9-2020

DETECTIF: Unified detection and correction of IoT faults in smart homes

Madhumita MALIICK

Archan MISRA

Niloy GANGULY

Youngki LEE

2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)

DETECTIF : Unified Detection & Correction of IoT Faults in Smart Homes

Madhumita Mallick*, Archan Misra#, Niloy Ganguly*, Youngki Lee$,
*Dept. of CSE, Indian Institute of Technology Kharagpur, India, #School of Information Systems, SMU, Singapore,
$Dept. of CSE, SNU, Korea
Email: madhumitamallick@iitkgp.ac.in, archanm@smu.edu.sg, niloy@cse.iitkgp.ac.in, youngkilee@snu.ac.kr

Abstract*—This paper tackles the problem of detecting a comprehensive set of sensor faults that can occur in IoT-instrumented smart homes customized to infer Activities of Daily Living (ADL) from the activation of sensor sets. Specifically, sensors can suffer faults that (a) span durations that vary between several seconds to hours, (b) can result in both missing or false-alarm sensor-events. Previous fault detection approaches are geared primarily to identify missing faults (absence of sensor readings) of a permanent (very long-lived) nature, or sporadic false-alarm events. We propose *DetectIF*, a fault-detection framework that detects faults of varying time duration, and identifies both missing and false-alarm sensor events. *DetectIF's* key novelties include developing rules capturing spatiotemporal correlations among sensors and augmenting those rules with statistical properties of such sensor-specific behavior. To test *DetectIF* under a variety of fault behavior, we develop a unified fault framework where the tuning of a couple of parameters allows us to generate and inject faults of desired type and duration into an underlying sensor stream. Experiments with such comprehensive fault data shows that *DetectIF* achieves 82-95% fault-detection accuracy, improving precision by a huge amount (33-66%) over competitive, state-of-the-art baselines. Moreover, we demonstrate the benefits of applying *DetectIF* on unmodified, benchmark smart home datasets: it is able to detect additional likely faults that prior fault detection approaches miss, and thus consequently achieve an average of 30% higher ADL recognition accuracy compared to prior state-of-the-art fault detection techniques.

***Index Terms*—Smart home, IoT sensors, Transient Faults, Unified Fault Detection, Activities of Daily Living**

I. Introduction

Automatic monitoring of Activities of Daily Living (ADLs) is a cornerstone for many smart home applications related to wellness monitoring and elderly care [1], [2]. Such ADL monitoring typically leverages on a variety of deployed IoT devices (such as passive infrared (PIR) motion/light sensors, cabinet-mounted contact sensors & cameras). While machine learning-based techniques for ADL recognition *given sensor data* have been widely explored, relatively little attention (e.g., [3], [4]) has been given to the problem of understanding how incorrect or missing sensor data affects such ADL recognition. Such impairments in sensor data can occur due to a variety of factors, such as network failures, sensor hardware failures or environmental artifacts (e.g., a motion sensor occluded by a piece of furniture).

In this work, we develop, and demonstrate the benefits from, an improved and *unified* fault detection mechanism for such smart home settings, where sensors are often heterogeneous and perform *event-based* reporting (in contrast to conventional wireless sensor networks or WSNs, where sensors are often homogeneous and perform *value-based* reporting). Broadly speaking, sensor failures can manifest in distinct ways, categorized by either *duration* or *type*:

- *Duration:* Faults can have potentially three types of durability: (a) *permanent*, often due to hardware damage, where the sensor stops functioning irreversibly; (b) *sporadic* or random, which often manifests itself in singleton outlier values for value-based sensors (e.g., a room thermostat that suddenly reports a single data point as "100°C" while otherwise reporting values of 22-24°C); or (c) *transient*, where the fault usually persists for moderate durations (e.g., a few minutes to several hours) and is caused by some external or human artifacts.
- *Type:* Broadly, the fault type can be either (a) *missing*, where the sensor does not report an event when it should, or (b) *false-alarm*, where the sensor fires incorrectly, even in the absence of an actual underlying event.

Past work on fault detection in such environments has focused primarily on permanent, *missing* faults–i.e., scenarios typically characterized by a hardware failure and a consequent absence of any firings/reports from the faulty sensor. However, empirical evidence suggests that *false-alarm* events, especially of a *transient* nature, are not uncommon in smart homes [5], [6]. For example, [7] described how motion sensors have faulty activations due to the presence of bright sunlight reflecting off walls– this phenomenon is clearly transient (e.g., occurring only during the afternoons) and may have seasonal variations as well (e.g., occurring only during summers). Similarly, humans can inadvertently put an obstruction (such as glass) in front of the sensor, accidentally dislodge the sensor or even move a sensor-equipped furniture item [8], thereby causing a sensor to behave erroneously for a while.

Our proposed approach, called *DetectIF*[1], is designed to find and isolate both missing and false-alarm faults (of varying duration) in a unified manner, and thereby significantly improve the accuracy of sensor-based ADL recognition. *DetectIF* is based on two key insights: (a) Similar to Idea [3],

[1]**Detect**ion of **I**oT **F**aults

978-1-7281-7374-0/20/$31.00 ©2020 IEEE
DOI 10.1109/WoWMoM49955.2020.00028

we exploit the fact that each individual ADL is associated with a spatiotemporal activation pattern of *multiple* distinct sensors, implying that an individual sensor failure is likely to manifest as an *anomalous* pattern of collective sensor behavior. However, unlike Idea, failure in *DetectIF* does not mean just missing sensor readings, but also incorporates possible false-alarm faults. (b) We note that different ADLs are characterized not just by the activation of multiple event-based sensors, but also by distinctive *temporal properties* (e.g., number of ON/OFF transitions) of individual sensor events. Accordingly, *DetectIF* incorporates an additional set of such *statistical temporal features* to help identify a broader range of anomalies.

Our key contribution is the development of an *Activity-Weighted* approach to help *DetectIF* detect multiple classes of faults in heterogeneous IoT deployments. The approach works by mining a set of high-confidence, high-support association rules from historical patterns of ADL-induced sensor activation, and then determining when the test data, for a given ADL segment, deviates significantly from the mined patterns. To cater for the variety of faults, *DetectIF* association rules are augmented to include several key statistical temporal features about a sensor's reporting pattern. The Activity-Weighted approach computes the deviation scores for each ADL label separately, even though the inferred activity labels for each segment itself are estimated from the underlying noisy data and can thus be erroneous.

A key obstacle to our work is the lack of empirically-derived, generic fault models that span a wide variety of sensor failures. Accordingly, we first meticulously design a unified fault model, that can parametrically capture a wide variety of failure types, and evaluate *DetectIF* by suitably injecting synthetic faults into multiple existing smart home datasets. We demonstrate that the *Activity-Weighted DetectIF* can identify such faults with an accuracy ranging between 82-95% (across the datasets), as well as isolate the fault duration fairly precisely (Intersection-over-Union (IoU) values of 69%-91%). We also show that *DetectIF* outperforms previously-proposed competitive baselines [3], [6], achieving large improvements in precision (33 - 66%) and recall (20 - 40%) in fault detection across a wide variation (20 sec-60 min) in fault duration Secondly, we apply *DetectIF* on the *unmodified* smart home datasets to identify additional likely faults that past approaches fail to isolate (e.g., transient faults) and quantify the performance gain. We show that *DetectIF* helps identify significant additional unique sensor faults than previously diagnosed, and consequently helps improve state-of-the-art ADL recognition accuracy by as much as 42% over the unmodified data (and, on average, average by 30% over prior competitive baselines).

II. Background & Preliminaries

We first summarize the various datasets used to subsequently evaluate our techniques and outline the different types of faulty behavior that a sensor may manifest.

