Sanity Checks in Smart Home Sensor Streams

Rania Taleb
Université du Québec à Montréal
Montréal, Québec, Canada

Roger Villemaire
Université du Québec à Montréal
Montréal, Québec, Canada

Hubert Kenfack Ngankam
Université de Sherbrooke
Sherbrooke, Québec, Canada

Sébastien Gaboury
Université du Québec à Chicoutimi
Saguenay, Québec, Canada

Sylvain Hallé
Université du Québec à Chicoutimi
Chicoutimi, Québec, Canada

Abstract

The integrity of sensor datasets used in smart home applications is crucial for tasks like activity recognition and automation. We identify common validity issues such as event ordering errors, lifecycle inconsistencies, and data corruption, which are often overlooked but can significantly affect the reliability of analyses. We present a toolbox based on the BeepBeep stream processing library that enables efficient verification of these sanity checks on data streams. Our analysis of several publicly available smart home datasets reveals that most of them violate key assumptions about sensor behavior, emphasizing the need for pre-validation.

CCS Concepts

• **Information systems** → **Data cleaning**; **Data mining**; **Data stream mining**.

Keywords

stream processing, smart homes, sensor logs

ACM Reference Format:
Rania Taleb, Roger Villemaire, Hubert Kenfack Ngankam, Sébastien Gaboury, and Sylvain Hallé. 2025. Sanity Checks in Smart Home Sensor Streams. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3672608.3707789

1 Introduction

The proliferation of smart home technologies has led to an increasing number of sensors embedded within domestic environments, continuously generating vast amounts of data. These sensor streams offer valuable insights into various aspects of smart home operations, including security, energy management, and occupant well-being [4, 13, 15, 15, 22, 29, 33, 42, 44]. All these operations suppose that the smart home platform operates correctly, and that the event logs it generates contain faithful and reliable data. Indeed, feeding algorithms with raw data that contains gaps, corrupted or inconsistent readings could have harmful consequences. For instance, a patient could be prevented from obtaining emergency assistance because their current actions are incorrectly or partially recorded; on the contrary, an occupant could be erroneously diagnosed as declining in health on the grounds of faulty or missing data. In other words, the results produced by the intricate analysis of sensor data proposed in these works is as good as the quality of the incoming data ("garbage in, garbage out").

Yet, there exist multiple reasons why the data fed to these algorithms might not be as faithful a record of the habitat's state and occupant's actions as expected. Device firmware can contain bugs, sensors can break, batteries can run out, hardware clocks can drift, and occupants themselves may accidentally interfere with the proper operation of the monitoring apparatus. However, there is little evidence that research works using smart home datasets perform any kind of pre-validation of the data fed to their algorithms. In most of the aforementioned works, raw data is seemingly fed directly to the proposed algorithms, and whether any checks have been made to ensure that the data was "sane" in the first place is almost never documented. Stated otherwise, there is no evidence that these approaches use a safety net that would perform a first pass of verification to ensure that basic assumptions about the input data are indeed satisfied before being analyzed further.

Yet these assumptions do exist, and are numerous when one makes the effort to make them explicit. Some are trivial and easy to verify, such as the fact that every sensor produces values within a documented range. But others are less so, and what is more, can lead to grossly incorrect results. For example, one may suppose that events are written to the data source in the temporal order of their occurrence, which may then simplify further processing made on this data. An algorithm assuming this and being fed out-of-order events may conclude to errors that actually did not occur. Similarly, it may be assumed that every "open" event emitted by a contact sensor is followed by a "close", which can cause interpretation errors if this is not the case. It can be seen that validating these hypotheses is a stateful process that goes beyond simple data cleaning, as is common practice in Big Data-based techniques.

This paper addresses this problem by providing a three-part contribution. First, after a brief overview of smart home technologies and their more specific application to activity recognition, Section 2 highlights various validity issues that datasets may be subject to, and implicit hypotheses that are typically held about the nature of these datasets. Then, Section 3 proposes, through the extension of an existing event stream processing system, abstract templates corresponding to common processing operations on sensor streams. These templates easily allow the evaluation of numerous "sanity checks" —that is, conditions that, if violated, reveal a potential malfunction at the level of the smart home platform or data collection.

SAC '25, March 31-April 4, 2025, Catania, Italy

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/10.1145/3672608.3707789

Finally, Section 4 presents the results of the analysis of more than half a dozen smart home datasets, with regard to the respect of these sanity checks. Our study ends with a surprising finding, namely that most of the datasets we examined violate most of the sanity conditions we consider.

2 Activity Recognition from Sensor Logs

Smart homes are residential environments equipped with a variety of interconnected sensors designed to collect and monitor data related to activities of daily living (ADL) of residents. These sensors, such as temperature, motion, and pressure sensors, gather information about environmental conditions and resident behaviors. Together, these devices contribute to efficient and responsive living in Ambient Assisted Living (AAL) environments [1, 18].

2.1 ADL Recognition

The data emitted by these sensors form continuous data streams, which are fundamental for real-time monitoring and automated control in smart homes. In the context of Internet of Things (IoT) devices, streams are typically processed continuously, as storing the entire dataset is impractical due to its volume and evolving nature. In this regard, Bifet *et al.* highlight the need for flexible learning models capable of adapting to dynamic, non-stationary data [10]. However, in the context of ADL recognition, streams are very often recorded and stored in the form of logs, since many approaches require the addition of *a posteriori* annotations to identify activities, before these logs can then be used as training data for learning a model. This aligns with the stream data analysis needs in smart home environments [3, 6], where timely insights and predictions are necessary for applications ranging from security to energy management.

Machine learning, clustering, and classification algorithms are commonly used to analyze data streams [9, 11, 12, 24–26, 30, 32, 39]. In smart homes, these algorithms can be used to classify and detect specific activities. For example, Das *et al.* [20] applied classification in the context of activity recognition by using one-class classification to learn normal activity patterns. This machine learning technique is trained on data representing typical, error-free activities performed by individuals. When new, unseen activity patterns are presented, the classifier identifies deviations from the norm, which are classified as activity errors. These errors are used to detect situations where individuals may need intervention or assistance, such as when someone with memory limitations struggles to complete daily tasks.

