Detecting and Reacting to Smart Home Novelties

Lawrence B. Holder[1*], Baxter Eaves[2], Patrick Shafto[2,3,4], Christopher Pereyda[1], Brian Thomas[1], Diane J. Cook[1]

[1*]School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA.
[2]Redpoll USA.
[3]Department of Math and Computer Science, Rutgers University, Newark, NJ, USA.
[4]School of Mathematics, Institute for Advanced Study, Princeton, NJ, USA.

*Corresponding author(s). E-mail(s): holder@wsu.edu;
Contributing authors: bax@redpoll.ai; patrick.shafto@rutgers.edu; christopher.pereyda@wsu.edu; bthomas1@wsu.edu; djcook@wsu.edu;

Abstract

To be robust, AI systems need to quickly detect and effectively react to novelties in their environments. Novelty is characterized by a sudden change in the environment where this is little time to collect new data and retrain. This is particularly true for smart home systems, where novelties abound and accurate handling is required for reliable health monitoring and home automation that can react quickly and effectively to the novelties. In this paper, we introduce a Bayesian nonparametric method, called OTACON, for novelty handling. OTACON finds surprising situations using change point detection and adapts accordingly. To evaluate this proposed approach, we design a smart home novelty generator that embeds novelties of varying type and difficulty into CASAS real-world smart home datasets. We observe that OTACON outperforms a state-of-the-art method for eight types of novel scenarios, in some cases even outperforming its own pre-novelty baseline. The results provide evidence that this method can boost AI systems in their ability to handle a variety of unexpected situations.

Keywords: novelty detection, change point detection, smart homes, activity recognition

1 Introduction

The ability to handle novelty is essential for AI systems. Novelty here is characterized by a sudden change in the environment requiring quick adaptation and little time to collect large amounts of new data and retrain. However, many existing systems are brittle [1, 2], in that they can achieve high performance by relying on the assumption that the domain's task stays the same. However, the real world is fraught with novelty. In the perceptual task of image classification, a new class may be introduced that the AI system has never seen before. In an action-based task, other players in the environment exhibit new behavior. Traditionally, the approach in such cases is to collect large amounts of data about the novel phenomenon and adapt the AI systems until performance recovers (i.e., long-term learning [3]). In some environments (e.g., autonomous driving), the AI system does not have the luxury to collect vast amounts of data about a novelty and train over a long period of time.

The goal of DARPA's Science of AI and Learning for Open-world Novelty (SAIL-ON) [4, 5] program was to advance the capability for AI systems to detect, characterize, and accommodate novelty. In this work, we investigate methods for reacting quickly to novelty in the context of a smart home activity recognition task. We introduce a smart home novelty generator for evaluating novelty-aware activity recognition systems and evaluate a new system designed to overcome brittleness by detecting and accommodating multiple types of novelty. In keeping with the SAIL-ON definition, we define novelty as a sudden, unannounced, and persistent change in the classes, attributes, actions, interactions, goals and behaviors of entities in the environment.

We focus here on activity recognition, specifically on identifying what activity is being performed by the inhabitant of a home based on sensor data collected in real time from the home. Such data may be used for health assessment, monitoring, and intervention, as well as security monitoring and automation. Classifying an inhabitant's activities is an essential intermediate step that has been found to improve performance when the activity is included as a feature in the learning task [6]. Still, most activity recognition methods assume the future mimics the past or assume significant data and time is available to adapt to novelty. Nevertheless, novelty can occur quickly. In a smart home setting, new visitors, utility outages, or sudden behavior changes all violate these assumptions and challenge AI systems to continue to perform at reasonable levels in the presence of these novelties.

For the smart home activity recognition task, the feedback from the environment is represented as a set of values from sensors in the environment. While additional activity classes could have been introduced, the generated data is based on real data collected from real smart homes, so we do not have real data on new activities and therefore chose not to explore this type of novelty. However, new sensors may appear in the data, and others disappear, since different homes had different numbers and types of sensors. Ultimately, the novelties will manifest in changes to the number, type and values of the sensor features, and so novelty detection and adaptation here can be viewed as a change in distribution over these features. However, due to the sudden nature of the change and the need to adapt quickly, change point detection followed by retraining are insufficient so handle the novelty in a timely manner.

We propose a new AI system, OTACON, to handle novelty in smart home activity recognition tasks. Additionally, we design a smart home novelty generator that systematically generates different types of novelty, evaluates systems such as OTACON on these novelties, and computes metrics that provide insight into the system's ability to detect and react to novelty. In keeping with the SAIL-ON protocol, the sources of the novelties in the smart home environment were not revealed to the OTACON developers until after the evaluation was completed, which would be the case in a real-world setting.

OTACON is a Bayesian nonparametric method for adding novelty handling capabilities to existing agents. OTACON builds a probabilistic model of the baseline world using the same training data as the task agent, after which it evaluates changes in the model-based surprisal (unlikeliness) of the individual records in the incoming data stream. OTACON exploits the idea that a probabilistic model will attribute different likelihoods to data generated by different models. Using change point detection, OTACON monitors the resulting stream of surprisals, looking for points that indicate a change in the underlying process. Once a surprisal change has been detected, OTACON infers when the change occurred and uses this information to trigger adaptation to the novel world.

The next section discusses related work in open-world novelty detection and reaction, and in smart home environments in particular. Section 3 provides details about the smart home environment and the types of novelties generated in the environment. Section 4 presents the OTACON approach to novelty detection and reaction. Section 5 describes how smart home novelties are generated and the metrics used to evaluate novelty detection and reaction performance. The state-of-the-art approach to activity recognition is also discussed, which is used as a baseline for comparison. Section 6 presents the results of the experimentation. Finally, Section 7 discusses the results and future directions.

2 Related Work

Performing robustly in the presence of novelty, also referred to as open-world AI [3, 7], has received significant attention lately due to the goal of overcoming brittleness in AI systems. Several AI systems have been proposed recently for handling novelty in a domain-independent manner. For example, the Rapid-Learn system [8] integrates planning and learning approaches that learn from failures and transfer knowledge over time to handle novelty. Rapid-Learn has been evaluated on the simulated worlds of Monopoly and Minecraft. The Hydra [9] and OPENMind [10] systems also combine a planner with components that facilitate model diagnosis and revision to adapt to novelty in Angry Birds and Minecraft. In more perceptual domains (e.g., open-set image recognition tasks), OpenHybrid [11] employs a hybrid deep learning encoder and classifier to detect and accommodate novel classes.

Ideas have been explored in the literature that are related to novelty detection. These include detection of anomalies, concept drifts, and change points. Anomaly detection refers to the process of identifying data points or events that significantly deviate from the expected data pattern. Anomalies are generally considered outliers - anomalous data are not consistent with data observed before or after the event

[12]. Numerous statistical and machine learning methods have been introduced to detect these situations to tackle challenges found in fraud detection, security, and healthcare [13]. While anomaly detection is related to novelty detection, they address different situations. Anomalies refer to isolated deviations from expected behavior, typically representing outliers; whereas, novelties represent sustained changes in the environment or task.

Concept drifts are changes over time in the observed data that cause previously learned models to become less accurate or obsolete [14, 15]. In contrast with anomalies, these drifts typically happen more gradually and represent sustained changes in the data. In machine learning, performance drops on previously-successful models may indicate that a concept drift has occurred. In such situations, the model can often recapture previous performance levels by forgetting old data and retraining on new [16, 17]. In contrast, novelty refers to sudden changes in the underlying task or environment the require more rapid detection and adaptation.

Change point detection is also related to novelty detection. Change points are abrupt shifts in the properties of observed data over time. These shifts occur rapidly, thus statistical methods are researched to identify the point in a time series where they occur [18]. Supervised and unsupervised deep learning approaches to change point detection have been developed for high-dimensional, complex data [19], but require significant data to establish a baseline. Change point detection typically focuses on changes in a small set of time series features; whereas, novelties such as subtle changes in the environment, may not significantly change small sets of features, but can significantly impact the performance of the AI system operating in the environment.

Systems for evaluating an AI method's open-world novelty capabilities are also increasingly available. In addition to the smart home activity recognition task we describe in this paper, the same novelty generator can be used to create novelties for the CartPole3D [20] and VizDoom [21] domains. Other novelty generators have followed a similar design for additional domains, such as Angry Birds [22], Minecraft [23], and Monopoly [24]. Evaluation of open-world novelty in the image recognition task has also prompted numerous challenges (e.g., COCO [25] and DIAS [26]).

In the context of smart homes, some attempts have been made to detect novelties using anomaly detection. While these methods neither characterize nor adapt to the novelties, they do introduce approaches to finding anomalies specific to such data. Detecting anomalous behavior can also detect some novelties, but these methods assume the underlying task has not changed, which is not the case for novelties in this work. Alaghbari et al. [27] create a deep network to look for activity patterns that deviate from an established norm and illustrate the technique on sample smart home data. Dahmen et al. [28] tailor an anomaly detection technique to find deviations from normal that indicate a clinical-relevant change in behavior. Here, clinician feedback guides the selection of unsupervised anomaly detection methods and hyperparameters to improve performance. While some detection methods have been considered, Chatterjee et al. [29] note that anomaly detection for smart homes and the larger IoT field is still in its infancy, with a need for unsupervised approaches and techniques that are not tailored to a particular piece of information such as energy consumption. In

contrast to these earlier methods, this paper focuses on detecting a wider variety of smart home novelties and reacting to such heterogeneous situations.

3 Smart Homes

A smart home is a physical environment equipped with sensors and computational components. The home acts as an intelligent agent, reasoning about the physical environment and its inhabitants to assess their state and suggest or take actions to improve their state. Because pervasive computing and machine learning technologies can transform any residence into a smart home, researchers have been uncovering their ability to model and recognize normal behavior [30] and quantify changes in behavior [31]. Building on this foundation, smart homes have been designed to assess a person's cognitive health [32, 33], detect exacerbations of chronic conditions [34], and design behavior-aware health interventions [35]. The same foundation provides a basis for monitoring the safety of the home [36, 37] and analyzing the relationship between behavior patterns and factors such as energy consumption [38] and air quality [39]. Ultimately, a home can automate selected actions in response to a stated objective, such as minimizing energy footprint [40, 41].

Because diverse varied capabilities depend on modeling, recognizing, and tracking routine behavior, smart home methods are typically not robust to behavior changes or changes in the environment. Much of the software will fail if, for example, there are unexpected visitors in the home, a sensor fails, or a person abruptly changes their behavior patterns. Given the strong tie between expected behavior and smart home success, we choose this problem as a testbed to design and evaluate a novelty agent.

3.1 CASAS Smart Home in a Box

In this project, we utilize the CASAS smart home infrastructure to collect data, embed novelties, and assess how effectively an agent can detect, characterize, and react to such novelties. The CASAS Smart Home in a Box (CASAS SHiB) [42], shown in Figure 1, contains a collection of ambient sensors, networking components, and a Raspberry Pi that collects, timestamps, and stores generated sensor readings. Sensors generate readings whenever they detect a change in state. Several downward-facing passive infrared (PIR) motion detectors are placed with removable adhesive strips on ceilings in each room and in the refrigerator. These generate a reading whenever motion is initiated in their 2m-diameter field of view or motion has stopped (has not been detected for 1.25 seconds). Magnetic door sensors are attached to exterior doors and cabinets that contain key items such as medicine dispensers. These generate a reading when the door state change from open to closed or vice versa. Additionally, the SHiB includes ambient light and temperature sensors that generate readings when there is a sufficient change in value.