A. Dataset

Our work uses benchmark datasets (which include timestamped *ground truth* labels for the relevant ADLs) from 7 smart homes which have been widely utilized in prior smart home-related research, including 5 single-resident facilities and 2 multi-resident (2 occupants) homes. Table I summarizes the salient characteristics of these datasets. The first three datasets {kA, kB, kC} [9] are single resident homes with only binary-valued, event-driven sensors that generate output only when the sensor value changes *state*. The sensors include passive infrared (PIR) sensors attached to lights/wall/ceilings to detect motion, contact sensors (reed switches) on cabinets and doors to detect their usage, and float sensors to measure the flush of the toilet. The next four datasets collectively captured the interleaved ADLs from the Washington State University's CASAS smart home project [10], including (a) two single-resident houses (labeled as Ar and Ml) equipped with both event-driven (such as motion & door sensors) and non-binary, value-based sensors (temperature), and (b) two (T1 and T2) two-resident facilities. T1 & T2 had similar sensors as AR & Ml, as well as additional binary contact sensors attached to cupboards and various items, and value-based sensors that measure and report consumption of water & electricity.

Label	Name	# User	Dur. (days)	# ADL (# Activity)	# Sensor (# Event)
kA	KasterenA [9]	1	25	16 (283)	14 (2006)
kB	KasterenB [9]	1	15	25 (172)	27 (22595)
kC	KasterenC [9]	1	19	27 (254)	23 (39861)
Ar	Aruba [10]	1	219	11 (6477)	39 (805268)
Ml	Milan [10]	1	82	15 (2310)	33 (288498)
T1	Twor9-10 [10]	2	249	25 (3745)	100 (711421)
T2	TworSmr [10]	2	63	8 (1016)	100 (366075)

TABLE I: Summary of the 7 smart home Datasets (single and two-user) – their duration, number of unique ADLs and activities performed by the user (s), number of sensors deployed and sensor activations (Events) recorded.

Cook_Breakfast begin		Eat_Breakfast begin		Eat_Dinner begin	
2009-04-02 07:12:05.002864	**M005** ON	2009-04-02 07:19:15.061569	**M005** ON	2009-04-02 17:19:15.061569	**M001** ON
2009-04-02 07:12:05.999979	**M002** ON	2009-04-02 07:19:17.456099	**M002** ON	2009-04-02 17:19:17.456099	**M002** ON
2009-04-02 07:12:09.061644	**M006** ON	2009-04-02 07:19:20.002564	**M006** ON	2009-04-02 17:19:20.002564	**M006** ON
2009-04-02 07:12:09.655459	**M005** OFF	2009-04-02 07:19:21.077354	**M001** ON	2009-04-02 17:19:21.077354	**M001** OFF
2009-04-02 07:12:10.656179	**M002** OFF	2009-04-02 07:19:22.098438	**M002** OFF	2009-04-02 17:19:22.098438	**T003** 22.5
2009-04-02 07:12:12.003479	**M006** OFF	2009-04-02 07:19:23.292649	**M005** OFF	2009-04-02 17:19:23.292649	**M001** OFF
2009-04-02 07:12:13.191519	**M006** ON	2009-04-02 07:19:23.895149	**M006** OFF	2009-04-02 17:19:23.895149	**M006** OFF
2009-04-02 07:12:15.004275	**M004** ON	2009-04-02 07:19:24.024109	**M006** ON	2009-04-02 17:19:24.024109	**M006** ON
2009-04-02 07:12:15.081328	**T003** 22.5	2009-04-02 07:19:27.034922	**M006** OFF	2009-04-02 17:19:27.034922	**M006** OFF
2009-04-02 07:12:16.486749	**M006** OFF	2009-04-02 07:19:31.202889	**T003** 23.5	2009-04-02 17:19:31.202889	**T003** 23.5
.....		.....		.....	
.....		.....		.....	
2009-04-02 07:18:22.037296	**M004** OFF	2009-04-02 07:25:32.736459	**T002** 22.5	2009-04-02 17:25:32.736459	**T005** 22.5
Cook_Breakfast end		Eat_Breakfast end		Eat_Dinner end	

Fig. 1: Sample sensor readings in different activity segments

B. Observations

From the datasets, we see a typical pattern of *ADL-driven sensor activation*: when a resident performs an activity, her movement and object interactions cause *multiple sensors* to generate binary/numeric values. For example, motion and door sensors generate binary values when a door is opened, while value-based sensors (e.g., temperature and power sensors) generate numeric values continually. Figure 1 illustrates how various sensors are triggered during three distinct activity segments (*Cook Breakfast*, *Eat Breakfast*, and *Eat Dinner*),

along with the corresponding manually-annotated ground truth ADL history. We see that every sensor reading is associated with a timestamp, a sensor ID, and a corresponding value. Moreover, note that the same sensor can be triggered more than once in an activity segment (e.g., "M006" was activated twice during the single instance of the *Cook Breakfast* ADL). The duration of the ADLs performed by users also exhibits a wide variation. While most activities last between 6-10 minutes (median), the minimum ADL duration is often in seconds (for ADLs such as *Get Drink, Personal Hygiene*), while the maximum duration can extend to several hours, for ADLs such as *Leave House, Sleeping*.

C. Unified Fault Model

We aim to build a unified failure detection framework, that can detect both *missing* and *false-alarm* faults, of varying durations. Isolated fault models have been proposed before–e.g., transient fault models in sensors such as shaft encoders and GPS [11], and sporadic/permanent fault models for binary smart home sensors [3], [6], [12]. To develop the unified *DetectIF* framework, we extend these to create a unified fault model, which accommodates the different types of failures in event-based smart home sensors. Moreover, given the lack of explicitly annotated fault data in the benchmark datasets, such a model helps us to generate and inject *synthetically-generated* faults to quantitatively study *DetectIF's* performance.

We maintain a sensor dictionary (set of all sensors operating in a particular smart home) whereby each sensor s_i is represented by a tuple $< s_i, r_i, \mathcal{T}_i >$ where $r_i = \{0..1\}$ is the reliability score of the sensor (higher reliability implies a better operation) and $\mathcal{T}_i$ is the mean failure time. To inject faults, we first divide the sensor stream into one-minute time slots. We then choose a parameter n, which represents the maximum fraction of sensors that can be simultaneously faulty. At each time slot, we then choose a number k uniformly at random from $\{0, 1, \ldots, |S|\}$ ($|S|$ is the total number of sensors), and designate k of the total set of sensors as 'faulty'. The likelihood of an individual sensor being selected is proportional to its reliability score, with the fault likelihood being higher for sensors present (in the original trace) in that slot compared to sensors absent in the trace–this ensures that the failure likelihood is more equitably distributed.

A chosen "faulty" sensor is then assumed to remain in the faulty state for the next t_i time slots. The choice of t_i gives rise to faults with 3 distinct temporal properties: (a) *Sporadic/Random:* here $t_i = 1$, i.e., a sensor fault lasts only for one single time unit, with faults having no correlation across different time slots; (b) *Transient:* in this case, t_i is exponentially distributed with an average value $\mathcal{T}_i$, where $1 \leq \mathcal{T}_i \leq \infty$; and (c) *Permanent:* In this case, time $t_i = \infty$ i.e., once the sensor enters a faulty state, the state will persist' without explicit manual intervention.

D. Three Types of Faults:

A sensor chosen to be faulty for t_i time slots can exhibit two broad fault types: *missing* or *false-alarm*. For both binary and numeric/value-based sensors, missing faults are simulated by causing the sensor not to fire (or generate a report) when it should. False-alarm faults are simulated by generating *phantom* readings (extremal value in case of numeric sensors) even though there is no actual underlying event. The faults can be further classified into two distinct *ADL-dependent* types {atypical, spurious}, based on whether the sensor s is a member (or not) of $\hat{S}_i$, the set of sensors that are associated with the corresponding ADL A_i. Thus this results in 3 distinct fault *types* (illustrated in Figure 2) elaborated next.