Some techniques such as decision trees are used by building models that classify human activities [37, 40, 41]. These models learn patterns from the data and attempt to predict which activity is happening based on the input. Other models are built using neural networks, trained using labeled datasets of activities, and used to monitor behavior and detect deviations that may indicate health issues or emergencies [14, 34]. For instance, Chen *et al.* [15] used self-supervised learning in the recognition of ADL through deep neural networks. These models were learned from pre-segmented activity data gathered from smart home environments containing labeled activity data. Support vector machines can also be used for these tasks [23, 28, 38]. Similarly, clustering can employed for ADL recognition; Akl *et al.* [2] applied k-means clustering to categorize room activity distributions into three groups: Cognitively Intact (CIN), Transition (TR), and MCI. To enhance accuracy, Affinity propagation was used to create exemplars representing each cluster.

The use of a knowledge-driven approach, supported by ontologies, sometimes helps in improving the recognition of activities in smart homes by providing structured representations of context and actions. Chen *et al.* [16] applied this approach by constructing ontologies to model both the smart home environment and ADLs. These ontologies capture the relationships between various objects, locations, and activities, allowing the system to interpret sensor data in a semantically meaningful way. Through subsumption reasoning, the system compares real-time sensor data against predefined activity models, enabling it to infer and recognize user activities accurately and adapt to individual user behaviors.

Finally, in a last category, Demongivert *et al.* [21] developed a distributed event-oriented architecture for activity recognition that employs autonomous agents communicating via event-based messaging. This system distributes computational tasks across multiple locations, and processes data in real-time, allowing it to track several smart homes simultaneously.

2.2 Existing Platforms and Their Uses

Multiple research teams and labs set up experimental platforms to collect datasets for activity recognition and smart home research. We provide a brief overview of some prominent research platforms, as well as the various research works that used the data produced by them and made publicly available.

CASAS [17] offers a platform that deploys sensor networks in homes to monitor daily activities such as sleeping, preparing meals and eating for aging populations. Using data from motion, temperature, and door sensors, they train machine learning models for activity recognition. CASAS also provides open-source datasets that are widely utilized in research on health monitoring, behavior analysis, and smart home assistance.

NEARS-Hub [36] focuses on lightweight edge computing for real-time monitoring in smart environments. Its goal is to create a framework that processes data locally in smart homes, particularly for elderly or health-monitored individuals. By capturing data from sensors and using edge computing, NEARS-Hub enables real-time decision-making and assistance without relying on a cloud infrastructure.

Amiqual4Home (A4H) [19] develops smart home testbeds that mimic real-life environments to study human interaction with ambient intelligence systems. Equipped with various sensors and devices, A4H aims to automate tasks and adapt to user needs while capturing detailed data on daily activities such as cooking, cleaning, and leisure. The research focuses on enhancing user comfort, energy efficiency, and health monitoring.

Smart Environments at Home and Elsewhere (SEArch) [5] develops smart environments aimed to improve the quality of life for elderly and disabled residents. Motion detectors and environmental sensors are deployed. The goal of this platform is to use sensor networks to monitor health and predict expected issues. Their datasets are used to design machine learning algorithms to detect abnormal behavior patterns in real time.

Other projects aim to develop smart home solutions specifically for elderly care and health monitoring. We mention the *Smart** project [7], which gathers data on energy consumption and appliance usage patterns in smart homes, with the aim of improving home automation and energy efficiency, and the PlaceLab [31], a highly instrumented home environment used to monitor a wide range of activities and human behaviors.

Some of the datasets produced by these platforms have been put to multiple uses, particularly in the field of e-Health. The so-called "Aruba" dataset from CASAS has been the subject of a large number of studies, principally focused on the recognition of activities of daily living. Fahad *et al.* [22] proposed a robust activity recognition system using the Aruba dataset to identify daily living activities where they noted that the activity "Resperate" was not correctly recognized by either method due to the limited number of occurrences, making it difficult to identify. Huang *et al.* [29] excluded this activity and focused on human activity recognition experiment involving the remaining 10 activities. They stated that the dataset is imbalanced, as some of the activities occur more frequently than others and around 54% of the entire sensor events have missing labels. In their part, Yala *et al.* [35] revealed that no method could identify the 'Dish Washing' activity, where most of its test instances are identified as 'Meal Preparation' activity. This is because the two activities run in the same location and trigger the same sensors.

Other CASAS datasets have been the focus of numerous studies as well. Tan *et al.* [42] introduced the front-door events classification algorithm. They proposed the brief-return-and-exit (BRE) event and studied how much a resident is active in terms of the number of exits per month. Aminikhanghahi *et al.* [4] focused on enhancing activity recognition by segmenting behavior-based sensor data, specifically using a Change Point Detection (CPD)-based activity segmentation method. The main idea was to improve the accuracy of activity recognition in smart homes by identifying the start and end points of activities in real-time, which provides clearer boundaries for recognizing activities. They discovered that some activities such as "Enter Home," "Leave Home," and "Bed Toilet Transition," are generally short and involve similar movements to other activities, making segmentation less effective. Bouchabou *et al.* [13] emphasized the significance of sensor placement to accurately capture activities. For instance, if a motion sensor is installed above a bed to detect its usage, but the bed is later moved to a different spot in the room, the sensor will no longer capture the relevant data. This limitation is crucial in real-world applications, where algorithms need to be resilient and continue functioning effectively even when information is lost due to such changes.

The Orange4Home dataset has been used by Song *et al.* [43] who propose a cognitive model for ADL. They observed that the regularity in the routines of this dataset influences prediction accuracy, leading to generally higher performance due to the consistent activity patterns. However, this suggests that the dataset may not fully capture more complex and irregular behaviors.

2.3 Validity Issues in Smart Home Datasets

It is evident that data stream quality has a profound impact on the performance of employed learning models. Addressing this challenge is critical for improving the robustness of machine learning systems, particularly in ADL recognition. Yet, there are multiple factors that can result in the data produced by a smart home platform to present issues regarding its validity. For instance:

- Heterogeneity: Events in smart home datasets are stored in multiple incompatible formats, often with different conventions even within the same format (e.g., CSV). This hinders the reusability of analysis scripts and discourages thorough data validation.
- Event Ordering: Sensor logs are often assumed to be time-ordered, but factors like race conditions, transmission delays, or batch reporting can violate this assumption. Analyses relying on strict ordering risk producing incorrect results.
- Data Corruption: Issues like malformed names or outlier values may indicate corrupted data. Smart home datasets typically lack error correction mechanisms, making corruption detection challenging.
- Low Battery: Sensors with low battery levels may produce inaccurate readings or encounter transmission errors. Logs containing such intervals should be interpreted cautiously.
- Outages: Gaps in the data stream can result from sensor malfunctions, power loss, or user interference. Mechanisms like ZigBee radio check-ins can help detect such outages.
- Outliers: Platform malfunctions may produce outliers in sensor data. While extensively studied, outliers in this context are framed as one aspect of broader validity issues.
- Lifecycle Issues: Stateful sensors (e.g., open/close sensors) may produce invalid event patterns (e.g., consecutive "open" events) or miss intermediate events, indicating potential malfunctions.
- Annotation Issues: Annotated datasets assume that each moment in time corresponds to at most one activity. Overlapping or missing labels raise concerns about labeling quality.
- Abnormal Behavior: Patterns in the logs that deviate from expected platform behavior (e.g., simultaneous motion detection in all rooms) suggest platform issues, distinct from erratic occupant behavior.