As shown on the right in Figure 1, the smart home collects readings each time a sensor detects a change in state. Each reading includes a corresponding date, time, sensor identifier, and current sensed state (ON/OFF for motion, OPEN/CLOSED for doors, and numeric values for light and temperature sensors). Researchers analyze these readings to understand activities within the home and use the models for home

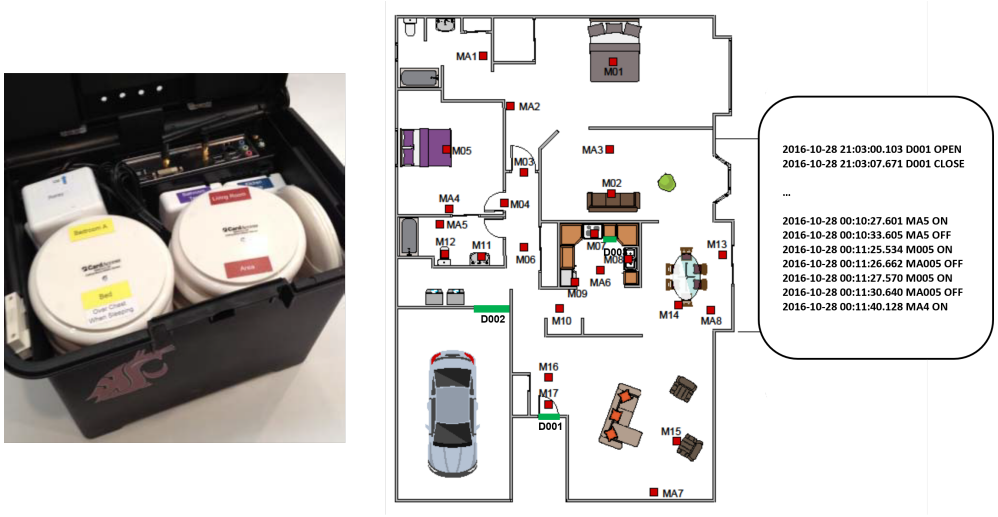


Fig. 1: CASAS SHiB. Downward-facing passive infrared motion detectors are installed inside the refrigerator, above each bed, each seating area, the kitchen sink, and the stove. Area motion detectors are placed in each large room. Magnetic door closure sensors are attached to external doors and selected cabinets. As sensors generate readings, the smart home adds timestamps and sensor identifiers to the readings and stores the data for later analysis.

monitoring and automation. A variety of machine learning techniques have been introduced for this task. These methods typically process a portion of the data at a time by moving a sliding window over the readings. Recent algorithms handle increasingly complex situations, including lack of training data and presence of multiple inhabitants [43, 44]. Once typical activity behavior is modeled reliably, corresponding smart home functions such as health assessment, security monitoring, and energy-efficient automation can be added.

3.2 Smart Home Novelties

While progress has been made in smart home design and analysis of smart home data, many of the current research methods assume that the underlying smart home processes are stationary. This includes the environment conditions and the behavior routines exhibited by inhabitants. As a result, models are typically trained and used for as long as needed, without revision. This assumption contradicts the real world, in which smart home agents require life-long learning.

In response to the need for continually-adaptive smart homes, we are interested in determining whether an agent can detect, characterize, and respond to smart home novelties. For this task, we created a smart home novelty testbed. The challenge for

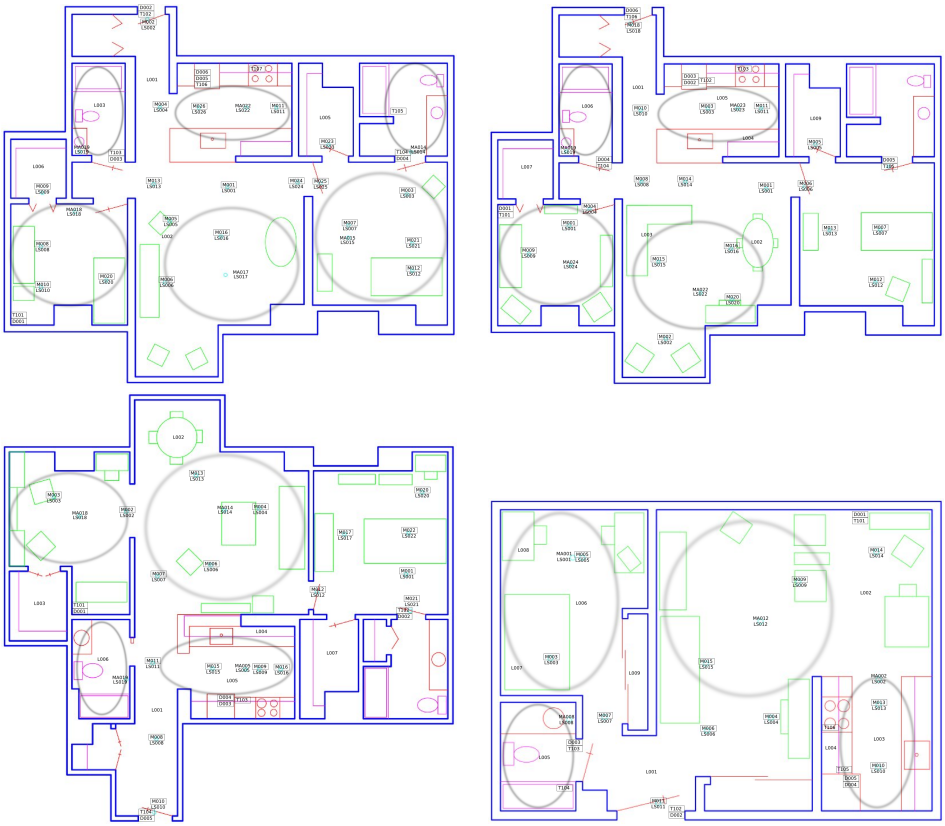


Fig. 2: Smart home testbed floorplans and sensor positions. Sensor labels starting with "M" are motion directors, "D" represents door sensors, "LS" represents light sensors, and "T" represents temperature sensors. From upper left, the testbed homes are labeled SH1, SH2, SH3, and SH4.

an agent in this testbed is to recognize an inhabitant's current activity based on timestamped sensor data. Testbed data are based on actual sensor readings collected in four smart homes. The home floorplans and locations of sensors are provided in Figure 2. In these homes, 11 activity classes are defined. These are *bed toilet transition*, *cook*, *eat*, *enter home*, *leave home*, *personal hygiene*, *relax*, *sleep*, *take medicine*, *wash dishes*, and *work*. Ground truth activity labels are provided for the testbeds using external annotators, as described elsewhere in the literature [18]. The ground truth labeling rules and process do not change during the experiment, before or after novelty introduction.

The feedback from the environment is represented as a set of values from sensors in the environment. While additional activity classes could have been introduced, the data is based on collections from real smart homes, so we do not have real data on new activities and therefore chose not to explore this type of novelty. However, new sensors

may appear in the data, and others disappear, since different homes have different numbers and types of sensors. The novelties manifest as changes to the number, type and values of the sensor features, and so novelty detection and adaptation here can be viewed as a change in the feature space and the distribution over these features.

For our experiments, we analyze 200 days of data from each smart home. Each day, or episode, is presented to the agent as a sequence of sensor vectors. The sensor vectors are not provided at a fixed frequency but only when any sensor value changes. Thus, one day's episode may contain a different number of activity recognition opportunities than another. Generally, the number of recognition opportunities varies little between days, though there is some variance due to the differences in homes and inhabitant behaviors. After each activity recognition opportunity (i.e., the agent is presented with a timestamped sensor vector), the agent returns the predicted activity. After the prediction is generated, the smart home environment provides the agent with its accuracy so far, i.e., the fraction of recognition opportunities for which the agent correctly classified the activity. The final performance of the agent for the episode is represented as the accumulated accuracy after the last prediction for the day.

At a randomly-selected point between days 30 and 50, a novelty is injected into the data.[1] For a novelty detection agent to be useful in a practical smart home setting, the agent needs to be able to respond to different types of novelties and different levels of difficulty. In the SAIL-ON program, eight types of novelty were defined to cover a variety of changes to the environment. Therefore, we create eight types of novelties for the smart home testbed. Each novelty type is further decomposed into three levels of difficulty: easy, medium, and hard. Difficulty here relates to the adaptation of the agent, not the detectability of the novelty. In fact, more difficult novelties tend to be easier to detect, because they result in a more dramatic change in agent performance when introduced. The novelty types with accompanying descriptions and difficulty levels are described below and summarized in Table 1. During a trial, the agent is not given any information about the type of novelty or even the presence or absence of novelty. The novelty descriptions were not revealed to the novelty-aware agent development team until the experimental evaluation was completed.

Novelty type 1 focuses on changes in objects (i.e., sensors) in the environment. The motivation is the case where new sensors are introduced into the environment. In our implementation of this novelty, some randomly-chosen sensors were disabled (i.e., they always return a value of 0) in the pre-novelty episodes, and in the novel episodes, some of these sensors were re-enabled. The difficulty depends on the number of re-enabled sensors, i.e., the more sensors re-enabled, the more difficult is the agent's adaptation. For this novelty type, while the novelty can be detected more easily based on data drift, there is no change in performance if the agent continues to use its current model, which is likely to be based on sensors enabled only in pre-novelty. However, there is an opportunity for the agent to improve performance by retraining on the additional sensor information once enough data is available. This object novelty type could also be implemented by disabling sensors, but we use this scenario in novelty type 4, where sensors in a particular area are disabled.

[1] Additional information is available at https://github.com/holderlb/WSU-SAILON-NG/tree/master/domains/smartenv.

Type	Focus	Homes	Description and Difficulty Levels
1	Objects	SH3	Increased sensor access
			Easy: *5 additional sensors*
			Medium: *10 additional sensors*
			Hard: *30 additional sensors*
2	Agents	SH4	Shift times to future episodes
			Easy: *Move readings 30-60 days in the future*
			Medium: *Move readings 120-150 days in the future*
			Hard: *Move readings 260-300 days in the future*
3	Actions	SH1, SH2	Different inhabitant & behavior, same floorplan
			Easy: *25% data from different inhabitant*
			Medium: *50% of data from different inhabitant*
			Hard: *100% data from different inhabitant*
4	Relations	SH3	Sensors turned off in one room
			Easy: *5 sensors turned off in study*
			Medium: *6 sensors turned off in living room*
			Hard: *7 sensors turned off in bedroom*
5	Interactions	SH1, SH2	Visitor in home
			Easy: *1 visitor*
			Medium: *2 visitors*
			Hard: *3 visitors*
6	Rules	SH2	Temperature decrease triggers motion sensor events
			Easy: *small decrease (5% chance triggers)*
			Medium: *medium decrease (10% chance triggers)*
			Hard: *large decrease (15% chance triggers)*
7	Goals	SH3	Attempt to mimic leave home activity
			Easy: *6 leave home events added to day*
			Medium: *12 leave home events added to day*
			Hard: *18 leave home events added to day*
8	Events	SH3	Daily activities are fast-forwarded
			Easy: *Fast-forward by factor of 2*
			Medium: *Fast-forward by factor of 5*
			Hard: *Fast-forward by factor of 10*

Table 1: Novelty types, novelty focus, and smart home datasets to which novelties are applied. Each novelty type is further characterized by three difficulty levels (easy, medium, and hard).