(a) A sensor $s \in \hat{S}_i$ is said to exhibit a *missing* fault if it fails to report values that should legitimately be present. For example, a motion sensor may be blocked temporarily because of an obstruction and may not report any motion activity within its sensing field.

(b) A sensor $s \in \hat{S}_i$ is said to exhibit an *atypical* (false-alarm) fault if: (i) s is a sensor that, under normal behavior, fires at least once during the occurrence of a specific ADL A_i; however, (ii) the behavior of s is currently inconsistent with historical data observed during past instances of activity A_i. For example, consider an ADL "Watch TV", where a motion sensor typically activates only during the start and at the end of the activity. However, because of bright sunlight reflecting off the opposite wall, the sensor can report continued motion *throughout* the activity.

(c) A sensor $s \notin \hat{S}_i$ is said to exhibit a *spurious* (false-alarm) fault if it is activated and reports values, even though it should not have been present *at all* during ADL A_i. For example, a pet animal may trigger a kitchen door/cupboard sensor, even though the human resident is performing a "Showering" ADL and is nowhere in the vicinity of that cupboard.

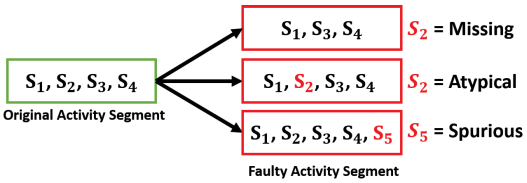

Fig. 2: Different types of Faults. Assuming S_2 is present in the original activity segment A_i, (a) S_2 may be *missing* (the top) due to some anomaly, (b) S_2 may be present but exhibiting *atypical*, and (c) Another sensor S_5, which reports but is not associated with A_i is termed as *spurious*.

III. DetectIF: Profiling Normal Sensor Behavior

Our fault/anomaly detection approach is based on the assumption that faults manifest abnormal patterns (both missing and false-alarms), which are inconsistent with prior historical data. The overall *DetectIF* approach thus has two distinct phases: (a) an initial *Training* phase (which we detail in this section), where past historical data is mined to build up *profiles/rules* of normal sensor behavior, and (b) a subsequent *Online Detection* phase (which is explained in Section IV), where future streams of observed sensor data are analyzed

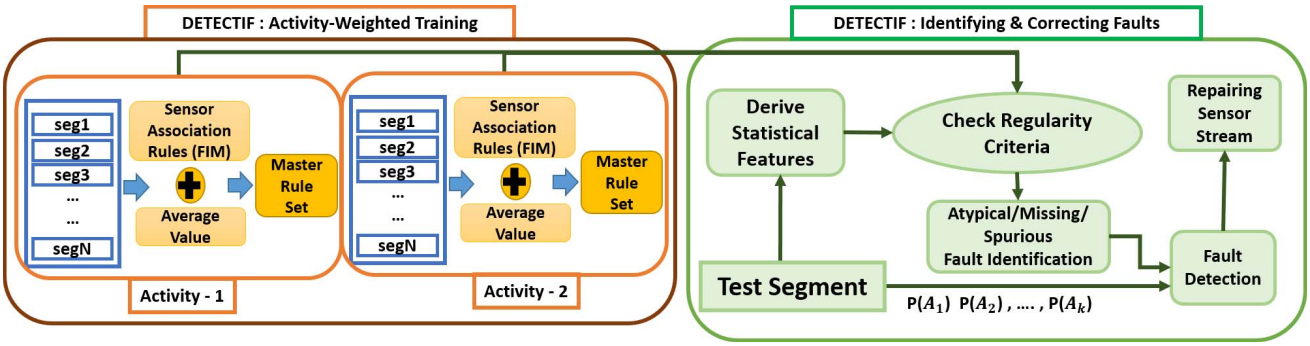

Fig. 3: Overall Architecture of *Activity-Weighted DetectIF* [illustrated with 2 ADLs].

to identify *anomalous deviations from such profiles*. Figure 3 illustrates the various steps in *DetectIF*.

More specifically, inspired by Idea [3], we utilize two insights: (a) An activity (ADL) usually involves the activation of multiple sensors. Accordingly, each ADL typically has one or more patterns of spatiotemporal correlation among multiple sensors (refer Figure 1); and (b) Given that the underlying ADL patterns are reasonably consistent (e.g., the number of 'fridge open' events does not vary dramatically across different 'Cooking' ADL instances) the overall spatiotemporal activation (or reporting) pattern of the associated sensors is also fairly *regular* in the absence of faults. Accordingly, the training phase involves the creation of a set of *extended* association rules, using the *Frequent Itemset Mining* [13] (FIM) algorithm. We assume that the sensor stream (used for training) has already been partitioned into distinct ADL segments using a state-of-the-art event segmentation technique (such as [14]–[16]).

A. Basic Association Rules

The input to the FIM algorithm consists of a set of *segments*, the values the constituent sensors exhibit within that segment, along with the annotated ground truth labels of the segments (e.g., 'Cooking', 'Eating'). We partition the segments into γ ADL-specific *master segment sets* (one for each of the γ distinct ADL labels), and FIM is *independently* applied to each of these γ segment sets.

Association Rules: Given a specific master segment set, if the sensors $\{s_1, s_2, s_3, \ldots, s_i, s_j\}$ co-occur in at least one of the segments, the FIM rules (Rl's) are written in the following format:

$$Rl :: \{s_1, s_2, s_3, \ldots, s_i\} \implies s_j :: \text{support} : \text{confidence} \quad (1)$$

where $s_1, s_2, \ldots, s_i, s_j$ are the sensors, $(s_1, s_2, \ldots, s_i)$ and s_j are the *antecedents* and the *consequent* respectively. *Support* defines the fraction in the training dataset (activities) containing $\{s_1, s_2, \ldots, s_i\}$ and s_j, while *confidence* denotes the conditional probability of s_j occurring when $s_1, s_2, \ldots, s_i$ occurs, *confidence*= $\text{support}(X \bigcup Y)/\text{support}(X)$, where $X = \{s_1, s_2, \ldots, s_i\}$, and $Y = s_j$.

B. Augmentation with Statistical Features

Unlike prior anomaly detection approaches, the *DetectIF* process computes and creates *augmented association rules* that include a few additional statistical features related to the *activation pattern* of each sensor.

Statistical Features for Binary (Event) Sensors: If a sensor (s) is a binary sensor (and $s \in Rl$), then the following values are computed separately for each ADL segment: (a). $s^{Rl}_{n_{on}}$ & $s^{Rl}_{n_{off}}$, the number of times the sensor s switched to the *ON* and *OFF* states, respectively. (b). $s^{Rl}_{d_{on}}$ & $s^{Rl}_{d_{off}}$, the total time duration (within the ADL segment) for which the sensor was s in the *ON* and *OFF* states, respectively. Subsequently, we compute the following 6 statistical features (for each sensor $s \in |S|$), across all the segments in the master segment set:

(i) mean μ and std. dev. σ for the $s^{Rl}_{n_{on}}$ and $s^{Rl}_{n_{off}}$ features;
(ii) the mode for the duration features ($s^{Rl}_{d_{on}}$ and $s^{Rl}_{d_{off}}$). Note that the mode can be a *set* of distinct values, in case the distribution is multi-modal.