3 Generic Patterns Applied to Sensors Datasets

There is little evidence that the works using datasets presented in Section 2.2 perform verifications for the various issues raised in Section 2.3. Reasons for such a lack of sanity checks are multiple: little perceived added value in a research environment with a pressure for new results, low estimated occurrence of such problems in the datasets. We also suspect such checks are not performed because no fast and easy way of specifying and evaluating them on a dataset exists. In order to uncover evidence of possibly invalid log data, one must have at their disposition a set of tools allowing the querying and manipulation of event logs in a flexible and format-agnostic way.

In this section, we present a set of event stream processing templates expressed at a high level of abstraction. Among the numerous stream processing platforms and systems available, we elected to express and implement these tasks in the idiom of the BeepBeep stream processing engine [27]. Arguments in favor of this system include the fact that it has a a formally defined operational semantics [8], has been under active development for a decade, is

available under the form of an open source Java library, and is easily extensible.

Conceptually, the tool provides a set of computation units called *processors*. Each processor ingests one or more streams of events, and produces one or more streams according to a calculation that is specific to each type. BeepBeep obviously operates in streaming fashion, meaning that each new input event fed to a processor triggers the calculation associated to that processor, and the new output events resulting from this calculation (if any) are made available for consumption immediately.

Processors can be connected or "piped" together to form directed graphs called *pipelines*. A pipeline is created by taking the output of a processor and feeding it as one of the inputs of another processor. Depending on the actual processor instances used in the pipeline and the way they are arranged in a graph, complex (and more importantly, *stateful*) calculations can be obtained. The basic processors provided by BeepBeep allow events to be decimated, filtered, sliced across multiple sub-streams, or aggregated on sliding windows. The elementary processors are very flexible and often offer greater expressiveness than the operators of the same name available in other stream processing systems. The core of BeepBeep lists close to 90 distinct processors, to which can be added dozens more in domain-specific extensions called *palettes*. Due to lack of space and the fact that these topics have been amply covered in past literature, the reader is referred to a recent textbook on the topic for more details about the platform [27].

3.1 Format Abstraction

A first step towards developing a toolbox for easy evaluation of validity conditions on a dataset is to abstract from the heterogeneity of the various datasets. Datasets use differing formats, both in the type of events recorded (XML, JSON, CSV) and in the way data is recorded as attributes. There is a need for a mechanism that reconciles these formats and presents a uniform higher-level view of events. A first part of our contribution is to make a synthesis of multiple popular smart home datasets, identify the common key concepts and provide an abstraction layer allowing these datasets to be processed in a format-agnostic way.

The approach we propose is to define an abstract interface exposing a number of objects and functions that provide access to different attributes of an event. Its purpose is to manipulate sensor events at a level of abstraction that hides the differences in the file formats across multiple datasets. According to our analysis of the datasets described in Section 2.2, we observe that, under various arrangements, events have the following common features:

- A *state*, which corresponds to a single value or reading produced by a sensor at a given moment
- A *timestamp* that determines the time at which a specific state reading has been produced
- An *index*, which corresponds to the position of the event in the physical ordering of the input source
- Two attributes specifying the placement of the sensor: the *location*, which is the broad area (e.g. room) where the sensor is located, and the *subject*, which is the particular element (e.g. bedhead, stove, door) that the sensor is observing
- A *model*, which is the physical device producing the reading
- The *name* of the sensor that produces the reading

It is further assumed that these last four attributes are nested within each other (i.e. for a given location, the value of subject uniquely defines a subject; for a given location-subject pair, the value of model uniquely defines a physical device, and so on).

Note, however, that the datasets considered concretize these different characteristics in various ways. Thus, in the Aruba dataset, each sensor has its own ID, which is part of the attributes contained in an event –it can therefore be queried directly. In contrast, the NEARS platform explicitly provides the quadruplet location-subject-model-sensor, but not an explicit ID for each sensor. In this case, it is therefore the quadruplet itself that is used as ID. In Orange4Home, all these elements are on the contrary agglomerated in a single attribute and separated by underscores; moreover, the same part in a name may sometimes refer to a subject, and sometimes to a model, depending on the actual sensor. The functions returning the various characteristics of an event therefore all read the same field, but extract different parts according to a regular expression to follow the dataset's unusual nomenclature.

Similarly, datasets use different conventions to represent the timestamp (which is standardized in Unix epoch regardless of its input format), and similarly for symbolic values produced by some sensors (for example the "open" and "closed" states of a contact sensor). Ultimately though, providing support for a given dataset or platform therefore boils down to defining the handful of BeepBeep `Function` objects that extract these various features. A processing pipeline can then be written in a way that disregards the type of the actual event source, by simply invoking functions of the appropriate event format object. This seemingly innocuous contribution actually opens the way to reusing the same analysis script across heterogeneous datasets.

3.2 Processing Templates

In order to detect the different situations where the validity of a dataset could be compromised, a user could write BeepBeep processor chains corresponding to each of the cases discussed above. However, we quickly notice that many of these operations involve a small number of patterns, which recur with some variations in many of these pipelines. A contribution of this article is therefore to propose parameterizable pipeline patterns, synthesizing the common processing performed in sensor log analysis, and allowing the user to define computations at a high level of abstraction. Some of these templates are graphically represented in Figure 1. We discuss them in the following.

Filter: this template takes as a parameter a pipeline P producing a stream of Boolean values. It outputs a stream made of all the events of $\overline{\sigma}$ at indices i for which P produces the value true ($\top$). If P represents a condition on the input stream, then this template outputs only events satisfying P and discards the others. For example, P could evaluate the function checking that an event's value is either "open" or "close", thus keeping from the log only data about contact sensors. Note, however, that P can also evaluate a stateful condition on multiple events; for instance, one could only keep events that are immediately followed by two more events with the same value.