Novelty type 2 focuses on changes in an inhabitant's routine behavior. Here, a behavior pattern is shifted into the future at the time novelty is introduced, though the rest of the environment remains the same. Novelty type 2 emulates the common situation where the inhabitant's behavior changes over time, with the amount of change increasing with time length. The difficulty level corresponds to the extent of this shift in time: 30-60 days (easy), 120-150 days (medium), 260-300 days (hard). More extensive shifts in time (and thus behavior) are easier to detect, but more difficult in terms of adaptation. This situation occurs in the actual smart homes where we collected data, because behavior for the older adult inhabitants changed with increasing age and onset of cognitive or physical health changes. A common type of observed changes in these situations was less time spent outside the home during the day or more frequent sleep interruptions during the night. The activity recognition agent is

a passive observer that makes predictions about activities. However, the agent's predictions have no effect on the world, nor can this agent affect the world in any way. This scenario is usually referred to as a perceptual task; whereas, in other action-based tasks (e.g., a player in a first-person shooter game), the agent can take actions that affect the world.[2]

Novelty type 3 focuses on changes in the actions performed by smart home inhabitants. This novelty is implemented by intermixing data the inhabitant of SH2 into the data of the data of the inhabitant of SH1. Floorplans SH1 and SH2 were the same. SH1's inhabitant was a healthy older adult, and SH2's inhabitant was a cognitively impaired older adult. The difficulty level was implemented by the extent of the intermixing of data: 25% (easy), 50% (medium), 100% (hard). Based on data collected in real CASAS smart homes, the newly-introduced data reflect different types of inhabitants (e.g., older adults, students, cognitively impaired individuals), which in turn allowed a categorization of how the difficulty level would change at the point of novelty introduction. In this case, the increased intermixing of impaired behavior into healthy behavior is easier to detect, but harder in terms of adaptation.

Novelty type 4 focuses on changes in the relationships between agents and objects in the world. Unlike novelty type 1, where sensors are randomly enabled, here sensors are disabled on the basis of their relationship to the inhabitant's behavior in the home. In particular, sensors that are in the same room are disabled, and the room is chosen based on the extent of the inhabitant's interaction with that room. The inhabitant interacted least with the study in SH3, so the easy difficulty was implemented by disabling (setting to zero), the five sensors in the study. The medium difficulty was implemented by disabling the six sensors of the living room, which was used by the inhabitant more often than the study. The hard difficulty was implemented by disabling the seven sensors of the bedroom, which was used more than the living room. Disabling the sensors in a room has an immediately impact on performance, because those sensors could more easily be correlated to the actions in this room (e.g., sleeping usually occurs in the bedroom), but alternative patterns can be learned over time, such as sleep occurs after a certain time and when no other sensors are triggered in the home.

Novelty level 5 focuses on change due to interactions between multiple inhabitants in the smarthome. This novelty was implemented by splicing in real data observed when one or more visitors were present in the home, such as service providers or family members. The difficulty level is implemented based on the number of visitors whose data is included in the post-novelty data stream. Since activity recognition performance is measured in reference to the permanent inhabitant of the home, the presence of visitors will trigger new data that may incorrectly be attributed to the inhabitant.

Novelty type 6 focuses on a change in the rules governing how the world works. In more simulated domains, this novelty can be implemented in creative ways (e.g., change the gravitational influence). Given that the smart home domain is derived from real data from real homes, the choices for rule changes are constrained. We

[2] Our novelty generator includes two action-based tasks, CartPole and VizDoom, in addition to the smart home task described in this paper.

therefore design this novelty using the very real possibility of a temperature decrease due to weather or a change in the heating and air conditioning in the home. Motion sensor sensitivity is affected by temperature, and lower temperatures lead to more frequent triggering of the motion sensors. The difficulty levels are based on the extent of the temperature decrease, which is implemented by additional random triggering of motion sensors throughout the home: 5% chance of triggering (easy), 10% chance (medium), 15% chance (hard). The increase of noise into the motion sensors increases the difficulty of the activity recognition task, but the generally low chances of noise still allow the novelty-aware agent to adapt over time.

Novelty type 7 focuses on changes in the goals of the inhabitant. One goal change in the elderly inhabitants of the smarthomes was the attempt of the inhabitant to appear to leave the home more often to appease a caregiver wanting the inhabitant to go on more walks outside the home. We simulated this behavior by inserting sensor data for the leave-home and enter-home activities into the sensor stream but did not otherwise modify the data. The challenge for the activity recognition agent is to determine that the inhabitant is not actually performing the leave-home and enter-home activities but is continuing the activity initiated before this interruption. These false leave-home/enter-home events occurred much closer in time than a real walk, which typically takes about an hour, and were randomly distributed throughout the day. The difficulty levels for this novelty are based on the number of false leave-home/enter-home events inserted into the data: 6 additional leave/enter home events per day (easy), 12 events added per day (medium), 18 events added per day (hard). The novelty-aware agent will likely incorrectly predict these are true leave/enter home activities, but can learn to distinguish them based on their shorter duration and atypical times during the day.

Finally, novelty type 8 focuses on external events that trigger a change in inhabitant behavior. We implemented this novelty by fast-forwarding the sensor stream, which simulates the inhabitant having increased energy and performing more activities at an increased speed. Specifically, we compress each day's (episode's) data by a factor $c < 1$, and then concatenate $1/c$ copies of this compressed data to fill in the entire day. For example, $c = 0.5$ results in the day's data being compressed from 24 hours to 12 hours, and then the day is composed of two copies of the compressed data. The difficulty levels are implemented based on the value of c: $c = 0.9$ (easy), $c = 0.5$ (medium), $c = 0.1$ (hard). While the decreased duration and atypical start times of the activities will reduce activity recognition performance, the increase in sensor readings also provides an increase in the number of activity recognition opportunities; thus, post-novelty performance could potentially exceed pre-novelty performance.

Data that we analyze in our experiments are drawn from real-world datasets. However, we need to give the novelty agent data for an arbitrary number of episodes (days). To meet this need, we employ SynSys a smart home synthetic data generator [45]. SynSys learns a model from the real sensor data, then generates an arbitrary amount of realistic synthetic data. This method demonstrated successful generation of realistic sensor sequences for a collection of 10 smart home datasets [45].

4 OTACON: A Novelty Aware Agent

OTACON offers a bolt-on framework for adding novelty handling to existing agents (see Figure 3). OTACON uses a probabilistic model to evaluate the surprisal of data as they are streamed to a task agent, as well as the surprisal of task agent outputs (such as label probabilities) and other meta-quantities (such as task performance). OTACON uses online change point detection to identify systematic changes in surprisal over time, which indicate changes in the data generating process, which indicates novelty. Once novelty is detected, OTACON triggers adaptation, which, for the CASAS agent, means replacing the training data with that experienced post-novelty and re-training. Algorithm 1 and the following discussion provide a more-detailed description of the OTACON agent.

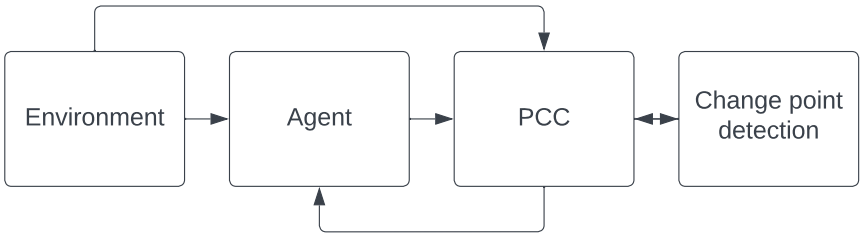


Fig. 3: OTACON architecture. The environment provides data to the task agent and to the probabilistic cross-categorization (PCC) engine. The agent provides information to PCC, which interfaces with the task agent via adaptation. PCC sends quantities to change point detectors which detect novelty as systematic changes in PCC outputs. Change point detectors trigger novelty adaptation in PCC and the task agent by way of PCC.

4.1 A statistical definition of novelty

Open world novelty can be formalized as a change in the data generating process. An agent is trained on data from process A, and at some future point, the process A changes into a distinct process B, potentially invalidating the agent trained on A. Detecting novelty is then a problem of detecting when the data-generating distribution has changed from streaming data. OTACON exploits the fact that the likelihoods of all observations $X = \{x_1, \ldots, x_n\}$ are identical under two models A and B if and only if A and B are identical. That is, $p(x|A) = p(x|B)\ \forall x \implies A = B$.

4.2 Modeling open worlds

OTACON uses probabilistic cross-categorization (PCC; [46]) to create an approximation of A, $\hat{A}$ from the agent training data. PCC is a hierarchical Bayesian non-parametric model for tabular data. Given a table with n rows and m columns, PCC uses a Dirichlet process to group the m columns into $1, \ldots, m$ *views* in partition $\mathbf{z}$, where $\mathbf{z}_v$ is the set of columns in view v; and within each view, uses another Dirichlet process, with concentration parameter $\alpha \in \mathbb{R}_+$, to group rows into $1, \ldots, n$ *categories* in partition $\mathbf{c}$, where $\mathbf{c}_{v,k}$ is the set of rows assigned to category k in view v. Bayesian non-parametrics are a natural way to describe open world learning because they do not assume a specific parameterization of the world. Dirichlet processes, in particular, elegantly handle novel observations by considering the probability that observations were generated from unseen categories. While changing the number of activity categories is not one of the novelties we test, the developers of OTACON did not know that. OTACON was designed to handle such novelties, which could show up in other real-world settings.

The PCC joint distribution is

$$p(x, \mathbf{z}, \mathbf{c}, \phi, \alpha) = p(\mathbf{z}|\alpha) \prod_{v=1}^{|\mathbf{z}|} p(\mathbf{c}_v|\alpha) \prod_{j \in \mathbf{z}_v} \prod_{k=1}^{|\mathbf{c}_v|} p(\{x_{i,j} : \mathbf{c}_{v,i} = k\}|\phi_j), \tag{1}$$

where ϕ_j is the prior on the component models of column j.

4.3 Detecting novelty

In OTACON, PCC sits beside a task agent, which in this case is a re-implementation of the baseline agent above. As pre-processed data, x, stream to the task agent, OTACON evaluates the surprisal of each observation, $-logp(x|\hat{A})$. The resulting surprisal stream is passed to online change point detection. For this work we use a Gaussian-Process-based detector [47], implemented in the `change point` rust crate[3], which, is often more successful at detecting gradual changes than Bayesian online change point detection [48]. The change point detector is explicitly looking for changes in the surprisal distribution over time. A change in the surprisal distribution indicates a change in the data generating process, which indicates that novelty has occurred.

4.4 Adapting to novelty

Once OTACON has inferred that novelty has been introduced at some point, it uses the runlength probabilities from the change point detector to infer at what specific time point the novelty was introduced. The runlength probabilities measure the probability distribution $P(r_t|s_{1:t})$ over the runlength r_t, which is the number of episodes since the last change point at time/episode t [48], given the sequence of suprisals s_i since the first episode in the trial. The number of episodes back in time where the novelty is likely to have been first introduced can be estimated as the maximal value of the runlength distribution.

[3] Available at https://crates.io/crates/changepoint/

The main while loop in Algorithm 1 iterates over each instance x_i from the environment, computes the surprisal value s_i, and feeds the surprisal value into the change point detector C using $C.observe(s_i)$. The change point detector updates the run-length probabilities and returns the probability that a change has occurred. If that probability is high enough, then OTACON transitions to "adapting" mode and the classifier model F is retrained. The classifier is then used to label the new instance, and the label is returned to the environment for evaluation.