Finally, we aggregate these features into a 6-element feature vector $s^{SF(Rl)}$ (*statistical features* of s in a Rule Rl):

$$s^{SF(Rl)} = \langle \mu^{s^{Rl}_{n_{on}}}, \sigma^{s^{Rl}_{n_{on}}}, \mu^{s^{Rl}_{n_{off}}}, \sigma^{s^{Rl}_{n_{off}}}, mode^{s^{Rl}_{d_{on}}}, mode^{s^{Rl}_{d_{off}}} \rangle \quad (2)$$

Statistical Feature for Numeric (Value) Sensors: For value-based sensors, we determine an appropriate 'typical' value through quantization unlike binary sensors. First, the raw readings are used to compute the standard deviation, which yields the quantization unit (a bin size B). Subsequently, each sensor reading (across all segments in the master segment set) is normalized into the range $(0, 100)$ by multiplying the raw value v by the normalizing factor $N(v) = 100 * \frac{(v - min(S))}{max(S) - min(S)}$ (where $max(S), min(S)$ represent the largest and smallest observed value of S), and then quantizing this normalized value into one of the B bins through the transformation $\lceil \frac{N(v) * B}{100} \rceil$. The 'typical value' $s^{SF(Rl)}$ is then obtained by finding the mode of this quantized distribution, i.e., $s^{SF(Rl)} = \langle mode^{s^{Rl}_{bin}} \rangle$

C. Extended Association Rules

Finally, we augment each of the FIM-derived association rules to incorporate these additional *statistical* features ($s^{SF(Rl)}$). The final extended rule is in the form of –

$$Rl :: \{s_1, s_2, s_3, \ldots, s_i\} \implies s_j : \text{support} : \text{confidence} : s_1^{SF(Rl)} : s_2^{SF(Rl)} : \ldots : s_i^{SF(Rl)} : s_j^{SF(Rl)} \quad (3)$$

At the end of this training phase, we thus have a set of extended association rules, one for each of the γ ADL labels. We designate this set of rules as the **master rule set**.

IV. DETECTIF : IDENTIFYING & CORRECTING FAULTS

We now detail *DetectIF's* mechanism for detecting and classifying sensor faults into three types {missing, atypical, spurious}, given a *test* ADL segment. We use three distinct types as the atypical and spurious cases result in distinct *sensor cleaning* techniques (Section IV-D). The *Activity-Weighted* approach of *DetectIF* assumes that an existing state-of-the-art activity recognition algorithm (e.g., [17]–[19]) is first used to provide the probability p_i that the segment is associated with ADL A_i ($i = \{1, \ldots, \gamma\}$). Figure 3 illustrates the steps in our approach, while Algorithm 1 outlines the formal steps.

ALGORITHM 1: *DetectIF*: Fault Identification and Detection

Data: Sg: A list of test segments of sensor events
Rl: Extended Rulebase in the form of
$\langle Rl :: \{s_a, s_{a+1}, \ldots, s_{a+p}, s_{a+p+1}, \ldots, s_q\} \implies s_j : support : confidence : s_a^{SF(Rl)} : s_{a+1}^{SF(Rl)} : \ldots : s_{a+p}^{SF(Rl)} : s_{a+p+1}^{SF(Rl)} : \ldots : s_q^{SF(Rl)} : s_j^{SF(Rl)} \rangle$ *//each Ac has its own set of Rl's* $\in [RL_{Ac}]$.
for $i \leftarrow 0$ **to** $|Sg|$ **do**
  $S \leftarrow \{s_1, s_2, s_3, \ldots, s_n\}$ //sensors in Sg
  AC: List of predicted activities: $\{Ac_1, Ac_2, \ldots, Ac_n\}$
  for $Ac \leftarrow 0$ **to** *AC* **do**
    for $Rl \leftarrow 0$ **to** $|RL|$ **do**
      $Rule_{score} = support_{Rl} * 1/(1 - confidence_{Rl})$
      If $s = consequent(Rl)$ is *Atypical*
      $s^{Atyp_{score}} \leftarrow s^{Atyp_{score}} + Rule_{score}$
      ElseIf $s = consequent(Rl)$ is *Missing*
      $s^{Miss_{score}} \leftarrow s^{Miss_{score}} + Rule_{score}$
      Else **for** $s \leftarrow 0$ **to** $|S|$ **do**
        Determine if $s \in S$ is *Spurious*
        $s^{Spu_{score}} \leftarrow s^{Spu_{score}} + Rule_{score}$
      end
      $s^{Atyp_{score}} = s^{Atyp_{score}} + s^{Atyp_{score}} * P(Ac)$,
      $s^{Miss_{score}} = s^{Miss_{score}} + s^{Miss_{score}} * P(Ac)$,
      $s^{Spu_{score}} = s^{Spu_{score}} + s^{Spu_{score}} * P(Ac)$
    end
  end
end

A. Shortlisting [Rl]

Given a test activity segment Sg, we first shortlist a set of relevant rules [RL] from the given *master rule set* associated to the corresponding Activity (Ac), such that for each rule $Rl \in [RL_{Ac}]$, antecedents(Rl) $\subseteq$ sensors(Sg) (the set of sensors in Sg), and *Regularity*(s, Sg, Rl)= 1; $\forall s \in antecedents(Rl)$.

Regularity(s, Sg, Rl)**:** Given an activity segment Sg that is associated with a shortlisted rule Rl ($s \in Sg$ & $s \in Rl$), the binary-valued regularity score checks whether the behavior of sensor s during the segment Sg conforms to its *statistical values* expressed in a rule Rl. Assume that the activity segment Sg is given by $\{s_a, s_{a+1}, \cdots s_{a+i}, s_{a+i+1}, \cdots, s_{a+k}\}$, where s_a to s_{a+i} are binary sensors and s_{a+i+1} to s_{a+k} are numeric sensors. The *statistical features* for this test segment can be represented by $\{s_a^{SF(Sg)}, s_{a+1}^{SF(Sg)}, \cdots s_{a+i}^{SF(Sg)}, s_{a+i+1}^{SF(Sg)}, \cdots, s_{a+k}^{Sg}\}$.

For a binary sensor $s \in Sg$ & $s \in Rl$, we define *Regularity*$(s, Sg, Rl) = 1$ (*Regularity=0* otherwise) if,(a) its total number of ON/OFF switches is within the std. dev. of the corresponding mean values, AND (b) its total ON/OFF durations equal to one of the corresponding modes–i.e., if:

$$(\mu^{s_{n_{on}}^{Rl}} - \sigma^{s_{n_{on}}^{Rl}}) \leq s_{n_{on}}^{Sg} \leq (\mu^{s_{n_{on}}^{Rl}} + \sigma^{s_{n_{on}}^{Rl}}), \text{ and}$$
$$(\mu^{s_{n_{off}}^{Rl}} - \sigma^{s_{n_{off}}^{Rl}}) \leq s_{n_{off}}^{Sg} \leq (\mu^{s_{n_{off}}^{Rl}} + \sigma^{s_{n_{off}}^{Rl}}), \text{ and} \quad (4)$$
$$s_{d_{on}}^{Sg} \in mode^{s_{d_{on}}^{Rl}}, \text{ and } s_{d_{off}}^{Sg} \in mode^{s_{d_{off}}^{Rl}}$$

Similarly, for a value sensor $s \in Sg$ and $s \in Rl$ *Regularity*(s, Sg, Rl) is 1 if its quantized value corresponds to one of the modes defined in Equation 2–i.e., $mode^{s_{bin}^{Sg}} \in mode^{s_{bin}^{Rl}}$; else $Regularity = 0$.