Successive: this template takes as a parameter a binary function f_Δ. It produces a stream of values obtained by evaluating $f_\Delta(x, y)$

on every pair of successive events x, y in the input stream $\overline{\sigma}$. This template is useful to calculate durations. For example, to identify gaps in a log, one can calculate the timestamp difference between two successive events.

Locate: when a condition on a stream is violated, or a potentially suspicious pattern is found in an event stream, it is often desirable to manually inspect the log or isolate the appropriate portion for additional analysis. This template takes as a parameter a 1:1 pipeline P producing a stream of Boolean values. It outputs a stream made of all the indices in $\overline{\sigma}$ for which P produces the value true ($\top$). This makes it very easy for a user to further investigate a potential issue: if a pipeline P indicates a violation, $Locate(\neg P)$ will directly identify the positions in the log where this violation can be found.

Count: this template takes as a parameter a 1:1 pipeline P producing a stream of Boolean values. It outputs a stream of numerical values, where the value at index i is the number of times P has produced the value true ($\top$) up to index i. For example, calculating the number of times a door has been open during a day is done by instantiating the *Count* template.

Threshold: a pattern taking a function f, an integer m and a numerical value t. It produces a stream of Boolean values; output event at index i is $\top$ if event at index i in the input is the first of a sequence of at least m successive events, such that applying $f(x)$ to each of them produces a value that exceeds some threshold t. This pattern is "smart" in the sense that it produces a single value $\top$ at the start of a sequence of contiguous events exceeding the threshold, and not a $\top$ value for every window of m events exceeding the threshold.

SliceBy: leverages BeepBeep's *Slice* processor. A function f is used to dispatch events into sub-streams (one per value produced by f). A distinct instance of processor P is run on each sub-stream. Once the end of the input stream is reached, the resulting values of all instances of P are then "played back" as the output stream.

Characterize: a descriptive pattern that helps get an overview of a set numerical values. It takes as a parameter a processor P, which extracts from the input log a stream of numerical values. The pattern then provides basic descriptive statistics about that stream of values, using the *BoxAndWhiskers* function calculating the various quartiles over which the data is distributed.

Follows: another descriptive pattern taking as input a processor P. The output of P is a stream of arbitrary values, which are then collected into bigrams made of all pairs of successive events. These bigrams are interpreted as edges in a weighted directed graph, where a bigram (x, y) indicates a directed edge between vertices x and y. Multiple occurrences of a bigram result in a higher weight.

Following the spirit of the BeepBeep system, these templates can naturally be composed and nested. This means not only that the output of one template can be connected to the input of another, but also that parameter P of a given template, when expecting a processor, can be instantiated by setting P to the instance of another template —or any other chain of processors. Thus, the *Characterize* pattern may not be applied only on raw numerical values fetched from events, but can also be applied to a stream of values produced by a complex chain of processors. Thus, one can calculate the elapsed time between each opening and closing of a door, and pipe this into *Characterize* to get an overview of the underlying numerical values. This flexibility allows the user to

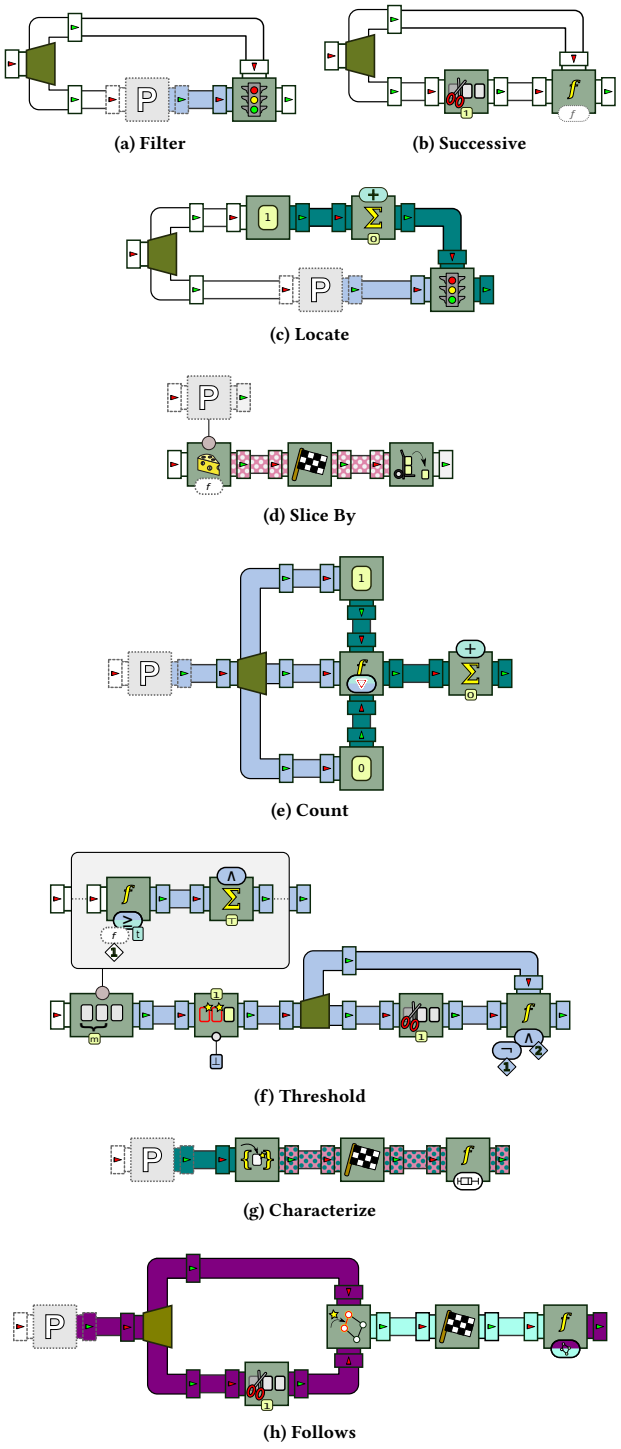


Figure 1: Generic patterns for processing sensor streams in smart homes.

Family	Instance	Ref.	Events	Format	Ann.
NEARS, Université de Sherbrooke	0032	[36]	186,532	JSON	N
	0034	[36]	382,268	JSON	N
	0102	[36]	689,842	JSON	N
	0104	[36]	461,980	JSON	N
	0105	[36]	385,772	JSON	N
CASAS, Washington State University	Aruba 1	[17]	1,719,558	CSV	Y
	Aruba 2	[17]	3,509,096	CSV	N
	HH115	[17]	2,240,010	CSV	Y
	HH130	[17]	1,156,820	CSV	Y
Amiqual4Home, Inria	Orange4Home	[19]	746,768	CSV	Y

Table 1: The sensor logs included in our analysis.

quickly create complex calculations, without having to create the low-level pipeline from scratch.