To avoid performance loss due to *nuisance novelties* (novelties that do not affect task performance), adaptation is triggered only if the agent performance has been negatively affected by the novelty. Adaptation occurs by replacing data in the training set with post-novelty data and periodically re-training the task agent. Once in "adapting" mode, retraining is triggered each time the agent has encountered τ additional training examples. The τ threshold is trained based on a set of mock novelties designed by the OTACON team.

The task agent is a re-implementation of the random forest classifier, as in the baseline agent, but more tightly integrated with the OTACON system. The OTACON classifier was initially trained on the same data as the baseline agent, but additional parameter tuning was performed to optimize the classifier for this data. This is why the OTACON system outperforms the baseline agent pre-novelty in some cases.

5 Novelty Evaluation

The evaluation of novelty-aware agents is provided by the publicly-available Smart Environment Novelty Generator (SmartEnvNG [49]. To evaluate a novelty-aware agent, we present a sequence of episodes to the agent. Each episode represents one day in the smart home, and the sequence of episodes represent contiguous days. The first few episodes contain no novelty. When novelty is introduced, all generated episodes contain one type of novelty. Evaluating an agent's ability to handle multiple types of novelty in a single episode is an interesting next step, but is left for future work.

The main performance measure that the agent is trying to maximize is the activity recognition accuracy over the instances in an episode. In our experiments, we included 200 episodes in each trial, and novelty is introduced at a uniform random point between episodes 30 and 50. The agent does not know if the episodes contain novelty or not. For each novelty type, the agent is evaluated on 90 trials, 30 for each of the three difficulty levels (easy, medium, hard). Performance is reported as an average over these trials. While the performance plots represent an average over the trials, the individual performance curves are first aligned so that the episode of novelty introduction is at the same point on the x-axis. This is why episode numbers are not indicated on the x-axes of our performance plots. The data to reproduce the trials used in these experiments is available from the SmartEnvNG repository [49].

5.1 Performance metrics

Evaluating an agent's performance in the presence of novelty requires more than the standard activity recognition metrics like precision, recall, accuracy, and AUC. Specifically, we want to evaluate an agent's ability to both detect and react to novelty and

Algorithm 1 OTACON

Input: Training data, x_{train}, processed for agent F. A stream, x_{new}, of processed episode data. A novelty threshold, $\epsilon \in [0, 1]$. The number of steps, τ, between post-novelty agent updates.

Yields: A generator of outputs from the task agent

$C \leftarrow$ new change point detector
$\hat{A} \leftarrow A|x_{train}$ ▷ PCC Model trained on x_{train}
$\hat{F} \leftarrow F|x_{train}$ ▷ Task Model trained on x_{train}
adapting $\leftarrow$ False
$x^* \leftarrow [\,]$ ▷ Post-novelty data
while $x_i \leftarrow next(x_{new})$ **do** ▷ Read the data stream
  append x_i to x^*
  if not adapting **then**
    $s_i = -\log p(x_i|\hat{A})$ ▷ Surprisal of x_i under PCC model
    $p_{change} \leftarrow C.observe(s_i)$ ▷ Probability a change has occurred
    if $p_{change} > \epsilon$ **then**
      $j \leftarrow$ index of change point inferred by C
      $x^* \leftarrow x^*[j:]$ ▷ remove all pre-novelty data from x^*
      adapting $\leftarrow$ True
    end if
  else
    $n^* \leftarrow |x^*|$
    if $n^* > \tau$ and $n^* \mod \tau = 0$ **then**
      $\hat{F} \leftarrow F|x^*$ ▷ Update/Retrain the task agent
    end if
  end if
  yield $\hat{F}(x_i)$ ▷ Produce output given x_i
end while

do so quickly. The SAIL-ON program developed several metrics to capture the different aspects of performance, which are described in Table 2. The first three metrics evaluate an agent's ability to detect novelty. One aspect of detection is the concept of a *correctly detected trial* (CDT), which means that the agent detected novelty after novelty was introduced and not before. The CDT metric in Table 2 measures the fraction of trials that are CDTs. The FN-CDT metric measures the number of episodes that elapsed after novelty was introduced before the agent detected novelty, but only for trials in which the agent eventually detects novelty. The third metric (FP) measures the fraction of trials in which the agent detects novelty before it was actually introduced. So, the first three detection metrics measure both how accurately and how quickly the agent detects novelty. A successful agent should achieve a high CDT, while maintaining low FN-CDT and FP.

An agent may achieve a high CDT and low FP by prolonging detection in order to maximize the chance of detection after novelty introduction. However, this will result in a poor (high) FN-CDT score. Also, the agent typically does not know when,

or if, novelty is coming, so even a prolonged wait may still result in a premature detection. In the experiments, and supported by the SmartEnvNG system, feedback on whether the classifier is correct is only provided after the agent detects novelty. So, artificially prolonging detection to improve detection scores will result in poor reaction performance due to lack of feedback about the novelty.

The remaining five metrics measure how well the agent reacts to novelty in terms of performance in the environment, which in this case is activity recognition accuracy. One unique aspect of these metrics is that the novelty-aware agent's performance is measured relative to the performance of a state-of-the-art (SOTA), non-novelty-aware agent. The idea is that the novelty-aware agent needs to perform at least as well as the SOTA agent before novelty is introduced. After novelty is introduced, performance for both the SOTA and novelty-aware agents will likely decrease, but ideally, the novelty-aware agent will eventually recover and exceed the performance of the non-novelty-aware SOTA agent. The ONRP and INRP metrics measure this by computing the novelty reaction performance (NRP) of the agent, which is defined as the ratio performance of the agent after novelty is introduced, or post-novelty, to the SOTA agent's performance before novelty is introduced, or pre-novelty. Thus,

$$NRP = \frac{P^{\alpha}_{post}}{P^{\beta}_{pre}}$$

where α represents the novelty-aware agent, and β represents the baseline SOTA agent. ONRP measures this ratio based on the *overall* post-novelty performance, and INRP measures this ratio based only on the *initial* first few post-novelty episodes. In our experiments, we used 10% (or 20) episodes to measure this initial NRP.

The OPTI, IPTI, and APTI metrics capture the performance task improvement (PTI) due to the novelty-aware agent compared to the baseline SOTA agent. While NRP compares post-novelty performance to pre-novelty performance, PTI is focused only on post-novelty performance. PTI is computed as the ratio of the novelty-aware agent's performance to the sum of the novelty-aware agent's performance and the SOTA agent's performance. Thus,

$$PTI = \frac{P^{\alpha}_{post}}{P^{\alpha}_{post} + P^{\beta}_{post}}$$

i.e., the fraction of post-novelty performance due to the novelty-aware agent α. We measure this for the overall post-novelty performance (OPTI), the *initial* 10% of episodes post-novelty (IPTI), and the final or *asymptotic* 10% of episodes post-novelty (APTI).

The SmartEnvNG system computes all the metrics in Table 2 and several additional metrics (performance before and after novelty, significance of performance drop, and the percentage of pre-novelty true negatives) to provide a complete picture of an agent's performance. Many of these performance metrics are computed relative to a baseline agent, so the SmartEnvNG provides baseline agents for both activity recognition and novelty detection. In addition to the metrics, the SmartEnvNG system produces plots of the agent's performance relative to a baseline agent's performance, similar to those in Figures 4 and 5.

Metric	Description
FN-CDT ↓	False negatives (FN) for correctly detected trials (CDTs)
CDT	Fraction of CDTs
FP ↓	Fraction of trials with at least one false positive (FP)
ONRP	Overall novelty reaction performance (NRP) as a fraction of baseline pre-novelty performance
INRP	NRP for the first 10% of episodes after novelty introduction
OPTI	Overall performance task improvement (PTI) for all episodes after novelty introduction
IPTI	Initial PTI based on first 10% of episodes after novelty introduction
APTI	Asymptotic PTI based on last 10% of episodes in a trial

Table 2: Performance metrics used to evaluated novelty detection (FN-CDT, CDT, DP) and reaction (ONRP, INRP, OPTI, IPTI, APTI). For FN-CDT and FP, smaller is better (↓).

5.2 Baseline activity recognition

The baseline agent is implemented using the random forest classifier [50], which continues to perform among the best approaches in the smart home domain when there are numerous complex activities [6, 18]. We used the random forest classifier from the Scikit Learn Python package version 1.5.1 with default parameters except for setting the criterion to "entropy", class weight to "balanced", and min samples split to 20. These parameters were found to work well across several smart home datasets. The baseline agent uses a fixed-length sliding window to label activities. The agent moves a window of fixed size (we use a window of size 30 in our experiments) over the sensor readings. Once a vector of features is extracted from the window, the feature vector is fed to the random forest classifier. The classifier generates an activity label, which is assigned to the last reading in the window. Finally, the window advances by one sensor reading and the process repeats.

Type	Feature	Description
time	day of week	$0 \ldots 7$
	hour of day	$0 \ldots 23$
	seconds past midnight	$0 \ldots 86,399$
window	window duration	seconds from first to last reading
	last sensor	sensor identifier
	dominant sensor	identity of most frequent sensor in current window
	previous dominant sensor	identity of most frequent sensor in previous window
	last location	room location of last sensor in window
	window complexity	entropy calculation of readings in window
	activity level change	difference in duration between first and second half of window
	transitions	number of location changes in window
sensors	reading count	number of readings for each sensor in window
	quiet time	elapsed time since previous reading for each sensor

Table 3: Features that are extracted from each window of smart home data.

Table 3 lists the features that are used by the baseline activity recognition algorithm. Because these sensors generate readings when there is a detected change in

state, rather than at regular time intervals, the timing of events within each window is insightful. A higher frequency of readings (equivalently, a shorter window duration) indicates a higher activity level in the home. Similarly, the amount of time that has elapsed since a sensor last generated a reading provides important context. For example, long delays between motion sensor readings in the bedroom may indicate deep sleep, as opposed to fitful sleep or reading in bed.

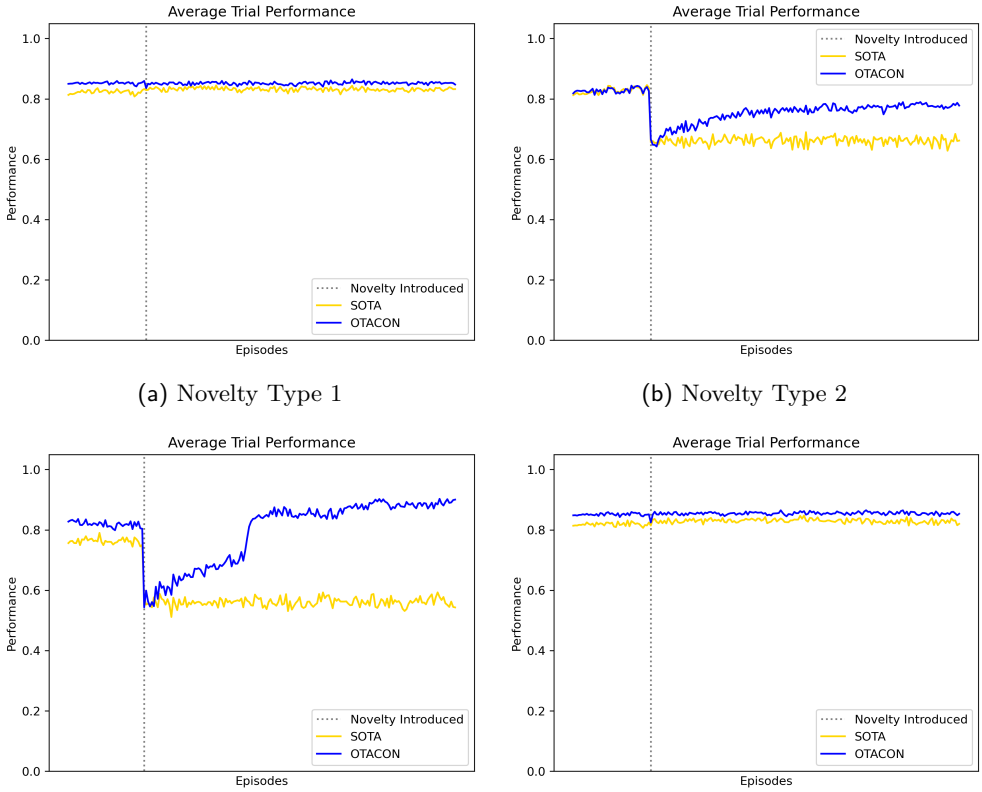
The baseline agent was trained on non-novel data sampled from the datasets for each of the four smart home environments shown in Figure 2. The performance of the baseline agent is shown in the performance plots as the "SOTA" agent along side the OTACON agent in Figures 4 and 5. The OTACON agent was also trained on non-novel data, but random initial conditions resulted in slightly different pre-novelty performance as seen in the performance plots. The baseline agent also has no novelty awareness and is not retrained at any point during the experiment. Thus, the baseline agent's performance does not recover post-novelty.