B. Per-Rule Sensor Fault Classification

If $Regularity = 0$, we infer that the sensor might potentially be faulty and classify it into one of 3 types as follows.
Atypical: For each shortlisted rule Rl, if consequent(Rl)$\in$ sensors(Sg), and *Regularity*(consequent(Rl), Sg, Rl)=0, then consequent(Rl) is classified as exhibiting an *atypical* fault. In other words, a sensor is classified as *atypical* if its usage-behavior deviates from its regular behavior (as defined in the rule Rl) while the other 'correlated' sensors behave normally, Its *anomaly score (AS)* is given by

$$AS(Rl) = [1/(1 - \text{confidence}_{Rl}) * \text{support}_{Rl}] \quad (5)$$

–i.e., the deviation is considered more noteworthy if, in the training master set, (a) the antecedent(Rl) and consequent(Rl) have occurred more frequently (higher support) or (b) the likelihood of observing consequent(Rl), given antecedent(Rl) is higher (higher confidence). If the sensor s exhibits *atypical* fault over multiple rules, the anomaly scores are summed up.
Missing: If consequent(Rl)$\notin$ sensors(Sg) and consequent(Rl) is not already shortlisted as an *atypical* fault, then consequent(Rl) is said to be *missing*, i.e., if a sensor expected to be present in the segment Sg is entirely absent, it is classified as *missing*. The anomaly score is calculated as before (using Eq. 5) and summed up over all such matching rules.
Spurious: Finally, a sensor s is classified to be *spurious* if $s \in sensors(Sg)$, but doesn't occur in the antecedent set of any of the shortlisted rules [RL] i.e. $s \notin antecedent(Rl), (\forall Rl \in [RL])$ nor in the consequents(RL). In other words, if a sensor is observed to be present in the test segment but was not observed concurrently with the other reporting sensors (in the Master segment set during training), it is likely to be a false-alarm fault. The *anomaly score* for all those shortlisted rules is then added to compute the anomaly score.

Via this process, given a segment Sg and an associated *master segment set* (i.e., a set of FIM-based rules for a specific ADL, obtained by mining the 'training' data), *DetectIF* classifies a faulty sensor into precisely one of {spurious, atypical, missing} categories and provides a corresponding *anomaly score*.

C. Fault Detection via Aggregation Across All Activities

The final declaration of whether a sensor is faulty (and its fault type) is then based on this overall *anomaly score*. Each activity-specific master set (i.e., $\forall Ac \in [AC]$) has an associated set of anomalous sensors along with their anomalous score and category (*Atypical/Missing/Spurious*). Subsequently, for a sensor $s \in Ac$, its activity-specific *anomaly score* is multiplied (weighted) by the probability of occurrence of Ac, and summed up across all activity labels to obtain a total *weighted anomaly score*. (In case a sensor s observed in the segment ($s \in Sg$) is classified as potentially atypical for some ADLs and spurious for others, the final choice between spurious and atypical is based on the dominant label among all the ADLs.) Finally, sensors whose *weighted score* exceed a pre-defined threshold (the threshold itself determined from prior data by the shoulder-locating method) are declared to be faulty.

D. Repairing the Sensor Stream

After identifying the faulty sensor data using *DetectIF's* detection techniques, we additionally *repair* the underlying sensor stream as follows:

1) First, we clean *spurious* faults in a test segment Sg. A sensor s classified as *spurious* fault has its events (or values) removed from the underlying sensor stream.
2) Next, if a sensor s is identified as exhibiting a *atypical* or *missing* fault, we select the extended association rule Rl with *maximum support* that has the maximum number of elements in $(sensors(Rl) \bigcap sensors(Sg))$, with the condition $s \in sensors(Rl)$. We then replace sensor events s^{Sg} with the likely behavior denoted by s^{Rl} from the extended association rule. In other words, we restore the behavior of the sensors that undergo *atypical* or *missing* fault in the test segment with their 'statistically representative' parameter values specified in Rl.

In Section VI-C, we shall demonstrate that such *DetectIF*-based repairing helps to improve the accuracy of ADL classification significantly.

V. Metrics & Baselines

Before proceeding to the experimental results, we first summarize the key evaluation metrics and algorithms used to demonstrate the superior performance of *DetectIF*.

Fault Evaluation Metrics: We use the two standard metrics **precision** and **recall** to evaluate the ability of *DetectIF* (and its competing alternatives) to detect the various types (*atypical*, *spurious*, or *missing*) of sensor faults.

A. Fault Detection Alternatives (Baseline Algorithms)

We shall compare *DetectIF* against two widely-used, state-of-the-art fault detection algorithms.

CLEAN [6]: This algorithm is used primarily to detect sporadic faults ('outliers') in binary sensors. The approach uses a hierarchical feature space of [timestamp, location, object, user], over sensor activations, to quantify the similarity between any two sensors and cluster sensor events, subsequently declaring smaller-sized clusters as 'anomalies'. CLEAN also assigns a greater weight (likelihood of being noisy) to an event if the sensor (i) has been diagnosed to be faulty recently or (ii) has historically exhibited higher failure frequency.

Idea [3]: This approach learns the functional redundancies (correlation) among multiple sensors, and aims to improve the robustness of ADL detection in the presence of permanent sensor failures ('missing' faults). To identify sensor failures, Idea assigns a sensor a continually updated *rarity* score which reflects deviations from its expected behavior.

B. Activity Detection Algorithms

To demonstrate *DetectIF's* ability to improve ADL detection accuracy, we shall utilize the following 3 most popular, state-of-the-art activity detection algorithms: (1) *Naive Bayes (NBC) Detector* [17], (2) *Hidden Markov Model (HMM) Detector* [18], and (3) *Frequent Item set Mining (FIM) based Detector* [3], [19], each of which operates on individual ADL *segments*–i.e., over a collection of sensor events delineated as belonging to a single ADL. Note that *DetectIF* is a sensor error-correction mechanism, and is independent of the specific ADL detection method chosen.

VI. Experimental Results

In this section, we use the dataset described in Section II (Table I) to evaluate the proposed *DetectIF* approach for fault detection. We first inject a unified set of synthetic faults, using the approach described in Section II-C–i.e., selecting a maximum of 10% of sensors to be faulty in a single time slot, equi-probably generating, exponentially-distributed (mean = $\mathcal{T}_i$, $\mathcal{T}_i$ ={1s, 10s, 20s, 40s, 1m, 2m, 5m, 10m, 15m, 20m, 30m, 60m}) *spurious* or *missing faults*. Using such synthetically faulty data, we evaluate *DetectIF's* ability to (a) detect the different types of faults (missing, atypical and spurious) and (b) delineate the corresponding fault duration, compared to the respective baseline algorithms. We then show how *DetectIF*-based cleaning of faults helps to improve the accuracy of subsequent ADL classification significantly. We subsequently apply *DetectIF* to an existing, *unmodified* dataset (where there are no explicit fault annotations) to understand its impact–i.e., see whether it can identify apparent faults that prior techniques cannot, and quantify the resulting improvement in ADL recognition accuracy.

A. Fault Detection Accuracy

While the sensor trace contains a combination of all 3 types of faults (spurious, missing, and atypical), we analyze the fault detection performance of each type separately.

1) Detecting spurious faults: Figure 4 reports the precision of detecting *spurious* faults averaged across all smart homes, as a function of the mean fault duration $\mathcal{T}_i$. When $\mathcal{T}_i$ is 1, i.e., when the fault is sporadic, CLEAN (which explicitly targets sporadic faults) performs slightly better than our approach (see inset). However, as the $\mathcal{T}_i$ increases, *DetectIF* begins to

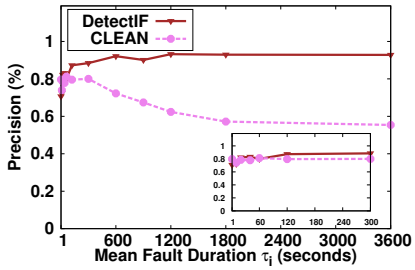

Fig. 4: Precision of detecting **Spurious** fault in all smart home datasets for various $\mathcal{T}_i$

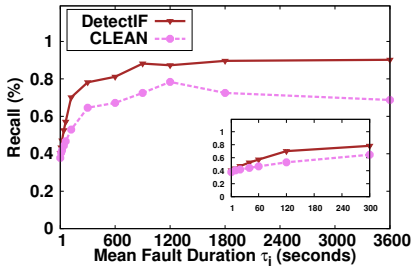

Fig. 5: Recall of detecting **Spurious** fault in all smart home datasets for various $\mathcal{T}_i$