Concretely, these patterns are bundled in the form of a "toolbox" that extends the BeepBeep library with these high-level objects. A user can create a pipeline by directly instantiating and connecting the desired processors inside a Java program; but more conveniently, the same processor chains can be obtained through simple shell scripts in the Groovy language.

4 Experimental Evaluation

Considering the validity issues raised in Section 2, and equipped with the high-level stream processing templates we defined in Section 3, we will now attempt to determine whether the logs provided by various research teams satisfy the minimal "sanity checks" we discussed earlier. To this end, we implemented BeepBeep pipelines to reveal the presence of violations of these assumptions, and set out to evaluate them on a set of publicly available sensor logs that have been widely used in the past. All the pipelines are publicly available online[1], while the datasets themselves can be retrieved from the respective research teams.

Table 1 presents a summary of the basic characteristics of the datasets we surveyed. They are taken from three independent research teams, and total more than 10 million events in either JSON or CSV format. Some of them contain annotations about ongoing activities done by the resident. All of them are concerned with a single person living in the smart home (hence not a multi-resident situation), with the possibility of occasional visitors. At the outset, we disclose our primary finding, namely that, surprisingly, most of the conditions considered are not met by the majority of the datasets studied (cf. Table 2).

Let us emphasize that the aim of this experiment is not to cast blame on the teams who collected this data. Setting up a smart home test environment, gathering data (especially with real participants), and documenting a dataset is a considerable task, and it is to the community's benefit that these data are made available. Rather, we observe that it is challenging for any dataset to be entirely free of failures, errors, and inconsistencies.

The following sections revisit the validity issues from the beginning of the paper, and describe the problems that the analysis pipelines allowed us to –easily– uncover.

4.1 Formatting Issues

At the lowest level of abstraction, we can report on issues related to the physical formatting of data in the files provided in each dataset.

[1] https://github.com/sylvainhalle/beepbeep-sensor-events

Instance	Low battery	Stuttering temp.	ZigBee heartbeats	Valid format	No swapped events	Lifecycle	Door episodes	No gaps	Suspicious ranges	Simultaneous presence	Zero value	Undersupported activity	No nested activities
0032	✓	–	–	✓	×	×	✓	×	×	×	–	–	–
0034	✓	–	–	✓	×	×	×	×	✓	×	–	–	–
0102	×	–	–	✓	×	×	✓	×	✓	×	–	–	–
0104	×	–	–	✓	×	×	✓	×	×	×	–	–	–
0105	×	–	–	✓	×	×	×	×	×	×	–	–	–
Aruba 1	–	×	–	×	✓	×	×	×	×	×	×	×	×
Aruba 2	–	×	–	×	✓	×	×	×	~	×	✓	–	–
HH115	×	×	–	×	✓	×	×	×	~	✓	×	✓	–
HH130	×	×	×	×	✓	×	×	×	✓	✓	✓	✓	–
O4H	–	–	–	✓	✓	×	×	×	×	×	✓	✓	✓

Legend: ✓: satisfied; ~ violated, but plausible; ×: violated; –: not applicable.

Table 2: Summary of sanity checks on datasets.

We admit that this simplistic check is not the cornerstone of our toolbox, and that these issues can usually be fixed with existing data cleaning tools.

4.1.1 Uniformity. The HH115 dataset contains two lines (out of 1.7M events) whose timestamp parameter has a different formatting from the others (missing digits for milliseconds) which can cause parsing exceptions when using, e.g. the `SimpleDateFormat` utility class in Java. The Aruba 1 dataset presents the same timestamp formatting issues. Moreover, the file starts with a space separating fields in each line, and after about 550,000 events, switches to tabs as the separator and remains so until the end. More interestingly, motion sensors at some point intermittently emit incorrectly formatted values, with the expected "`ON`" and "`OFF`" replaced by strings like "`ONc`", "`OFF5cc`" or "`OFcF`" for a period of approximately 10 hours. Finally, some lines have incorrect values in the column standing for a sensor's name. A single line (number 1523045) has `LEAVEHOME` as the sensor name and `180` as its value; a single other line (1530062) rather has `ENTERHOME` and 6592. This obviously does not match any documented sensor.

4.1.2 Event Ordering. In the NEARS dataset, we discovered that events are not physically written in the file in the order dictated by their timestamps. In the case of small time differences between mis-ordered events, one could suggest as a possible cause either slight synchronization issues between each device's clock, or delays in the transmission of events causing them to be mildly shuffled. However, in the case of NEARS, some events have a timestamp that deviates from their neighbors by several hours. Desynchronization of clocks must also be ruled out, since those shuffling issues occur even when considering the events produced by a single sensor in isolation. We can only speculate about the precise cause of these re-orderings, and whether they warrant considering timestamp data with caution.

4.2 Incomplete Information

A second type of issue is related to the gaps that can be found in the recorded data, which indicates either a general loss of the sensor platform, or a failure of the recording infrastructure to properly save the events generated by the sensors.

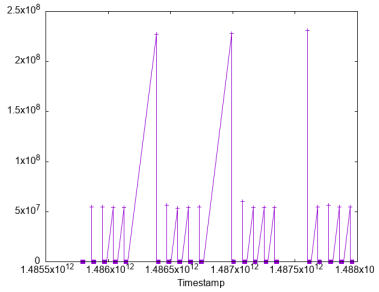


Figure 2: Time since the latest sensor update for Orange4Home dataset. Spikes indicate extended periods of time where no sensor produced any event.

4.2.1 Temporal Gaps. All the datasets we considered contain extended periods of time where no event is generated from any of the documented sensors. For instance, the NEARS dataset contains a gap of almost one day where no sensor produced any event. Additional gaps can be found when looking at the event rates produced by individual sensors (e.g. *Coffee pot* which goes silent after 23 days and remains so forever). Similarly, the Orange4Home dataset has three intervals of approximately 69 hours where no sensor reports any data; it also contains several more periods where no data is reported for approximately 13 hours (see Figure 2). On its side, the HH115 dataset contains a gap of 6 *days* where no data is recorded. Although the dataset comes with a detailed documentation file, it does not warn the user of this large gap in the data.