5.3 Additional Baselines

As discussed in the Related Work section, typical approaches to novelty detection involve concept drift detection or data drift detection. A popular approach for concept drift detection is the Drift Detection Method (DDM) [51]. This method looks for significant changes in the performance of a classifier. We show results using DDM for novelty detection in the next section, but as a preview, due to the high variance in classifier performance across trials, the DDM was prone to earlier detection. A popular approach for data drift detection is the Incremental Kolmogorov-Smirnov test (IncKS) [52]. This test looks for significant changes in the features (see Table 3. While IncKS is designed to look for drift in each feature individually, this also led to early detection in most trials. But if the test is extended to require data drift in a significant percentage of features, the novelty detection is more robust. OTACON still outperforms these methods due to its focus on the joint distribution across features, rather than individual features.

We also tested an ideal adaptation approach, called IDEAL, that retrains the random forest classifier after each episode, regardless of whether novelty is present or detected. IDEAL is trained on about 50,000 examples for each smarthome testbed, same as OTACON and the SOTA baseline. But IDEAL is allowed to accumulate additional examples, up to 100,000 per testbed, and retrains on the latest 100,000 after each episode. This results in an advantage over the OTACON system, since OTACON only receives the correct classification, after it detects novelty. Both approaches receive the correct answer as feedback according to a budget that is set to 50%, but is variable with the SmartEnvNG system. The number of classifications generated per episode varies from a few hundred to over 1,000. The IDEAL approach shows the best possible adaptation, and the results show that OTACON achieves this level on most tasks.

These additional baselines (DDM, IncKS, IDEAL) are available for use within the SmartEnvNG system. The DDM and IncKS methods are implemented using the Frouros drift detection package [53], which includes many more methods that can be utilized within a novelty-aware agent evaluated using the SmartEnvNG system.



(a) Novelty Type 1

(b) Novelty Type 2

(c) Novelty Type 3

(d) Novelty Type 4

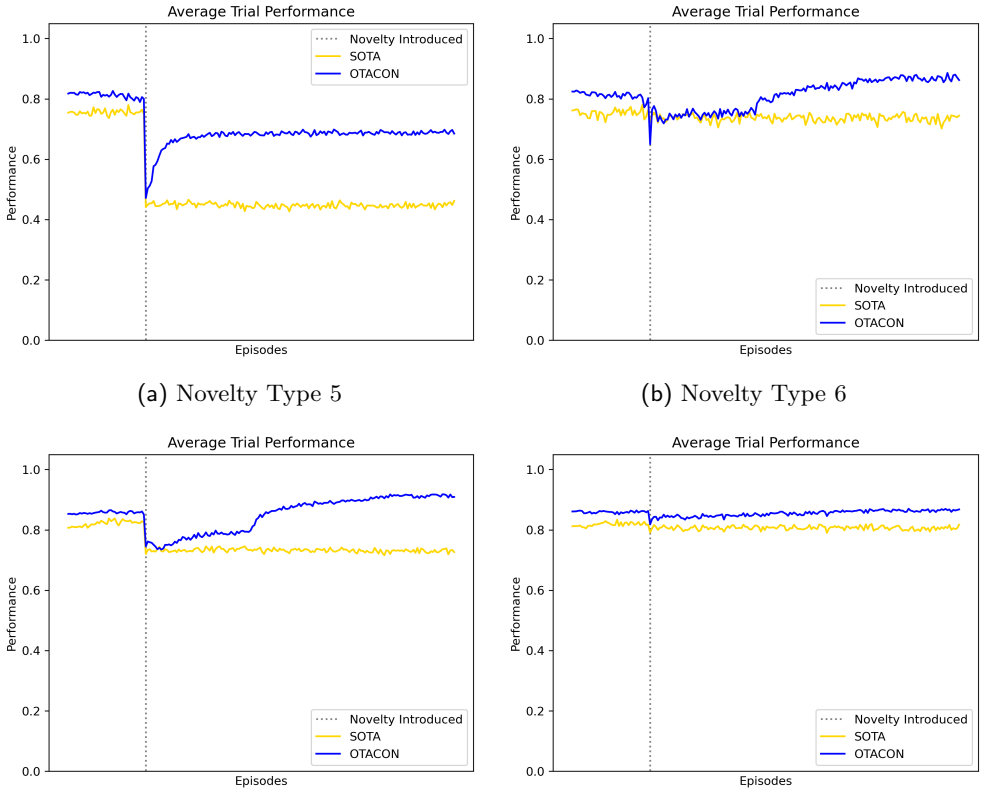
Fig. 4: Performance across novelty types 1-4. Performance is measured as the activity recognition accuracy over the instances in an episode, or single day, in the life of the smart home.

6 Experimental Results

This section shows the main experimental results for the OTACON system on the eight novelty levels from our smart home testbeds. For comparison, we also show results from the additional baselines (DDM, IncKS, IDEAL).

6.1 OTACON Results

Figures 4 and 5 show the plots of OTACON's performance, relative to the SOTA non-novelty-aware baseline, over the course of a trial for each novelty type. As mentioned earlier, each trial consists of 200 episodes, and novelty is introduced at a randomly-selected point between episodes 30-50. Once novelty is introduced, all post-novelty episodes contain the novelty. The plots are aligned so that the episode where novelty



(a) Novelty Type 5

(b) Novelty Type 6

(c) Novelty Type 7

(d) Novelty Type 8

Fig. 5: Performance across novelty types 5-8. Performance is measured as the activity recognition accuracy over the instances in an episode, or single day, in the life of the smart home.

is introduced is at the same point along the x-axis. The plots represent an average over 90 trials for each novelty type, which consists of 30 trials for each of the three difficulty levels (easy, medium, hard). Table 4 shows the values of the eight metrics described in Section 5, and Figure 7 charts these values. Again, we emphasize that the agents receive no information about the presence of novelty during a trial, and the novelty descriptions were not revealed to the agent development team until after these experiments were completed.

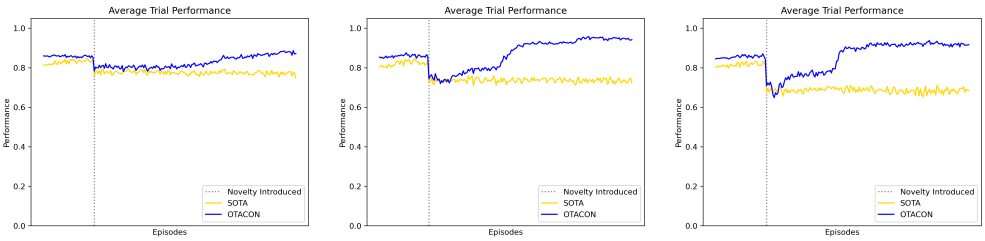
In the pre-novelty phase the performance of OTACON and SOTA are close, but generally OTACON outperforms SOTA pre-novelty, except for novelty type 2. The main reason for this difference is that the random forest classifier in OTACON is a re-implementation of the classifier in SOTA, not the exact same implementation. The OTACON classifier was initially trained on the same data as the SOTA agent, but

additional parameter tuning was performed to optimize the classifier for this data. So, the OTACON system can outperform the SOTA agent in pre-novelty in some cases.

Overall, the plots show good reaction performance to the novelties. For novelty types 1 and 4, and to some extent novelty type 8, there is little impact on performance due to these novelties. This is reflected in the high FN-CDT values for these novelties in Table 4 and Figure 7, because without a drop in performance, there is not much impetus for an agent to detect or adapt to novelty. This type of novelty has been referred to previously as *nuisance* novelty [54]. Novelty types 1 and 4 deal with the addition or removal of sensors from the sensor vector provided to the agent. The lack of performance drop indicates that the agent was resilient to these types of novelty. Novelty 8 is not fundamentally changing the behavior in the environment, but just compressing the behavior into a shorter time. While we see some initial drop in performance, the agent quickly recovers, and even exceeds pre-novelty performance, as indicated by an IPTI > 1.0 for novelty 8. When the novelty provides additional recognition opportunities, as in novelty type 8, especially for activities that the agent is better at recognizing, the agent's post-novelty performance can exceed its pre-novelty performance.

Results for the other novelty types (2, 3, 5, 6, 7) show a consistent ability for the agent to detect and react to novelty. FN-CDT and FP values are relatively low meaning that the agent detects novelty quickly and rarely detects novelty early. The detection performance for novelty type 6 is not as good as the other four novelty types, showing 11% false positive detections (FP=0.111), correct detection on only 57% of the trials (CDT=0.567), and about 36 post-novelty episodes needed before the detection is made (FN-CDT=36.5). Novelty type 6 involves an increase in temperature, which results in motion sensors being triggered more often without a change in inhabitant behavior. A systematic increase across all motion sensors is harder to detect than a change in inhabitant behavior that would most likely impact only a subset of sensors. Overall, the agent has good reaction performance on these five novelties. This is best reflected in the OPTI values ranging from 0.525 to 0.602. An OPTI value above 0.5 indicates that the increased in performance post-novelty is due to the improved performance of the novelty-aware agent. And that this performance is increasing as the agent has more experience with the novelty is indicated by the fact that IPTI $<$ OPTI and APTI $>$ OPTI. So, while these novelties resulted in a significant initial decrease in performance, the agent was able to recover its performance, and in some cases (3, 6, 7) exceed its pre-novelty performance. The latter phenomenon due to the increase in recognition opportunities provided by increased inhabitant behavior (type 3), increased sensor events (type 6), and increased frequency of leave-home events (type 7).

The results thus far have been averaged over all three difficulty levels. Results for different difficulty levels are typically as expected, i.e., the initial performance decrease is larger as difficulty increases. For example, Figure 6 shows the performance of the agent averaged over the 30 trials each for easy, medium and hard difficulty levels for novelty type 7. The results show that the initial post-novelty decline in performance increases with difficulty. Unexpectedly, the number of episodes before performance recovery decreases with difficulty. This is likely due to the increase in the fictitious



(a) Difficulty: Easy (b) Difficulty: Medium (c) Difficulty: Hard

Fig. 6: Performance across different difficulty levels for novelty type 7. Performance is measured as the activity recognition accuracy over the instances in an episode, or single day, in the life of the smart home.

leave home and enter home activities at higher difficulty levels, giving the agent more instances with which to retrain and adapt their activity recognition approach.

