Sheldon. An Adaptive Deep-Ensemble Anomaly-Based Intrusion Detection System for the Internet of Things. In *2021 IEEE World AI IoT Congress (AIIoT)*, pages 0187–0196, May 2021.

[106] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A. Benson, and Vijay Sivaraman. Detecting Volumetric Attacks on loT Devices via SDN-Based Monitoring of MUD Activity. In *Proceedings of the 2019 ACM Symposium on SDN Research*, pages 36–48, San Jose CA USA, April 2019. ACM.

[107] Ali Tekeoglu and Ali Şaman Tosun. A Testbed for Security and Privacy Analysis of IoT Devices. In *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 343–348, October 2016. ISSN: 2155-6814.

[108] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications*, 87:185–192, 2017.

[109] Minh Pham, Dan Yang, and Weihua Sheng. A Sensor Fusion Approach to Indoor Human Localization Based on Environmental and Wearable Sensors. *IEEE Transactions on Automation Science and Engineering*, 16(1):339–350, January 2019. Conference Name: IEEE Transactions on Automation Science and Engineering.

[110] Christopher Osiegbu, Seifemichael B. Amsalu, Fatemeh Afghah, Daniel Limbrick, and Abdollah Homaifar. Design and Implementation of an Autonomous Wireless Sensor-Based Smart Home. In *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, August 2015. ISSN: 1095-2055.

[111] Rijad Alisic, Marco Molinari, Philip E. Paré, and Henrik Sandberg. Ensuring Privacy of Occupancy Changes in Smart Buildings. In *2020 IEEE Conference on Control Technology and Applications (CCTA)*, pages 871–876, August 2020.

[112] Jiwon Choi, Hayoung Jeoung, Jihun Kim, Youngjoo Ko, Wonup Jung, Hanjun Kim, and Jong Kim. Detecting and Identifying Faulty IoT Devices in Smart Home with Context Extraction. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 610–621, June 2018. ISSN: 2158-3927.

[113] Monowar H Bhuyan, Dhruba K Bhattacharyya, and Jugal K Kalita. Towards Generating Real-life Datasets for Network Intrusion Detection. *International Journal of Network Security*, 2015.

[114] Joseph R Rose, Matthew Swann, Gueltoum Bendiab, Stavros Shiaeles, and Nicholas Kolokotronis. Intrusion Detection using Network Traffic Profiling and Machine Learning for IoT. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 409–415, June 2021. ISSN: 2693-9789.

[115] Marcin Szczodrak, Yong Yang, Dave Cavalcanti, and Luca P. Carloni. An open framework to deploy heterogeneous wireless testbeds for Cyber-Physical Systems. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 215–224, June 2013. ISSN: 2150-3117.

[116] Parth Bhatt and Anderson Morais. HADS: Hybrid Anomaly Detection System for IoT Environments. In *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, pages 191–196, December 2018.

[117] Pedro H. A. D. de Melo, Adriano Araújo Martins de Resende, Rodrigo Sanches Miani, and Pedro Frosi Rosa. Evaluation of one-class algorithms for anomaly detection in home networks. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 682–689, November 2021. ISSN: 2375-0197.

[118] Nilesh Vishwasrao Patil, C. Rama Krishna, and Krishan Kumar. SSK-DDoS: distributed stream processing framework based classification system for DDoS attacks. *Cluster Computing*, 25(2):1355–1372, April 2022.

[119] Gang Liu, Lili Li, Liangliang Zhang, Qing Li, and S. S. Law. Sensor faults classification for SHM systems using deep learning-based method with Tsfresh features. *Smart Materials and Structures*, 29(7):075005, May 2020. Publisher: IOP Publishing.

[120] Zhiyou Ouyang, Xiaokui Sun, and Dong Yue. Hierarchical Time Series Feature Extraction for Power Consumption Anomaly Detection. In Kang Li, Yusheng Xue, Shumei Cui, Qun Niu, Zhile Yang, and Patrick Luk, editors, *Advanced*

Computational Methods in Energy, Power, Electric Vehicles, and Their Integration, Communications in Computer and Information Science, pages 267–275, Singapore, 2017. Springer.

[121] Wei Zhang, Xiaowei Dong, Huaibao Li, Jin Xu, and Dan Wang. Unsupervised detection of abnormal electricity consumption behavior based on feature engineering. *IEEE Access*, 8:55483–55500, 2020.

[122] Bilal Al Baalbaki, Jesus Pacheco, Cihan Tunc, Salim Hariri, and Youssif Al-Nashif. Anomaly Behavior Analysis System for ZigBee in smart buildings. In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–4, November 2015. ISSN: 2161-5330.