4.2.2 ZigBee Heartbeats. The HH130 dataset is the only one to include ZigBee devices which, according to its documentation, are expected to make a radio check-in at predefined intervals (once every 15 or 30 minutes, depending on the device). We found that all ZigBee devices listed in the dataset violate the radio check-in condition, which was defined as failing to send a message for more than 60 minutes.

4.2.3 Other Potential Gaps. In the HH115 dataset, temperature sensors are expected to send a new event only when the temperature changes. Yet there are 22 occurrences (across 6 different sensors) that report two successive temperature readings with the same value. This indicates either that the sensors do not behave as reported in the documentation, or that events between the two successive temperature readings, with a different value, have not been recorded. At other times, sensors mentioned in the dataset documentation do not appear in the log. This is the case again in HH115, where the sensor types `Control4-Radio` and `Control4-Button` and the corresponding sensors `BT001` to `BT008` do not produce a single event in the whole dataset. Note that in reverse, in the HH130 dataset, we found sensors in the log that are not mentioned in the documentation.

4.3 Suspicious Ranges

Without performing full-fledged outlier analysis on a stream of numerical data, one can still devise basic rules of thumb to spot wildly aberrant sensor readings by examining their range of values. For instance, by stipulating that temperature should reasonably be comprised between 0°C and 40°C, we were able to spot a single abnormal value in excess of 200°C for three different sensors in the Aruba 1 dataset. Examining surrounding values of those same sensors, one can suspect that this is caused by a missing decimal period, which would frame it as a formatting issue instead of a true outlier. Still, it is worth wondering why only these three values suffer from this problem, especially when the file is generated (as is supposed) automatically. Similarly in the NEARS 0105 dataset, the oven temperature sensor sometimes reports values all the way down to −65°C. This reading is part of an interval where oven temperature fluctuates between freezing and room temperatures over the course of about 15 minutes.

In contrast, the HH115 dataset also has a sensor whose temperature readings are in the interval $[40, 55]$. However, in that case, the sensor map shows it is located over the stove range, which makes these values plausible. This is a prime example of the fact that the violation of a sanity check is not *necessarily* indicative of an error, only that further inspection is warranted.

The Orange4Home dataset reports 0 as the power consumption for all appliances in the kitchen throughout the whole log, which would suggest they are never used. This is despite the fact that both voltage and current for these appliances is non-null. The problem can be found in reverse in the NEARS 0032 and 0104 datasets: some appliances report a non-null power consumption despite their voltage and/or current being at 0 in the entire log.

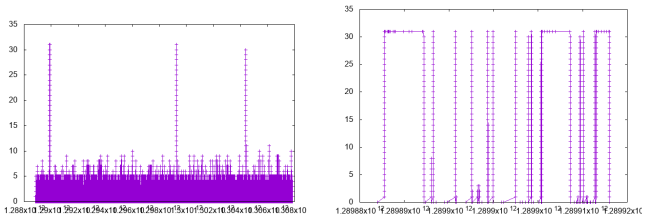
4.4 Lifecycle Issues

The datasets we studied also present issues related to sensors exhibiting a lifecycle behavior.

4.4.1 Invalid Transitions. In the NEARS dataset, almost all such sensors violate their lifecycle because of the aforementioned gap in the data (causing, for example, the occurrence of two successive "close" events on each side of the gap). However, outside of gaps, all datasets reported events violating the lifecycle of a door or a switch during what appears to be periods of normal operation. We noted 36 invalid transitions out of 17,825 contact events in the NEARS dataset which could not be explained. In the Orange4Home dataset, the on/off lifecycle is violated 14 times out of about 41,000 toggle events. Similarly in the Aruba dataset, almost all motion and door contact sensors violate the on/off or open/closed cycle at least a few times; there is a total of 128 violations out of 1.6 million events.

4.4.2 Suspicious Intervals. In addition to unexpected on/off or open/close transitions, we can also report sensors reporting a state of an object for either an unusually long, or an unusually short period of time. In the NEARS dataset, a drawer in the kitchen has been open on June 17th and seemingly remained open until the end (that is, for multiple weeks). Similarly, the main door contact sensor remained in the "open" state for 35 hours in one apartment, and 18 days for another. On its side, in the HH115 dataset, the bathroom door seemingly remained open for 22 days, while the fridge door remained open for two months. This is the case, despite the fact that the rest of the platform apparently operates normally.

In reverse, a cycling of sensor states at short intervals may also indicate an issue. In the NEARS dataset, we found an interval of time where the front door opens and closes multiple times per second over the course of several seconds. One could assume that



(a) Complete log (b) Zoom in

Figure 3: The "glissando" phenomenon discovered in the Aruba 1 sensor log. Figure (a) shows the entire log, and figure (b) shows the detail of the first spike.

this is a situation where the door slams (for example, due to the wind), but it would be unwise to conclude from this data that a person repeatedly operated the door.

4.5 Simultaneous Presence

In general, it is not suspicious for several motion sensors to report movement simultaneously, as their coverage areas overlap and these sensors exhibit a certain latency. However, the situation becomes more curious when a large number of these sensors report movement at the same time, especially knowing that the habitats studied are inhabited by a single person.

4.5.1 Sporadic Simultaneous Presence. Surprisingly, we discovered that most datasets contain short time spans where movement is reported in multiple rooms of the apartment at the same time. In the case of the Orange4Home dataset, this situation occurs a single time over the whole log, where motion is reported by sensors in two rooms on two different floors. In contrast, in the NEARS dataset, we witness multiple occasions where motion is reported in *all* rooms simultaneously. This can hardly be explained by the presence of a visitor in the apartment.

4.5.2 Aruba Glissando. Perhaps the most unexpected phenomenon can be found in the Aruba dataset. Analysis of the motion sensors, and in particular the number of these sensors simultaneously reporting motion at a given point in time, revealed a surprising behavior. At a few points in the event stream, the sensors transition to the ON state (indicating that motion is detected) one after the other, and remain ON, until all the sensors in the apartment are ON simultaneously. The sensors then switch back to the OFF state, again one after the other, until they are all OFF again. This kind of "glissando" sometimes repeats itself immediately a few times, after which the sensors resume what appears to be normal behavior. Figure 3a shows how this phenomenon manifests itself in the form of spikes where all 31 sensors in the apartment are ON at once. Figure 3b zooms in on the period of time covering the first of this phenomenon. What appears as a single spike is actually a sequence of multiple intervals where all sensors are ON at the same time.