	Novelty Type							
Metric	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
FN-CDT ↓	103.0	13.5	6.5	95.2	0.9	36.5	18.7	48.2
CDT	0.167	0.944	0.900	0.167	0.911	0.567	0.867	0.222
FP ↓	0.000	0.033	0.022	0.011	0.089	0.111	0.000	0.000
ONRP	1.037	0.938	1.177	1.045	0.915	1.158	1.108	1.058
INRP	1.032	0.823	0.787	1.041	0.809	0.982	0.911	1.026
OPTI	0.506	0.534	0.590	0.507	0.602	0.525	0.539	0.515
IPTI	0.505	0.512	0.520	0.507	0.576	0.499	0.507	0.511
APTI	0.506	0.543	0.615	0.509	0.608	0.540	0.556	0.517

Table 4: Performance metrics across all novelty types. For FN-CDT and FP, smaller is better (↓).

6.2 Results from Additional Baselines

As described in the Related Work section, methods for novelty detection and adaptation already exist, in particular, concept and data drift detection as well as incremental learning. In this section, we compare OTACON to such methods. We should point out that the particular scenario we are addressing in this work is when the novelty is sudden, as opposed to a slow drift, and the agent typically does not have to luxury to train incrementally due to the lack of time to retrain and the cost of collecting additional examples when quick adaptation is paramount to maintain adequate performance in the domain. But we can still consider the ideal case in terms of incrementally learning and see that OTACON meets, or in some cases, exceeds performance compared to the ideal.

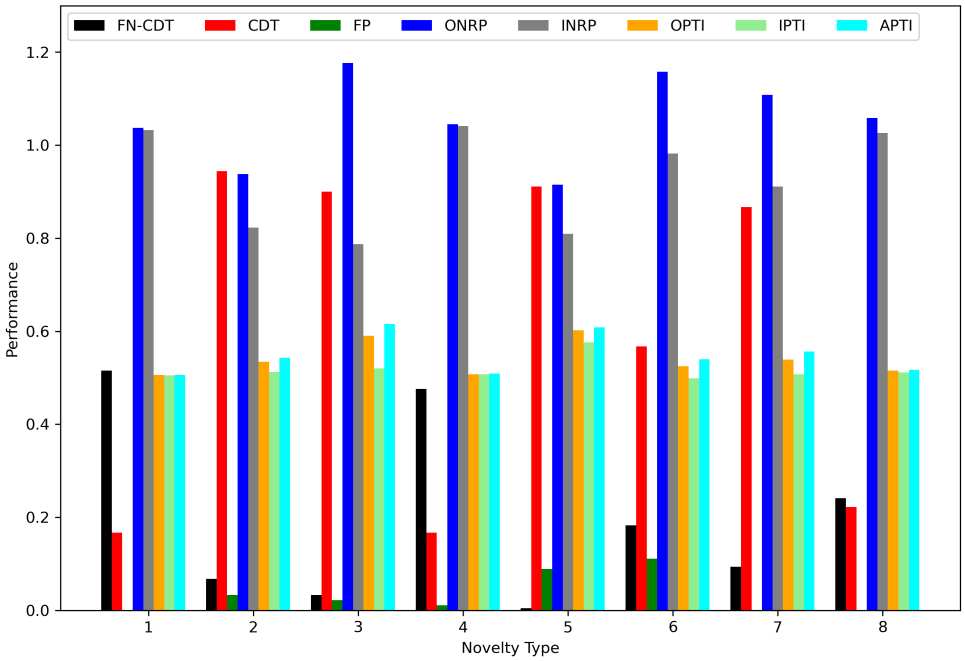


Fig. 7: Performance metrics across all novelty types. For FN-CDT and FP, smaller is better. FN-CDT values are normalized based on the number of episodes in trial to fit on the same scale as the other metrics.

6.2.1 Concept and Data Drift Detection Methods

We evaluated two drift detection techniques for novelty detection: Drift Detection Method (DDM) [51] for concept drift and Incremental Kolmogorov-Smirnov (IncKS) [52] for data drift. The DDM method has three main parameters: *warning level*, *drift level*, and *minimum number of instances*. The *warning level* and *drift level* are expressed in standard deviations of the error, with default values are 2.0 and 3.0, respectively, meaning that DDM detects drift if the error exceeds the *drift level* for examples collected since exceeding the *warning* level. This detection process begins only after a minimum number of instances have been collected; the default value for this parameter is 30. For the smart home environment, the error has high variance; Figure 8 shows a single trial for novelty level 5. Therefore, using the default parameters, the DDM method detects drift almost immediately, resulting in no correctly detected trials (CDT=0), false positive detections on every trial (FP=0), and FN-CDT being undefined. We performed some tuning on the parameters and found that *warning level = 4.0*, *drift level = 6.0*, and minimum number of instances = 10,000 yielded better results. Table 5 shows the three novelty detection metrics (FN-CDT, CDT, FP) for the DDM method with these parameter settings. Note that the false positive detections (FP) are still high, much worse than OTACON. For those few correctly

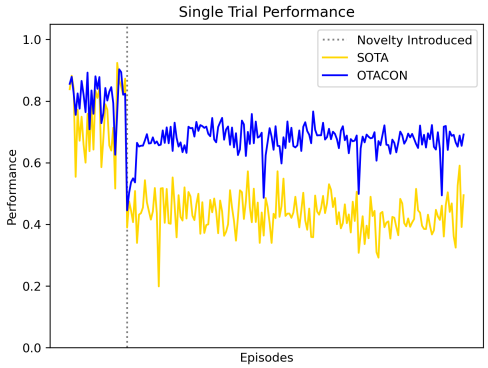


Fig. 8: Performance for a single trial from novelty level 5. Performance is measured as the activity recognition accuracy over the instances in an episode, or single day, in the life of the smart home.

detected trials in which DDM detected novelty after it was introduced, the number of false negatives (FN-CDT) is comparable to OTACON. Given the high variance in the error, concept drift detection methods are likely to have high false positive rates without significant tuning.

DDM	**Novelty Type**							
Metric	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
FN-CDT ↓	47.6	-	3.5	2.1	0	18.7	1.6	17.5
CDT	0.189	0.000	0.044	0.211	0.022	0.067	0.244	0.211
FP ↓	0.800	1.000	0.956	0.767	0.978	0.922	0.756	0.767
OTACON								
FN-CDT ↓	103.0	13.5	6.5	95.2	0.9	36.5	18.7	48.2
CDT	0.167	0.944	0.900	0.167	0.911	0.567	0.867	0.222
FP ↓	0.000	0.033	0.022	0.011	0.089	0.111	0.000	0.000

Table 5: Novelty detection metrics across all novelty types for the DDM concept drift detection method. For FN-CDT and FP, smaller is better (↓). OTACON results are repeated here for reference.

The IncKS method has two main parameters: *window size* and α. The *window size* (default = 10) is the number of instances over which to compute the statistic as the trial proceeds. The α parameter (default = 0.001) is the threshold on the p-value of the KS statistic such that if the p-value $< \alpha$, then drift is detected. The IncKS method is applied to each feature independently, and so the chance that one feature would meet the drift detection threshold is high, which, like the concept drift detection method, resulted in early detection in almost every trial. So, we introduced a third parameter, *percent features* (default = 50%) such that at least this percentage of features had to trigger drift detection at some point in the past. Even with this added

constraint, the IncKS method always detected early, which indicates that change in feature values is also highly variable for this domain, even before the introduction of novelty. With additional tuning, we were able to find parameter values (*window size = 100*, $\alpha = 0.0001$, *percent features = 60%*) that resulted in reasonable detection results (see Table 6, in particular, lower FN-CDT, but at the expense of significant false alarms (i.e., low CDT and high FP). So, overall, OTACON shows superior performance to traditional concept and data drift detection methods.

IncKS	**Novelty Type**							
Metric	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
FN-CDT ↓	9.3	5.1	6.7	6.2	16.2	29.7	1.9	7.0
CDT	0.222	0.122	0.889	0.367	0.865	0.900	0.278	0.289
FP ↓	0.778	0.878	0.111	0.633	0.144	0.089	0.722	0.711
OTACON								
FN-CDT ↓	103.0	13.5	6.5	95.2	0.9	36.5	18.7	48.2
CDT	0.167	0.944	0.900	0.167	0.911	0.567	0.867	0.222
FP ↓	0.000	0.033	0.022	0.011	0.089	0.111	0.000	0.000

Table 6: Novelty detection metrics across all novelty types for the IncKS data drift detection method. For FN-CDT and FP, smaller is better (↓). OTACON results are repeated here for reference.

6.2.2 Incremental Learning for Adaptation

The typical machine learning approach for adapting to changes in the environment is to learn incrementally over time. This can be triggered only when novelty is detected, as is done in OTACON, or applied continuously. As pointed out earlier, the scenario investigated here further constrains the learner with the assumption that additional training examples and retraining time are limited, and quick adaptation is rewarded. Still, we can evaluate the scenario without these constraints to see how a continuous, incremental learner can perform in the ideal case and compare this to OTACON's adaptation performance.

For this evaluation, we ran the same trials on what we call the IDEAL method, which accumulates examples over time, keeping the 100,000 most recent examples, and retraining the random forest classifier, same one used in the SOTA method, after each episode of a trial, including the initial non-novel episodes. A similar set of plots, as show in Figures 4 and 5, are provided in the Appendix that show the performance of the IDEAL method (see Figures 9 and 10). The results show that the IDEAL method adapts quickly to the novelty, and OTACON achieves similar results. There are some novelties in which OTACON does not quite achieve ideal (levels 4, 6, 8), but OTACON actually exceeds the ideal performance in one case (level 3). In the case of level 3, where the novelty is the inclusion of data from a new inhabitant, the IDEAL method likely overfits to one inhabitant or the other, while OTACON achieves a more general model.

So, even an ideal approach that assumes plenty of examples post-novelty and plenty of time to retrain does not greatly outperform the OTACON approach. Furthermore, the experimental scenario used here provides additional training examples only once OTACON detects novelty; whereas, the ideal scenario provided additional training from the start of the trial, before novelty, which is more of a quixotic scenario compared to the real-world setting in which these novelty-aware AI agents are expected to perform.

7 Discussion and Conclusions

Smart homes provide a unique and challenging environment in terms of requiring the recognition of complex activities based on sensor data in the presence of open-world novelty. The collection of large amounts of real-world data has facilitated the creation of our smart home novelty generator, which can generate multiple types of novelties and evaluate AI systems in terms of their ability to detect and react to novelty using several new metrics. We use the novelty generator to evaluate the OTACON approach, which adds novelty-adaptive capabilities to existing activity recognition methods.

The results presented in Section 6 demonstrate that OTACON is able to detect and adapt to novelty, and in some instances is able to surpass pre-novelty performance. Compared to traditional approaches for drift detection and incremental learning, again, OTACON was found to be superior both in terms of robust novelty detection and near-ideal adaptation. We can attribute much of OTACON's performance to two advances during its development. First is the monitoring on task agent outputs and performance. Monitoring task agent labels through PCC allows OTACON to quantify whether the agent is labeling as expected, and agent performance gives OTACON a way to validate its conclusions. Second, using a deque-like structure for the task agent data for the purpose of adaptation allowed OTACON to maintain better-than-chance performance at the moment of novelty while adapting quickly. Initially, we found that complete retraining using only novel data leads to poor performance and slower adaptation.