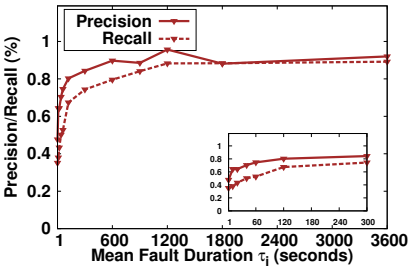

Fig. 6: Precision and Recall of detecting **Atypical** fault in all smart home datasets for various $\mathcal{T}_i$

TABLE II: Precision(Recall): **Spurious** fault detection across different smart homes (single & double resident) and sensors (binary & numeric)

	$\mathcal{T}_i$ =1 sec		$\mathcal{T}_i$ =60 sec		$\mathcal{T}_i$ =20 min		$\mathcal{T}_i$ =60 min	
	CLEAN	***DetectIF***	**CLEAN**	***DetectIF***	**CLEAN**	***DetectIF***	**CLEAN**	***DetectIF***
kA-kB-kC	**0.82**(0.4)	0.75(0.4)	0.81(0.45)	**0.82(0.6)**	0.61(0.85)	**0.94(0.9)**	0.56(0.68)	**0.93(0.92)**
Ar-Ml	**0.75**(0.33)	0.67(**0.49**)	0.8(0.5)	**0.875(0.59)**	0.65(0.75)	**0.9(0.89)**	0.55(0.74)	**0.92(0.91)**
T1-T2	**0.75(0.4)**	0.7(0.37)	0.8(0.45)	**0.81(0.5)**	0.6(0.75)	**0.95(0.8)**	0.56(0.65)	**0.93(0.87)**

TABLE III: Precision(Recall): **Missing** fault detection across different smart homes (single & double resident) and sensors (binary & numeric)

	$\mathcal{T}_i$ =1 sec		$\mathcal{T}_i$ =60 sec		$\mathcal{T}_i$ =20 min		$\mathcal{T}_i$ =60 min	
	Idea	***DetectIF***	**Idea**	***DetectIF***	**Idea**	***DetectIF***	**Idea**	***DetectIF***
kA-kB-kC	0(0)	0.5(0)	0.2(0.25)	**0.67(0.5)**	0.67(0.75)	**0.88(0.78)**	0.82(0.81)	**0.90(0.82)**
Ar-Ml	0(0)	**0.55**(0)	0.25(0.25)	**0.71(0.51)**	0.65(0.78)	**0.88(0.79)**	0.88(**0.83**)	**0.89(0.83)**
T1-T2	0(0)	**0.42(0.1)**	0.2(0.25)	**0.69(0.45)**	0.67(0.68)	**0.83(0.75)**	0.82(**0.82**)	**0.85**(0.79)

outperform CLEAN. More specifically, our precision stabilizes for $\mathcal{T}_i > 10$ minutes, while CLEAN's performance continues to deteriorate. This is because with a longer fault duration, the fault-likelihood (weightage) assigned by CLEAN increases, implying that the sensor continues to be flagged as faulty (increasing the false-positive rate) even when the transient failure has ended. Table II plots the precision values separately for different types of smart homes (single/double resident) and sensors (binary/binary+numeric). We see that *DetectIF* performs significantly well detecting *spurious* faults in terms of precision (up to 95%) in all types of smart homes.

Figure 5 shows corresponding averaged recall values, while Table II details the changes in recall for individual types of homes (in brackets). We see that *DetectIF's* recall grows as $\mathcal{T}_i$ increases, stabilizing above 80% once $\mathcal{T}_i$ exceeds 10 minutes. Overall, *DetectIF* outperforms CLEAN across a wider range of failure behavior.

2) Detecting missing faults: Figures 7 and 8 show the precision and recall results for *missing* fault detection averaged over all smart homes. Similarly, the precision (recall) values for individual smart homes types are reported in Table III. In this case, we compare our approaches against *Idea* (which explicitly targets such faults). *DetectIF* has both high precision (∼88%) and high recall (∼80%) as compared to *Idea* when the sensor readings are missing for even modestly long durations ($\geq$ 20 min). *DetectIF* outperforms Idea, especially when the faults are sporadic or highly transient (lower values of $\mathcal{T}_i$)—in an extreme situation, Idea's precision/recall = 0 when $\mathcal{T}_i = 1$ sec. Indeed, Idea is built to detect permanent missing faults– accordingly, its performance improves only when the fault durations are longer (e.g., $\mathcal{T}_i = 60$ min).

In addition to sporadic/transient missing faults, we also emulate *permanent* missing faults in our sensor stream by emulating $t_i = \infty$ (i.e., once failed, the sensor stays faulty for the rest of the smart home trace) for 5% of the total sensors. We evaluated *DetectIF* using data from 3 separate representative smart homes, comparing it with Idea in terms of the *time to detection*. As shown in Figure 9, *DetectIF* detects the missing sensor within an average of 15-20 minutes for all 3 environments, whereas Idea takes >1.5 hours on average to detect such missing sensors. It appears that *DetectIF's* use of additional statistical features allows it to identify such faults much faster than Idea, which merely utilizes the binary absence/presence of a sensor during a specific ADL.

3) Detecting atypical faults: As explained in Section II, we also introduced a third type of fault – *atypical*, which has not been considered in prior work. Atypical faults are essentially those where a sensor that is expected to fire during an ADL (i.e., is part of that ADL's rule antecedents) does fire, but its *statistical behavior is abnormal*. Such anomalies are detected due to the additional statistical features employed by *DetectIF*, which prior approaches (such as CLEAN or Idea) fail to detect. Accordingly, we show the precision and recall results of only *DetectIF* in Figures 6. In addition, Tables IV shows the precision (recall) for individual types of homes and sensors.

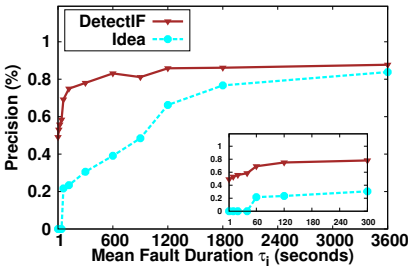

Fig. 7: Precision of detecting Missing fault in all smart home datasets

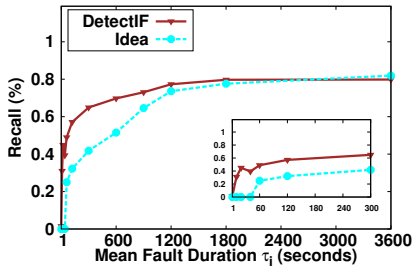

Fig. 8: Recall of detecting Missing fault in all smart home datasets

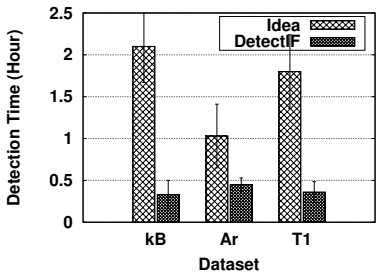

Fig. 9: Time taken to detect failed sensors across three smart home datasets

TABLE IV: Precision(Recall): **Atypical** fault detection across different smart homes (single & double resident) and sensors (binary & numeric)

	$\mathcal{T}_i$ =1 sec	$\mathcal{T}_i$ =60 sec	$\mathcal{T}_i$ =20 min	$\mathcal{T}_i$ =60 min
kA-kB-kC	**0.5(0.4)**	**0.77(0.53)**	**0.98(0.96)**	**0.95(0.95)**
Ar-Ml	**0.48(0.4)**	**0.75(0.6)**	**0.96(0.93)**	**0.93(0.92)**
T1-T2	**0.45(0.35)**	**0.72(0.45)**	**0.93(0.76)**	**0.88(0.8)**

As can be seen, both precision and recall improve gradually as $\mathcal{T}_i$ is larger, i.e., our approach improves as the failure duration increases (fault duration becoming transient rather than random), and stabilizes in case of permanent faults, obtaining up to ∼95% precision and ∼93% recall in single user smart homes (both binary and binary+numeric sensor-equipped). We note that, the *regularity* feature plays a vital role in identifying this type of faults. The reason being it is able to flag a sensor as faulty (based on the abnormality of its ON/OFF transitions and duration) even when it is normally expected to be present in an ADL.