As one can see, in the Aruba 1 dataset, there are three such episodes (over an 8-month period), while in the Aruba 2 dataset there are six (over 13 months); they occur spontaneously, with no apparent regularity, and at any time of the day. One can only speculate on the precise nature of this phenomenon, but it is absolutely unlikely that these readings are produced by the activity of a human being (who would then be moving simultaneously throughout the *entire* apartment). However, the Aruba dataset documentation does not explain these unusual passages in the flow of events, nor does it even warn users of the presence of these passages. Moreover, the Aruba 1 dataset curiously associates two of these intervals with an activity: the first with *Sleeping*, and the third with *Relaxing*.

4.6 Annotation Issues

The following issues only apply on datasets that contain annotations about activities performed by the occupant.

4.6.1 Overlapping Activities. In some datasets, intervals of time where an activity takes place are indicated by "begin" and "end" markers throughout the log; in our study, this is the case for Aruba and Orange4Home. However, such a way of labeling the dataset opens the possibility for activities to be nested or to overlap (i.e. a "begin" marker for an activity is observed without the "end" marker of the previous activity being observed first).

This phenomenon does not occur in the Orange4Home log. However, the Aruba 1 dataset contains 36 occurrences of such overlapping. For example, the *Relax* activity very often encompasses a begin/end pair of the *Eating* activity. The *Housekeeping* activity contains occurrences of *Leave Home* and *Enter home* begin/end; *Meal Preparation* overlaps several times with either *Eating* or *Relax*. On one particular occurrence, the begin/end pair of the *Meal Preparation* activity spans almost 6 hours, during which several other activities (*Relax*, *Enter*, *Leave Home*) take place.

4.6.2 Under-Supported Activities. One could describe some sensors as "passive" in that they report the state of the apartment at regular intervals (regular readings of temperature, electricity consumption) and generate data independently of activity—or even the presence—of an individual. On the other hand, all the datasets studied also contain "active" sensors, i.e., sensors generating events following a concrete action taken by a resident; for example, the opening of a door, the activation of a switch, or simply the movement of an individual in a room.

In this context, and without presuming anything about the behavior or habits of a resident, it is reasonable to assume that a range of sensor data associated with an explicitly labeled activity is supported by the presence of events induced by a person's action, via active sensors. One might question the labeling in the case of an activity that contains, for instance, only temperature readings. However, there is indeed an occurrence in the Aruba 1 dataset of an activity during which no motion or door sensor is involved.

5 Conclusion and Future Directions

In conclusion, to derive valuable insights in smart home environments, it is essential to ensure the integrity of sensor data. Our toolbox, built on the BeepBeep stream processing library, provides a comprehensive solution for detecting potentially erroneous sensor behavior and verifying data quality before analysis. By addressing the gaps in existing methodologies, this toolbox enhances the reliability of smart home applications, ensuring that decisions made from sensor data are based on accurate and consistent information.

Without being alarmist, we can still question the consequences of the fact that the datasets we studied all present several violations of simple validity conditions. We have seen that many studies have used these datasets and draw various conclusions, whether in activity recognition or for other purposes. Whether these potential "problems" have been identified and managed by the different researchers who have used these data is unclear, and could represent an issue from a methodological standpoint.

This work also lends itself to several extensions. Thus, by virtue of the functioning of the BeepBeep system, all sanity checks can be evaluated not only on pre-recorded logs, but also in real time on streams of events produced as they occur. It would therefore be possible to imagine the implementation of alarms to warn managers of potential problems with the functioning of the smart home platform (and thus avoid, for example, gaps in the data of several days and other aberrations). Finally, the *Locate* template could be used to mark the events of a log involved in the violation of one of the sanity checks considered. This elementary form of traceability would make it possible to indicate to users that the result of a calculation should be considered with caution, because it consumes such flagged events.

References

[1] G. Acampora, D. J. Cook, P. Rashidi, and A. V. Vasilakos. A survey on ambient intelligence in healthcare. *Proc. IEEE*, 101(12):2470–2494, 2013.

[2] A. Akl, B. Chikhaoui, N. Mattek, J. A. Kaye, D. Austin, and A. Mihailidis. Clustering home activity distributions for automatic detection of mild cognitive impairment in older adults. *J. Ambient Intell. Smart Environ.*, 8(4):437–451, 2016.

[3] M. M. Alam, L. Torgo, and A. Bifet. A survey on spatio-temporal data analytics systems. *ACM Comp. Surv.*, 54(10s):219:1–219:38, 2022.

[4] S. Aminikhanghahi and D. J. Cook. Enhancing activity recognition using cpd-based activity segmentation. *Pervasive Mob. Comput.*, 53:75–89, 2019.

[5] J. C. Augusto, J. G. Gimenez-Manuel, M. Quinde, C. L. Oguego, S. M. M. Ali, and C. James-Reynolds. A smart environments architecture (Search). *Appl. Artif. Intell.*, 34(2):155–186, 2020.

[6] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, and S. Maniu. Data stream analysis: Foundations, major tasks and tools. *WIREs Data Mining Knowl. Discov.*, 11(3), 2021.

[7] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. Smart*: An open data set and tools for enabling research in sustainable homes. *Proc. SustKDD.*, 01 2012.

[8] A. Bédard and S. Hallé. Formal verification for event stream processing: Model checking of BeepBeep stream processing pipelines. *Inf. and Comput.*, 293:105058, 2023.

[9] A. Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, volume 207 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

[10] A. Bifet and J. Gama. IoT data stream analytics. *Ann. des Télécommunications*, 75(9-10):491–492, 2020.

[11] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. In M. J. Zaki, J. X. Yu, B. Ravindran, and V. Pudi, editors, *PAKDD*, volume 6119 of *LNCS*, pages 299–310. Springer, 2010.

[12] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl. MOA: massive online analysis, a framework for stream classification and clustering. In T. Diethe, N. Cristianini, and J. Shawe-Taylor, editors, *WAPA*, volume 11 of *JMLR Proceedings*, pages 44–50. JMLR.org, 2010.

[13] D. Bouchabou, S. M. Nguyen, C. Lohr, B. Leduc, and I. Kanellos. A survey of human activity recognition in smart homes based on IoT sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. *Sensors*, 21(18):6037, 2021.

[14] A. F. Cavalcante, V. H. de Lima Kunst, T. de Menezes Chaves, J. D. T. de Souza, I. M. Ribeiro, J. P. Quintino, F. Q. B. da Silva, A. L. M. Santos, V. Teichrieb, and A. E. F. D. Gama. Deep learning in the recognition of activities of daily living using smartwatch data. *Sensors*, 23(17):7493, 2023.