The experimental results show that some novelties are more difficult to detect, and some are more difficult to adapt to. OTACON observes the world through the lens of the task agent. In this case, the task agent is a random forest model trained on engineered features; it is not a causal model of how the activity of one or more occupants produces sensor activity, thus OTACON did not adapt as well to novelties that add new dimensions to the task. For example, adding multiple agents, as in novelty type 5, requires that the task agent identifies the target occupant as well as their activity. Additionally, novelties that can cause inconsistencies in feature processing present a challenge. For example, if the features are built using a temporal sliding window, a novelty that causes inconsistencies and errors in the timestamp could result in near random data, which would require adaptation in the data pipeline, which is outside OTACON's scope.

In future work, we would like to endow OTACON with the ability to propose latent variables to help adapt in scenarios that increase the scope of the task agent. For example, in novelty type five, once OTACON detects stagnation in adaptation below

the baseline level, it could propose a latent feature along side the original features. The latent feature would be conditioned on all other features and on how well it predicts the correct label. The task agent could be expanded to include the new latent feature, or PCC could attempt to learn a function to correct errors in the task agent.

The smart home novelty generator [49] could be improved along three fronts. First, the inclusion of human activity recognition datasets beyond those collected by the CASAS project (e.g., mobility-focused datasets [55], video [56]) would enhance the generality of the evaluation. Second, improvements to the ability to augment real data with similar synthetic data (e.g., using generative adversarial networks [57]) would allow for larger and longer-term experimentation. And third, additional novelty types (e.g., new activities, introduction of interventions to assist inhabitant [58]) would also enhance the generality of the novelty generator and provide a more challenging task to novelty-aware agents.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. 1954372 and by the National Institutes of Health under Grant No. R01EB009675. Research was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) and was accomplished under Cooperative Agreement Number W911NF2020004 to Washington State University and W911NF2020001 to Rutgers University and Redpoll. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the DARPA or ARO, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Declarations

Financial interests: Baxter Eaves is co-founder and CEO of Redpoll AI. Patrick Shafto is co-founder and CSO of Redpoll AI.

References

[1] Cummings, M.L.: Rethinking the maturity of artificial intelligence in safety-critical settings. AI Magazine **42**(1), 6–15 (2021) https://doi.org/10.1002/j.2371-9621.2021.tb00005.x

[2] Choi, C.Q.: 7 revealing ways AIs fail: Neural networks can be disastrously brittle, forgetful, and surprisingly bad at math. IEEE Spectrum **58**(10), 42–47 (2021) https://doi.org/10.1109/MSPEC.2021.9563958

[3] Liu, B., Robertson, E., Grigsby, S., Mazumder, S.: Self-initiated open world learning for autonomous AI agents. In: AAAI Spring Symposium (2022). https://usc-isi-i2.github.io/AAAI2022SS/papers/SSS-22_paper_36.pdf

[4] DARPA: Teaching AI Systems to Adapt to Dynamic Environments. https://www.darpa.mil/news-events/2019-02-14. [Online; accessed 29-May-2023] (2019)

[5] Chadwick, T., Chao, J., Izumigawa, C., Galdorisi, G., Ortiz-Pena, H., Loup, E., Soultanian, N., Manzanares, M., Mai, A., Yen, R., Lange, D.S.: Characterizing Novelty in the Military Domain (2023). https://arxiv.org/abs/2302.12314

[6] Krishnan, N.C., Cook, D.J.: Activity recognition on streaming sensor data. Pervasive Mob. Comput. **10**, 138–154 (2014) https://doi.org/10.1016/j.pmcj.2012.07.003

[7] Langley, P.: Open-world learning for radically autonomous agents. Proceedings of the AAAI Conference on Artificial Intelligence **34**, 13539–13543 (2020) https://doi.org/10.1609/aaai.v34i09.7078

[8] Goel, S., Shukla, Y., Sarathy, V., Scheutz, M., Sinapov, J.: Rapid-learn: A framework for learning to recover for handling novelties in open-world environments. In: 2022 IEEE International Conference on Development and Learning (ICDL), pp. 15–22 (2022). https://doi.org/10.1109/ICDL53763.2022.9962230

[9] Klenk, M., Piotrowski, W., Stern, R., Mohan, S., Kleer, J.: Model-based novelty adaptation for open-world AI. In: Proceedings of the 31st International Workshop on Principles of Diagnosis (2020). https://prl-theworkshop.github.io/prl2022-icaps/papers/PRL2022_paper_3.pdf

[10] Musliner, D., Pelican, M., McLure, M., Johnston, S., Freedman, R., Knutson, C.: OpenMIND: Planning and adapting in domains with novelty. In: Proceedings of the Ninth Annual Conference on Advances in Cognitive Systems (2021). https://www.sift.net/sites/default/files/publications/main_5.pdf

[11] Zhang, H., Li, A., Guo, J., Guo, Y.: Hybrid models for open set recognition. In: Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III, pp. 102–117. Springer, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-58580-8_7

[12] Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. ACM Comput. Surv. **54**(2) (2021) https://doi.org/10.1145/3439950

[13] Jiang, M., Hou, C., Zheng, A., Han, S., Huang, H., Wen, Q., Hu, X., Zhao, Y.: ADGym: design choices for deep anomaly detection. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS '23. Curran Associates Inc., Red Hook, NY, USA (2023). https://openreview.net/pdf?id=9CKx9SsSSc

[14] Gonçalves, P.M., de Carvalho Santos, S.G.T., Barros, R.S.M., Vieira, D.C.L.: A comparative study on concept drift detectors. Expert Systems with Applications **41**(18), 8144–8156 (2014) https://doi.org/10.1016/j.eswa.2014.07.019

[15] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering **31**(12),

2346–2363 (2019) https://doi.org/10.1109/TKDE.2018.2876857

[16] Bayram, F., Ahmed, B.S., Kassler, A.: From concept drift to model degradation: An overview on performance-aware drift detectors. Knowledge-Based Systems **245**, 108632 (2022) https://doi.org/10.1016/j.knosys.2022.108632

[17] Hinder, F., Vaquet, V., Hammer, B.: One or two things we know about concept drift—a survey on monitoring in evolving environments. part a: detecting concept drift. Frontiers in Artificial Intelligence **7** (2024) https://doi.org/10.3389/frai.2024.1330257

[18] Aminikhanghahi, S., Cook, D.J.: Enhancing activity recognition using cpd-based activity segmentation. Pervasive and Mobile Computing **53**, 75–89 (2019) https://doi.org/10.1016/j.pmcj.2019.01.004

[19] Xu, R., Song, Z., Wu, J., Wang, C., Zhou, S.: Change-point detection with deep learning: A review. Frontiers of Engineering Management **12**(1), 154–176 (2025) https://doi.org/10.1007/s42524-025-4109-z

[20] Boult, T.E., Windesheim, N.M., Zhou, S., Pereyda, C., Holder, L.B.: Weibull-open-world (wow) multi-type novelty detection in cartpole3d. Algorithms **15**(10) (2022) https://doi.org/10.3390/a15100381

[21] Wydmuch, M., Kempka, M., Jaśkowski, W.: Vizdoom competitions: Playing doom from pixels. IEEE Transactions on Games **11**(3), 248–259 (2019) https://doi.org/10.1109/TG.2018.2877047

[22] Gamage, C., Pinto, V., Xue, C., Stephenson, M., Zhang, P., Renz, J.: Novelty generation framework for ai agents in angry birds style physics games. In: 2021 IEEE Conference on Games (CoG), pp. 1–8 (2021). https://doi.org/10.1109/CoG52621.2021.9619160

[23] Goss, S.A., Steininger, R.J., Narayanan, D., Olivença, D.V., Sun, Y., Qiu, P., Amato, J., Voit, E.O., Voit, W.E., Kildebeck, E.J.: Polycraft World AI Lab (PAL): An Extensible Platform for Evaluating Artificial Intelligence Agents (2023). https://arxiv.org/abs/2301.11891

[24] Kejriwal, M., Thomas, S.: A multi-agent simulator for generating novelty in monopoly. Simulation Modelling Practice and Theory **112**, 102364 (2021) https://doi.org/10.1016/j.simpat.2021.102364

[25] Rambhatla, S.S., Chellappa, R., Shrivastava, A.: The Pursuit of Knowledge: Discovering and Localizing Novel Categories using Dual Memory . In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9133–9143. IEEE Computer Society, Los Alamitos, CA, USA (2021). https://doi.org/10.1109/ICCV48922.2021.00902

[26] Moon, W., Park, J., Seong, H.S., Cho, C.-H., Heo, J.-P.: Difficulty-aware simulator for open set recognition. In: Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXV, pp. 365–381. Springer, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-19806-9_21 . https://doi.org/10.1007/978-3-031-19806-9_21

[27] Alaghbari, K.A., Md. Saad, M.H., Hussain, A., Alam, M.R.: Activities recognition, anomaly detection and next activity prediction based on neural networks in smart homes. IEEE Access **10**, 28219–28232 (2022) https://doi.org/10.1109/ACCESS.2022.3157726

[28] Dahmen, J., Cook, D.J.: Indirectly supervised anomaly detection of clinically meaningful health events from smart home data. ACM Trans. Intell. Syst. Technol. **12**(2) (2021) https://doi.org/10.1145/3439870

[29] Chatterjee, A., Ahmed, B.S.: Iot anomaly detection methods and applications: A survey. Internet of Things **19**, 100568 (2022) https://doi.org/10.1016/j.iot.2022.100568

[30] Bianchi, V., Bassoli, M., Lombardo, G., Fornacciari, P., Mordonini, M., De Munari, I.: IoT wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment. IEEE Internet of Things Journal **6**(5), 8553–8562 (2019) https://doi.org/10.1109/JIOT.2019.2920283

[31] Sprint, G., Cook, D.J., Fritz, R.S., Schmitter-Edgecombe, M.: Using smart homes to detect and analyze health events. Computer **49**(11), 29–37 (2016) https://doi.org/10.1109/MC.2016.338

[32] Robben, S., Englebienne, G., Kröse, B.: Delta features from ambient sensor data are good predictors of change in functional health. IEEE Journal of Biomedical and Health Informatics **21**(4), 986–993 (2017) https://doi.org/10.1109/JBHI.2016.2593980

[33] Forbes, G., Massie, S., Craw, S., Fraser, L., Hamilton, G.: Representing temporal dependencies in smart home activity recognition for health monitoring. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2020). https://doi.org/10.1109/IJCNN48605.2020.9207480

[34] Fritz, R., Wuestney, K., Dermody, G., Cook, D.J.: Nurse-in-the-loop smart home detection of health events associated with diagnosed chronic conditions: A case-event series. International Journal of Nursing Studies Advances **4**, 100081 (2022) https://doi.org/10.1016/j.ijnsa.2022.100081

[35] Pereyda, C., Raghunath, N., Minor, B., Wilson, G., Schmitter-Edgecombe, M., Cook, D.J.: Cyber-physical support of daily activities: A robot/smart home partnership. ACM Trans. Cyber-Phys. Syst. **4**(2) (2019) https://doi.org/10.1145/