TABLE V: Fault duration IoU score for various fault types

	Missing			Spurious			Atypical		
$\mathcal{T}_i$ =	**40s.**	**5m.**	**15m.**	**40s.**	**5m.**	**15m.**	**40s.**	**5m.**	**15m.**
Aob	0.52	0.57	**0.69**	0.74	0.76	**0.88**	0.63	0.71	**0.8**
Awt	0.65	0.69	**0.77**	0.8	0.89	**0.91**	0.78	0.87	**0.89**

B. Inferring Fault Duration

Besides evaluating the detection accuracy, we also check how accurate *DetectIF* is in delineating the total duration (i.e., (start, end) times) of such faults. To investigate this, we conducted experiments where we injected faults with 3 different mean durations ($\mathcal{T}_i$= {40 secs, 5 mins, 15 mins}). To evaluate the accuracy of duration estimation, we utilize the IoU (Intersection over Union) score–i.e., the ratio of the intersection of the predicted and actual fault durations to the the union of these time intervals.

From Table V, we see that *DetectIF* perform poorly when the fault duration is small, and the fault type is *missing*. A *missing* sensor is one that is effectively absent during an entire ADL segment; accordingly, its fault duration (estimated as the time from its previous activation till the next activation) implicitly spans across multiple ADL segments. The estimated fault duration for *missing* faults is thus larger than 6-10 mins

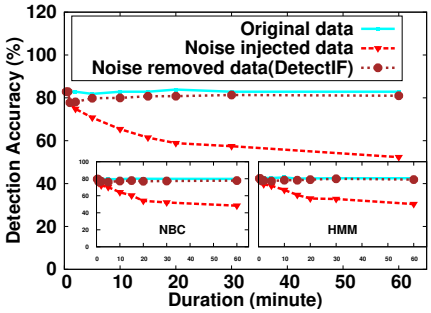

Fig. 10: ADL accuracy (noisy vs. cleaned data): FIM, NBC and HMM Activity Detection (the latter two in the inset)

(the typical duration of most ADL segments), resulting in low IoU for short-lived failures. However, this IoU score increases as $\mathcal{T}_i$ increases. On the other hand, we are able to isolate *atypical* and *spurious* faults much accurately, achieving IoU values of ∼90% across all fault durations.

C. Impact on ADL Detection

To measure the impact of sensor faults (and the subsequent repair of such faults) on the ultimate goal of activity detection, we compared ADL detection accuracy before and after the application of *DetectIF* to detect and repair sensor faults (see Section IV-D). Figure 10 plots the results for the 3 different ADL detection algorithms described in Section V-B, averaged over all datasets. The top line in three plots signifies the activity detection performance in the original data, without any additional synthetically-injected sensor faults, thus representing the best possible ADL detection performance.

As we see, initially when $\mathcal{T}_i < 60$ sec., ADL detection is relatively unaffected, even if the underlying faults (noise) are not cleaned. In other words, *sporadic* faults do not cause any significant impairment to ADL detection accuracy. However, as the mean fault duration $\mathcal{T}_i$ increases, the ADL detection accuracy degrades significantly (dropping to below 50%). However, when we apply *DetectIF* to clean the faults, we see that all 3 algorithms exhibit significant performance improvement, achieving accuracy that is virtually indistinguishable from the original sensor stream (the one without any added synthetic faults).

D. Practical Impact on Existing Datasets

We finally extend our experiments to study the effectiveness of *DetectIF* on the original datasets–i.e., without injecting any synthetic failure data. As the original datasets do not have any explicit fault annotation, we can study this effectiveness only indirectly, by quantifying the number of (apparent) faults detected and the resulting impact on ADL recognition accuracy. For conciseness of exposition, we consider the kB (KasterenB) dataset (which is known to be inherently noisy) and apply *DetectIF* on the unmodified sensor stream to identify additional likely faults. For comparison, we use prior fault detection approaches—{*CLEAN, Idea*}-as competitive baselines.

On manually examining the sensor patterns flagged out as anomalous by *DetectIF*, we observe the likely occurrence of both permanent *and* transient faults. For example, sensor 28, which is installed in the kitchen, is constantly 'on' during the last day of data collection. As an another example, on Day 5, *DetectIF* detects a missing "Toilet Door" sensor for the "Use Toilet" activity. On further inspection we find that, the door was previously closed in the last "Go to Bed" activity, and reported to be open in the subsequent "Take Shower" activity 20 minutes later, but was apparently undisturbed during the intervening "Use Toilet" activity!

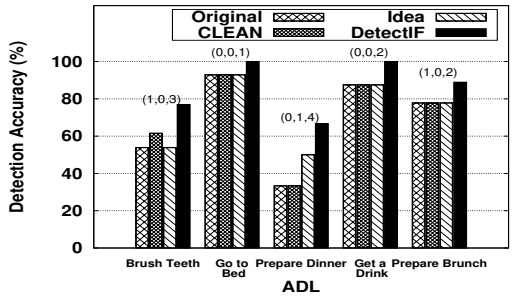

Fig. 11: ADL Detection Accuracy with/without fault detection & repair on kB dataset. The triplet on each bar represents the number of unique sensor faults identified by (*CLEAN*, *Idea*, *DetectIF*) respectively.

We measure how many such unique sensor faults (spurious+missing) *DetectIF* is able to identify, comparing this with *CLEAN* (which detects only spurious presence) and *Idea* (which detects only missing presence). As in Section VI-C, we also quantify the resulting improvement in ADL detection after *repairing* such identified faults. Figure 11 presents the results for the kB dataset for some hand-picked activities: the triplet on each bar represents how many *unique sensor faults* are identified by (*CLEAN*, *Idea*, *DetectIF*) respectively. We see that *DetectIF* is always able to detect a larger number of faults compared to *CLEAN* and *IDEA*, and is able to substantially improve ADL detection accuracy. More specifically, *DetectIF* can improve ADL detection accuracy by up to 42% compared to that achieved in the absence of any fault detection. Also, ADL recognition accuracy with *DetectIF* is, on average, 30% higher than that achieved with either CLEAN or Idea-based fault detectors.

VII. Related Work

Fault Detection in Wireless Sensor Networks: The problem of fault detection has received considerable attention in wireless sensor networks. Much of this work is restricted either to purely homogeneous sensor networks or to handling only numeric sensors [20]–[23]. [24] proposes an ARIMA based framework to detect transient faults, where significant deviations from time-series predictions of sensor values are flagged as potential faults. Alternately, [25] exploit Bayesian methods, built upon spatiotemporal relationships, to detect faults. All these solutions do not consider heterogeneous IoT/sensor deployments, containing a mix of numeric and event-driven sensors.

Several papers [26]–[28] have applied a packet monitoring based approach for network-level detection of 'missing' faults. There are a few recent research on data failures of heterogeneous sensor networks, albeit without the generality of our *DetectIF* approach. [29] deploys special instrumentation for monitoring sensor platform parameters to detect sensor faults in an agricultural setting. [11] assumes an abstract sensor model where multiple sensors measure the same variable, and the inconsistencies among these measurements are used to detect transient faults. Obviously, in instrumented smart homes settings such assumptions do not hold.