[123] Holden Gordon, Christopher Batula, Bhagyashri Tushir, Behnam Dezfouli, and Yuhong Liu. Securing Smart Homes via Software-Defined Networking and Low-Cost Traffic Classification. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1049–1057, July 2021. ISSN: 0730-3157.

[124] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning, November 2018.

[125] Abdullah Al Farooq, Ehab Al-Shaer, Thomas Moyer, and Krishna Kant. IoTC2: A Formal Method Approach for Detecting Conflicts in Large Scale IoT Systems. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 442–447, April 2019. ISSN: 1573-0077.

[126] Jingjing Ren, Daniel J. Dubois, David Choffnes, Anna Maria Mandalari, Roman Kolcun, and Hamed Haddadi. Information Exposure From Consumer IoT Devices: A Multidimensional, Network-Informed Measurement Approach. In *Proceedings of the Internet Measurement Conference*, pages 267–279, Amsterdam Netherlands, October 2019. ACM.

[127] Imtiaz Ullah and Qusay H. Mahmoud. An Anomaly Detection Model for IoT Networks based on Flow and Flag Features using a Feed-Forward Neural Network. In *2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*, pages 363–368, January 2022. ISSN: 2331-9860.

[128] Talles Quintão Pessoa and Fátima Duarte-Figueiredo. NodePI : An integrated platform for smart homes. In *2017 IEEE 9th Latin-American Conference on Communications (LATINCOM)*, pages 1–6, November 2017.

[129] Samaneh Aminikhanghahi and Diane J. Cook. A Survey of Methods for Time Series Change Point Detection. *Knowledge and Information Systems*, 51(2):339–367, May 2017.

[130] Gang Yang, Chaojing Tang, and Xingtong Liu. Dualac2nn: Revisiting and alleviating alert fatigue from the detection perspective. *Symmetry*, 14(10), 2022.

[131] Davide Chicco, Matthijs J. Warrens, and Giuseppe Jurman. The matthews correlation coefficient (mcc) is more informative than cohen's kappa and brier score in binary classification assessment. *IEEE Access*, 9:78368–78381, 2021.

[132] Thomas F. Monaghan, Syed N. Rahman, Christina W. Agudelo, Alan J. Wein, Jason M. Lazar, Karel Everaert, and Roger R. Dmochowski. Foundational Statistical Principles in Medical Research: Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value. *Medicina*, 57(5):503, May 2021.

[133] Hossein Jafari, Xiangfang Li, Lijun Qian, and Yuanzhu Chen. Community based sensing: A test bed for environment air quality monitoring using smartphone paired sensors. In *2015 36th IEEE Sarnoff Symposium*, pages 12–17, September 2015.

[134] Hongfei Xue, Wenjun Jiang, Chenglin Miao, Fenglong Ma, Shiyang Wang, Ye Yuan, Shuochao Yao, Aidong Zhang, and Lu Su. DeepMV: Multi-View Deep Learning for Device-Free Human Activity Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–26, March 2020.

[135] J. Lundström, J. Synnott, E. Järpe, and C. D. Nugent. Smart home simulation using avatar control and probabilistic sampling. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 336–341, March 2015.

[136] Tam Van Nguyen, Jin Gook Kim, and Deokjai Choi. ISS: The Interactive Smart home Simulator. In *2009 11th International Conference on Advanced Communication Technology*, volume 03, pages 1828–1833, February 2009. ISSN: 1738-9445.

[137] Nasser Alshammari, Talal Alshammari, Mohamed Sedky, Justin Champion, and Carolin Bauer. OpenSHS: Open Smart Home Simulator. *Sensors*, 17(5):1003, May 2017. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[138] Julien Bruneau, Wilfried Jouve, and Charles Consel. DiaSim: A parameterized simulator for pervasive computing applications. In *2009 6th Annual International Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous*, pages 1–10, July 2009.

[139] J. Synnott, L. Chen, C.D. Nugent, and G. Moore. The creation of simulated activity datasets using a graphical intelligent environment simulation tool. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4143–4146, August 2014. ISSN: 1558-4615.

[140] Jörg Cassens, Felix Schmitt, Tobias Mende, and Michael Herczeg. CASi – A Generic Context Awareness Simulator for Ambient Systems. In Fabio Paternò, Boris de Ruyter, Panos Markopoulos, Carmen Santoro, Evert van Loenen, and Kris Luyten, editors, *Ambient Intelligence*, Lecture Notes in Computer Science, pages 421–426, Berlin, Heidelberg, 2012. Springer.