3365225

[36] Dahmen, J., Cook, D., Wang, X., Honglei, W.: Smart secure homes: A survey of smart home technologies that sense, assess, and respond to security threats. Journal of Reliable Intelligent Environments **3** (2017) https://doi.org/10.1007/s40860-017-0035-0

[37] Dahmen, J., Thomas, B., Cook, D., Wang, X.: Activity learning as a foundation for security monitoring in smart homes. Sensors **17**, 737 (2017) https://doi.org/10.3390/s17040737

[38] Zhang, Y., Srivastava, A., Cook, D.: Machine learning algorithm for activity-aware demand response considering energy savings and comfort requirements. IET Smart Grid **3**, 730–737 (2020) https://doi.org/10.1049/iet-stg.2019.0249

[39] Lin, B., Huangfu, Y., Lima, N., Jobson, B., Kirk, M., O'Keeffe, P., Pressley, S.N., Walden, V., Lamb, B., Cook, D.J.: Analyzing the relationship between human behavior and indoor air quality. Journal of Sensor and Actuator Networks **6**(3) (2017) https://doi.org/10.3390/jsan6030013

[40] Alzoubi, A.: Machine learning for intelligent energy consumption in smart homes. International Journal of Computations, Information and Manufacturing (IJCIM) **2** (2022) https://doi.org/10.54489/ijcim.v2i1.75

[41] Thomas, B.L., Cook, D.J.: Activity-aware energy-efficient automation of smart buildings. Energies **9**(8) (2016) https://doi.org/10.3390/en9080624

[42] Cook, D.J., Crandall, A.S., Thomas, B.L., Krishnan, N.C.: Casas: A smart home in a box. Computer **46**(7), 62–69 (2013) https://doi.org/10.1109/MC.2012.328

[43] Bouchabou, D., Nguyen, S.M., Lohr, C., LeDuc, B., Kanellos, I.: A survey of human activity recognition in smart homes based on iot sensors algorithms: Taxonomies, challenges, and opportunities with deep learning. Sensors **21**(18) (2021) https://doi.org/10.3390/s21186037

[44] Soleimani, E., Nazerfard, E.: Cross-subject transfer learning in human activity recognition systems using generative adversarial networks. Neurocomputing **426**, 26–34 (2021) https://doi.org/10.1016/j.neucom.2020.10.056

[45] Dahmen, J., Cook, D.: Synsys: A synthetic data generation system for healthcare applications. Sensors **19**(5) (2019) https://doi.org/10.3390/s19051181

[46] Shafto, P., Kemp, C., Mansinghka, V., Tenenbaum, J.B.: A probabilistic model of cross-categorization. Cognition **120**(1), 1–25 (2011) https://doi.org/10.1016/j.cognition.2011.02.010

[47] Saatçi, Y., Turner, R., Rasmussen, C.E.: Gaussian process change point models.

In: Proceedings of the 27th International Conference on International Conference on Machine Learning. ICML'10, pp. 927–934. Omnipress, Madison, WI, USA (2010). https://dl.acm.org/doi/10.5555/3104322.3104440

[48] Adams, R.P., MacKay, D.J.C.: Bayesian Online Changepoint Detection (2007). https://arxiv.org/abs/0710.3742

[49] Holder: Smart Environment Novelty Generator. GitHub repository (2025). https://github.com/holderlb/WSU-SmartEnv-NG

[50] Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001) https://doi.org/10.1023/A:1010933404324

[51] Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Bazzan, A.L.C., Labidi, S. (eds.) Advances in Artificial Intelligence – SBIA 2004, pp. 286–295. Springer, Berlin, Heidelberg (2004). https://link.springer.com/chapter/10.1007/978-3-540-28645-5_29

[52] Reis, D.M., Flach, P., Matwin, S., Batista, G.: Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16, pp. 1545–1554. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939836 . https://doi.org/10.1145/2939672.2939836

[53] Céspedes Sisniega, J., López García: Frouros: An open-source python library for drift detection in machine learning systems. SoftwareX **26**, 101733 (2024) https://doi.org/10.1016/j.softx.2024.101733

[54] Boult, T., Grabowicz, P., Prijatelj, D., Stern, R., Holder, L., Alspector, J., Jafarzadeh, M., Ahmad, T., Dhamija, A., Li, C., Cruz, S., Shrivastava, A., Vondrick, C., Walter, S.: Towards a unifying framework for formal theories of novelty. Proceedings of the AAAI Conference on Artificial Intelligence **35**, 15047–15052 (2021) https://doi.org/10.1609/aaai.v35i17.17766

[55] Reyes-Ortiz, J., Anguita, D., Ghio, A., Oneto, L., Parra, X.: Human Activity Recognition Using Smartphones. UCI Machine Learning Repository (2013). https://doi.org/10.24432/C54S4K

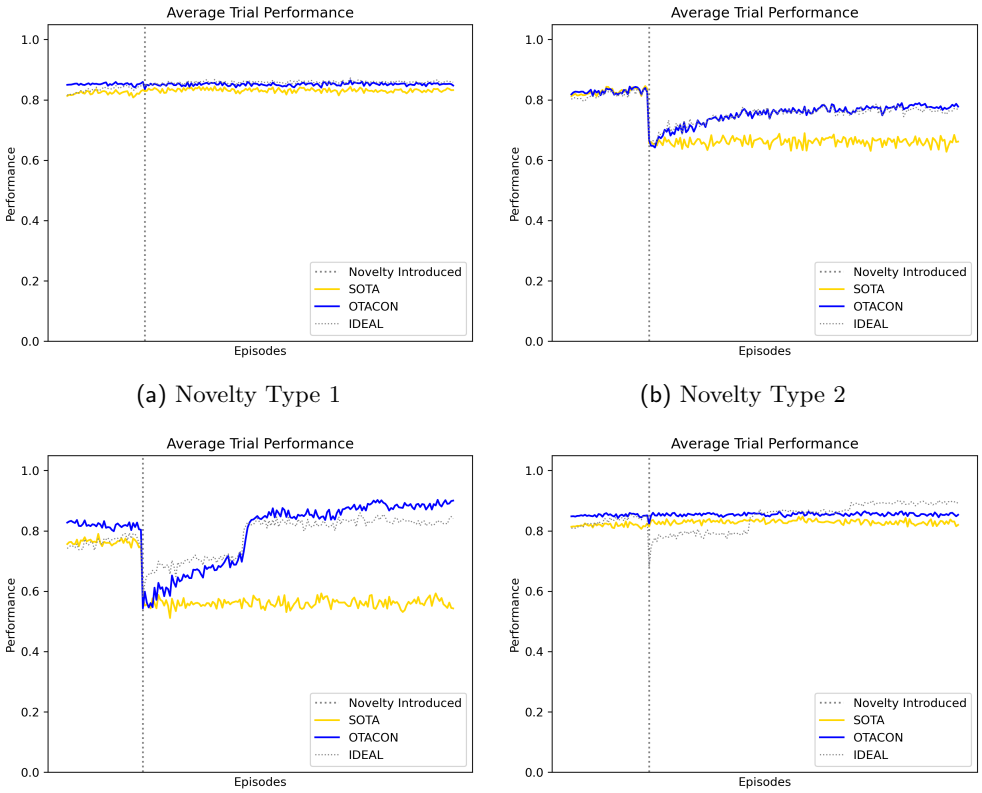
[56] Vaquette, G., Orcesi, A., Lucat, L., Achard, C.: The daily home life activity dataset: A high semantic activity dataset for online recognition. In: 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), pp. 497–504 (2017). https://doi.org/10.1109/FG.2017.67

[57] Briscoe, J., Gebremedhin, A., Holder, L., Cook, D.: Adversarial creation of a smart home testbed for novelty detection. In: AAAI Spring Symposium (2022). https://usc-isi-i2.github.io/AAAI2022SS/papers/SSS-22_paper_71.pdf

[58] Cook, D.J., Strickland, M., Schmitter-Edgecombe, M.: Detecting smartwatch-based behavior change in response to a multi-domain brain health intervention. ACM Trans. Comput. Healthcare **3**(3) (2022) https://doi.org/10.1145/3508020

A Ideal Incremental Learning Results

Figures 9 and 10 show the results of OTACON compared to the IDEAL incremental learning approach that provides additional training examples and retraining time after each episode of a trial; whereas OTACON only receives examples after detecting novelty. Still in this limited scenario, OTACON achieves ideal performance for most novelty levels and even exceeds ideal performance in one level. See the main text for more discussion of these results.



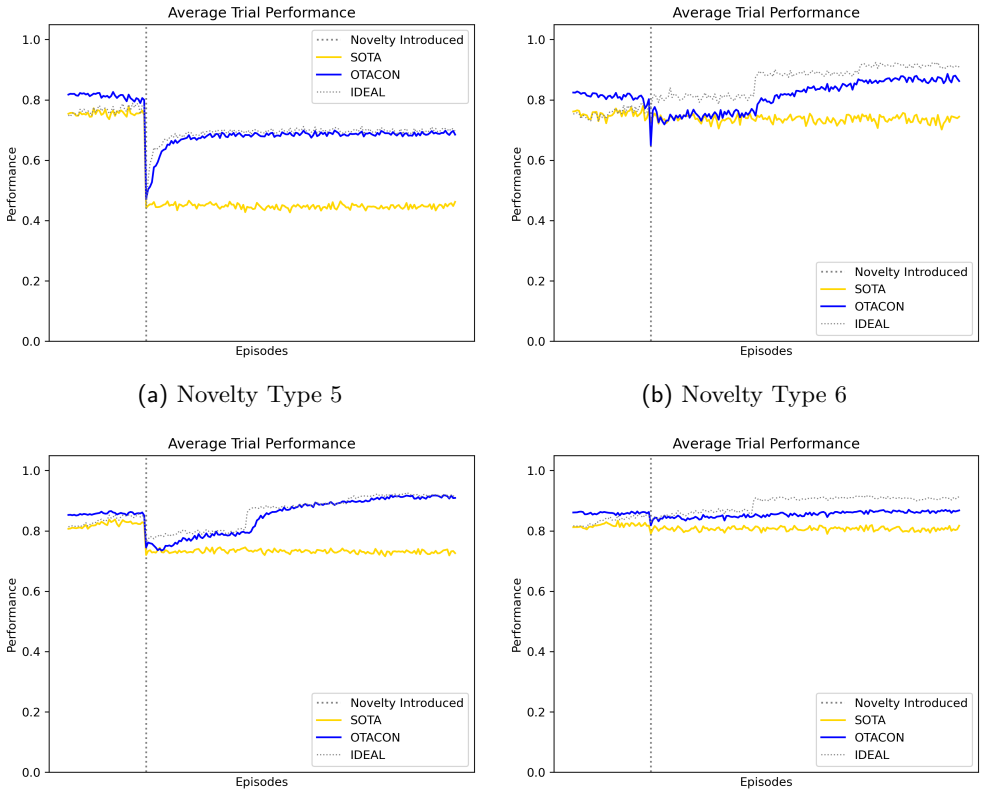
(a) Novelty Type 1

(b) Novelty Type 2

(c) Novelty Type 3

(d) Novelty Type 4

Fig. 9: Performance of OTACON and IDEAL across novelty types 1-4. Performance is measured as the activity recognition accuracy over the instances in an episode, or single day, in the life of the smart home.



(a) Novelty Type 5

(b) Novelty Type 6

(c) Novelty Type 7

(d) Novelty Type 8

Fig. 10: Performance for OTACON and IDEAL across novelty types 5-8. Performance is measured as the activity recognition accuracy over the instances in an episode, or single day, in the life of the smart home.