Fault Detection in Ambient Assisted Living: As explained in Section V, CLEAN [6] proposes an outlier detection technique to detect sporadic and systematic spurious faults, whereas Idea [3] proposes a permanent missing failure detection method. In Section VI, we have already compared our results with both CLEAN and Idea. DICE [4] proposes a sensor fault detection method based on static modeling of sensor transition probabilities. While DICE technically detects transient faults, it can do so only when certain transition rules, based on these probability models, are violated. Consequently, it often takes up to 30 minutes to detect and identify a fault. *DetectIF*, on the other hand, detects faults independently for each distinct ADL segment. Moreover, sensor correlations and transitions often vary dramatically based on activities and time of day. Hence, without considering the complete range of activation patterns and their contexts, such static probability models are likely to be less useful. SMART and 6thsense [12], [30] perform context-aware fault detection by employing multiple classifiers, each learning activity labels from different subsets of sensors and comparing their outputs. The complexity for constructing the classifier profile grows exponentially with the number of sensors. Further, both approaches treat the classifiers as black boxes; thus, they are able only to detect a single failure at a time.

VIII. Conclusion & Future Directions

We have presented *DetectIF*, a novel unified fault detection approach that can reliably detect a comprehensive set of sensor faults of various durations (sporadic, transient, permanent) and types (missing, spurious, atypical). *DetectIF* builds association rules for fault detection via spatiotemporal correlation of multiple sensors and ADL and augments such rules with several

features related to the patterns of individual sensor activation. *DetectIF* results in significant performance improvement in real-world smart home benchmark datasets: when tested with synthetic fault data, it improves the accuracy of sensor fault detection by ∼30-65%, as well as improves the accuracy of eventual ADL detection by over 30%. Additionally, applying *DetectIF* on existing smart home datasets allows ADL recognition algorithms to achieve 30% higher accuracy, compared to the use of alternative fault detection techniques. Overall, *DetectIF* provides the previously-missing ability to detect *atypical* (a category of false-alarms where the anomaly is in the *firing pattern* of the sensor) faults; its performance gains are more pronounced when such faults last for moderately-long duration (several minutes to hours).

There are, however, several possible future directions for improvement. At present, *DetectIF* does not utilize sensor-specific features (e.g., its mean-time-to-failure, constant offsets, or its voltage response [29]) to effectively define a-priori failure likelihoods; including that in the model (e.g., by adding a multiplicative term to Equation 5) would be an immediate future work. Moreover, *DetectIF* currently detects faults in an independent, *memory-less* fashion on each ADL segment. For faults that are longer in duration or have specific temporal patterns (e.g., periodic every afternoon), incorporating *cross-segment* correlation should further improve *DetectIF's* performance. Additionally, it will be useful to combine user behavior-specific features, and contextually categorize abnormal sensor patterns (e.g. sensor failure due to ambient change, due to behavior change, or due to a serious health crisis for instance fall/fire detection) to further enhance the capabilities of overall smart home assistance system.

IX. Acknowledgement

We thank Intel (India) for supporting the research through PhD fellowship program.

References

[1] V. Rialle, C. Ollivet, C. Guigui, and C. Herv, "What do family caregivers of alzheimer's disease patients desire in smart home technologies?" *Methods of information in medicine*, vol. 47, pp. 63–69, 2008.

[2] S. Sikkes, E. De Lange-de Klerk, Y. Pijnenburg, and P. Scheltens, "A systematic review of instrumental activities of daily living scales in dementia: room for improvement," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 80, no. 1, pp. 7–12, 2009.

[3] P. A. Kodeswaran, R. Kokku, S. Sen, and M. Srivatsa, "Idea: A system for efficient failure management in smart iot environments," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2016, pp. 43–56.

[4] J. Choi, H. Jeoung, J. Kim, Y. Ko, W. Jung, H. Kim, and J. 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)*. IEEE, 2018, pp. 610–621.

[5] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Transactions on Sensor Networks (TOSN)*, vol. 5, no. 3, p. 25, 2009.

[6] J. Ye, G. Stevenson, and S. Dobson, "Fault detection for binary sensors in smart home environments," in *Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on*. IEEE, 2015, pp. 20–28.

[7] M. Flöck, *Activity monitoring and automatic alarm generation in AAL-enabled homes*. Logos Verlag Berlin GmbH, 2010.

[8] S. Munir and J. A. Stankovic, "Failuresense: Detecting sensor failure using electrical appliances in the home," in *Mobile Ad Hoc and Sensor Systems (MASS), 2014 IEEE 11th International Conference on*. IEEE, 2014, pp. 73–81.

[9] Benchmark datasets; datasets for activity recognitions. [Online]. Available: https://sites.google.com/site/tim0306/datasets

[10] Wsu casas dataset. [Online]. Available: http://ailab.wsu.edu/casas/datasets/

[11] J. Park, R. Ivanov, J. Weimer, M. Pajic, S. H. Son, and I. Lee, "Security of cyber-physical systems in the presence of transient sensor faults," *ACM Transactions on Cyber-Physical Systems*, vol. 1, no. 3, p. 15, 2017.

[12] K. Kapitanova, E. Hoque, J. A. Stankovic, K. Whitehouse, and S. H. Son, "Being smart about failures: assessing repairs in smart homes," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 2012, pp. 51–60.

[13] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, vol. 22, no. 2. ACM, 1993, pp. 207–216.

[14] A. S. Crandall and D. J. Cook, "Using a hidden markov model for resident identification," in *2010 Sixth International Conference on Intelligent Environments*. IEEE, 2010, pp. 74–79.

[15] M. Mallick, P. Kodeswaran, S. Sen, R. Kokku, and N. Ganguly, "Tsfs: An integrated approach for event segmentation and adl detection in iot enabled smarthomes," *IEEE Transactions on Mobile Computing*, 2018.

[16] N. Roy, A. Misra, and D. Cook, "Infrastructure-assisted smartphone-based adl recognition in multi-inhabitant smart environments," in *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2013, pp. 38–46.

[17] T. van Kasteren and B. Krose, "Bayesian activity recognition in residence for elders," in *3rd IET International Conference on Intelligent Environments (IE 07)*, 2010, pp. 209–212.

[18] L. Liao, D. Fox, and H. Kautz, "Location-based activity recognition using relational markov networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2005.

[19] P. Rashidi, D. Cook, L. Holder, and M. Schmitter-Edgecombe, "Discovering activities to recognize and track in a smart environment," *Knowledge and Data Engineering, IEEE*, vol. 23, pp. 527–539, 2010.

[20] M. Mourad and J.-L. Bertrand-Krajewski, "A method for automatic validation of long time series of data in urban hydrology," *Water Science and Technology*, vol. 45, no. 4-5, pp. 263–270, 2002.

[21] N. Ramanathan, T. Schoellhammer, E. Kohler, K. Whitehouse, T. Harmon, and D. Estrin, "Suelo: human-assisted sensing for exploratory soil monitoring studies," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2009, pp. 197–210.

[22] L. Fang and S. Dobson, "Unifying sensor fault detection with energy conservation," in *International Workshop on Self-Organizing Systems*. Springer, 2013, pp. 176–181.

[23] S. Guo, Z. Zhong, and T. He, "Find: faulty node detection for wireless sensor networks," in *Proceedings of the 7th ACM conference on embedded networked sensor systems*. ACM, 2009, pp. 253–266.

[24] A. B. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 3, p. 23, 2010.

[25] K. Ni and G. Pottie, "Bayesian selection of non-faulty sensors," in *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*. IEEE, 2007, pp. 616–620.

[26] R. N. Duche and N. P. Sarwade, "Sensor node failure detection based on round trip delay and paths in wsns," *IEEE Sensors journal*, vol. 14, no. 2, pp. 455–464, 2014.

[27] I. C. Paschalidis and Y. Chen, "Statistical anomaly detection with sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 7, no. 2, p. 17, 2010.

[28] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*. ACM, 2005, pp. 255–267.

[29] T. Chakraborty, A. U. Nambi, R. Chandra, R. Sharma, M. Swaminathan, Z. Kapetanovic, and J. Appavoo, "Fall-curve: A novel primitive for iot fault detection and isolation," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2018, pp. 95–107.

[30] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thsense: A context-aware sensor-based attack detector for smart devices," in *26th {USENIX} Security Symposium ({USENIX} Security 17)*, 2017, pp. 397–414.