[141] Shadan Golestan, Alexandr Petcovici, Ioanis Nikolaidis, and Eleni Stroulia. Simulation-Based Deployment Configuration of Smart Indoor Spaces. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 358–363, 2019.

[142] Xianzhong Ding, Wan Du, and Alberto Cerpa. OCTOPUS: Deep Reinforcement Learning for Holistic Smart Building Control. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 326–335, New York NY USA, November 2019. ACM.

[143] Mahmud Hossain, Shahid Noor, Yasser Karim, and Ragib Hasan. IoTbed: A Generic Architecture for Testbed as a Service for Internet of Things-Based Systems. In *2017 IEEE International Congress on Internet of Things (ICIOT)*, pages 42–49, Honolulu, HI, USA, June 2017. IEEE.

[144] Ghada Alsuhli and Ahmed Khattab. A Fog-based IoT Platform for Smart Buildings. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 174–179, February 2019.

[145] Pradnya Gaonkar, Advait Lonkar, GVK Sasirekha, Jyotsna Bapat, and Debabrata Das. Comfort Management System for Smart Buildings: An Open-Source Scalable Prototype. In *2020 IEEE-HYDCON*, pages 1–5, September 2020.

[146] Driss Benhaddou, Lotanna Afogbuom, Farouk Attia, and Muhammad Anan. Autonomous Living Building: Adapting to Occupant's Behavior. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 1803–1808, June 2019. ISSN: 2376-6506.

[147] Quanqing Xu, Zhaozheng He, Zengxiang Li, and Mingzhong Xiao. Building an Ethereum-Based Decentralized Smart Home System. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 1004–1009, December 2018. ISSN: 1521-9097.

[148] Jun Zhang, Guangming Song, Hui Wang, and Tianhua Meng. Design of a Wireless Sensor Network Based Monitoring System for Home Automation. In *2011 International Conference on Future Computer Sciences and Application*, pages 57–60, June 2011.

[149] Miroslav Bures, Bestoun S. Ahmed, Vaclav Rechtberger, Matej Klima, Michal Trnka, Miroslav Jaros, Xavier Bellekens, Dani Almog, and Pavel Herout. PatrIoT: IoT Automated Interoperability and Integration Testing Framework. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 454–459, April 2021. ISSN: 2159-4848.

[150] Qiaozhi Xu and Junxing Zhang. piFogBed: A Fog Computing Testbed Based on Raspberry Pi. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, October 2019. ISSN: 2374-9628.

[151] Mingfu Li and Hung-Ju Lin. Design and Implementation of Smart Home Control Systems Based on Wireless Sensor Networks and Power Line Communications. *IEEE Transactions on Industrial Electronics*, 62(7):4430–4442, July 2015. Conference Name: IEEE Transactions on Industrial Electronics.

[152] Dali Zhu, Na Pang, Gang Li, Wenjing Rong, and Zheming Fan. WiN: Non-invasive Abnormal Activity Detection Leveraging Fine-Grained WiFi Signals. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 744–751, August 2016. ISSN: 2324-9013.

[153] Yao Ge, Shuja Ansari, Amir Abdulghani, Muhammad Ali Imran, and Qammer H. Abbasi. Intelligent Instruction-Based IoT Framework for Smart Home Applications using Speech Recognition. In *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 197–204, August 2020.

[154] Ehsan Nazerfard, Parisa Rashidi, and Diane J. Cook. Discovering Temporal Features and Relations of Activity Patterns. In *2010 IEEE International Conference on Data Mining Workshops*, pages 1069–1075, December 2010. ISSN: 2375-9259.

[155] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering Activities to Recognize and Track in a Smart Environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):527–539, April 2011. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[156] Alli Mäkinen, Jaime Jiménez, and Roberto Morabito. ELIoT: Design of an emulated IoT platform. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–7, October 2017. ISSN: 2166-9589.

[157] Razvan Beuran, Lan Tien Nguyen, Toshiyuki Miyachi, Junya Nakata, Ken-ichi Chinen, Yasuo Tan, and Yoichi Shinoda. QOMB: A Wireless Network Emulation Testbed. In *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pages 1–6, November 2009. ISSN: 1930-529X.

[158] Chuong Dang Le Bao, Nhan Ly Trong, and Quan Le-Trung. UiTiOt: A Container-Based Network Emulation Testbed. In *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing*, pages 161–166, Ho Chi Minh City Vietnam, January 2017. ACM.