[15] H. Chen, C. Gouin-Vallerand, K. Bouchard, S. Gaboury, M. Couture, N. Bier, and S. Giroux. Enhancing human activity recognition in smart homes with self-supervised learning and self-attention. *Sensors*, 24(3):884, 2024.

[16] L. Chen, C. D. Nugent, and H. Wang. A knowledge-driven approach to activity recognition in smart homes. *IEEE Trans. Knowl. and Data Eng.*, 24(6):961–974, 2012.

[17] D. J. Cook. Learning setting-generalized activity models for smart spaces. *IEEE Intell. Syst.*, 27(1):32–38, 2012.

[18] D. J. Cook, J. C. Augusto, and V. R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive Mob. Comput.*, 5(4):277–298, 2009.

[19] J. Cumin, G. Lefebvre, F. Ramparany, and J. L. Crowley. A dataset of routine daily activities in an instrumented home. In S. F. Ochoa, P. Singh, and J. Bravo, editors, *UCAmI*, volume 10586 of *LNCS*, pages 413–425. Springer, 2017.

[20] B. Das, D. J. Cook, N. C. Krishnan, and M. Schmitter-Edgecombe. One-class classification-based real-time activity error detection in smart homes. *IEEE J. Sel. Top. Signal Process.*, 10(5):914–923, 2016.

[21] C. Demongivert, K. Bouchard, S. Gaboury, B. Bouchard, M. Lussier, M. Parenteau, C. Laliberté, M. Couture, N. Bier, and S. Giroux. A distributable event-oriented architecture for activity recognition in smart homes. *J. Reliab. Intell. Environ.*, 7(3):215–231, 2021.

[22] L. G. Fahad, S. F. Tahir, and M. Rajarajan. Activity recognition in smart homes using clustering based classification. In *ICPR*, pages 1348–1353. IEEE, 2014.

[23] A. Fleury, M. Vacher, and N. Noury. Svm-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results. *IEEE Trans. Inf. Technol. Biomed.*, 14(2):274–283, 2010.

[24] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet. A survey on ensemble learning for data stream classification. *ACM Comp. Surv.*, 50(2):23:1–23:36, 2017.

[25] H. M. Gomes and A. Bifet. Practical machine learning for streaming data. In R. Baeza-Yates and F. Bonchi, editors, *KDD*, pages 6418–6419. ACM, 2024.

[26] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *SIGKDD Explor.*, 21(2):6–22, 2019.

[27] S. Hallé. *Event Stream Processing With BeepBeep 3: Log Crunching and Analysis Made Easy*. Presses de l'Université du Québec, 2018.

[28] Y. Hu, B. Wang, Y. Sun, J. An, and Z. Wang. Genetic algorithm-optimized support vector machine for real-time activity recognition in health smart home. *Int. J. Distributed Sens. Networks*, 16(11):155014772097151, 2020.

[29] X. Huang and S. Zhang. Human activity recognition based on transformer in smart home. In *CACML*, pages 520–525. ACM, 2023.

[30] D. Ienco, A. Bifet, I. Zliobaite, and B. Pfahringer. Clustering based active learning for evolving data streams. In J. Fürnkranz, E. Hüllermeier, and T. Higuchi, editors, *DS 2013*, volume 8140 of *LNCS*, pages 79–93. Springer, 2013.

[31] S. Intille, K. Larson, J. Beaudin, E. Tapia, P. Kaushik, J. Nawyn, and T. McLeish. The placelab: a live-in laboratory for pervasive computing research (video). In *PERVASIVE 2005 Video Program, Online, May 2005*, 12 2010.

[32] H. Kremer, P. Kranen, T. Jansen, T. Seidl, A. Bifet, G. Holmes, and B. Pfahringer. An effective evaluation measure for clustering on evolving data streams. In C. Apté, J. Ghosh, and P. Smyth, editors, *SIGKDD*, pages 868–876. ACM, 2011.

[33] B. D. Minor and D. J. Cook. Forecasting occurrences of activities. *Pervasive Mob. Comput.*, 38:77–91, 2017.

[34] S. A. Mohamed and U. Martinez-Hernandez. A light-weight artificial neural network for recognition of activities of daily living. *Sensors*, 23(13):5854, 2023.

[35] Y. Nawal, M. Oussalah, B. Fergani, and A. Fleury. New incremental SVM algorithms for human activity recognition in smart homes. *J. Ambient Intell. Humaniz. Comput.*, 14(10):13433–13450, 2023.

[36] H. K. Ngankam, M. Lussier, A. Aboujaoudé, H. Pigot, S. Gaboury, K. Bouchard, M. Couture, N. Bier, and S. Giroux. NEARS-Hub, a lightweight edge computing for real-time monitoring in smart environments. In J. Bravo, S. F. Ochoa, and J. Favela, editors, *UCAmI*, volume 594 of *LNNS*, pages 125–138. Springer, 2022.

[37] M. Prossegger and A. Bouchachia. Multi-resident activity recognition using incremental decision trees. In A. Bouchachia, editor, *ICAIS 2014*, volume 8779 of *LNCS*, pages 182–191. Springer, 2014.

[38] H. Qian, Y. Mao, W. Xiang, and Z. Wang. Recognition of human activities using SVM multi-class classifier. *Pattern Recognit. Lett.*, 31(2):100–111, 2010.

[39] J. Read, A. Bifet, G. Holmes, and B. Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Mach. Learn.*, 88(1-2):243–272, 2012.

[40] V. G. Sánchez and N. Skeie. Decision trees for human activity recognition modelling in smart house environments. *Simul. Notes Eur.*, 28(4):177–184, 2018.

[41] V. Stankovski and J. Trnkoczy. Application of decision trees to smart homes. In J. C. Augusto and C. D. Nugent, editors, *Designing Smart Homes, The Role of Artificial Intelligence*, volume 4008 of *LNCS*, pages 132–145. Springer, 2006.

[42] T. Tan, M. Gochoo, F. Jean, S. Huang, and S. Kuo. Front-door event classification algorithm for elderly people living alone in smart house using wireless binary sensors. *IEEE Access*, 5:10734–10743, 2017.

[43] S. Xinjing, D. Wang, C. Quek, A.-H. Tan, and Y. Wang. Spatial-temporal episodic memory modeling for adls: encoding, retrieval, and prediction. *Complex & Intelligent Systems*, 10, 12 2023.

[44] N. Yala, B. Fergani, and A. Fleury. Towards improving feature extraction and classification for activity recognition on streaming data. *J. Ambient Intell. Humaniz. Comput.*, 8(2):177–189, 2017.