Facilitating Human Activity Data Annotation via Context-Aware Change Detection on Smartwatches

Ali Akbari

Department of Biomedical Engineering, Texas A&M University, College Station, Texas, USA, aliakbari@tamu.edu

Jonathan Martinez

Department of Computer Science and Engineering, Texas A&M University, College Station, Texas, USA, jmartinez0304@tamu.edu

Roozbeh Jafari

Department of Biomedical Engineering, Computer Science and Engineering, and Electrical and Computer Engineering, Texas A&M University, College Station, Texas, USA, rjafari@tamu.edu

ABSTRACT

Annotating activities of daily living (ADL) is vital for developing machine learning models for activity recognition. In addition, it is critical for self-reporting purposes such as in assisted living where the users are asked to log their ADLs. However, data annotation becomes extremely challenging in real-world data collection scenarios, where the users have to provide annotations and labels on their own. Methods such as self-reports that rely on users' memory and compliance are prone to human errors and become burdensome since they increase users' cognitive load. In this paper, we propose a light yet effective context-aware change point detection algorithm that is implemented and run on a smartwatch for facilitating data annotation for high-level ADLs. The proposed system detects the moments of transition from one to another activity and prompts the users to annotate their data. We leverage freely available Bluetooth low energy (BLE) information broadcasted by various devices to detect changes in environmental context. This contextual information is combined with motion based change point detection algorithm, which utilizes data from wearable motion sensors, to reduce the false positives and enhance the system's accuracy. Through real-world experiments, we show that the proposed system improves the quality and quantity of labels collected from users by reducing human errors while eliminating users' cognitive load and facilitating the data annotation process.

CCS CONCEPTS

• Human-centered computing → Ubiquitous and mobile computing systems and tools

KEYWORDS

Change point detection, context detection, motion sensor, BLE, activity recognition, data annotation, smartwatch app

1 Introduction

Monitoring activities of daily living (ADL) provides vital contextual information for various applications including health monitoring, assisted living, security and surveillance, and mobile services [1], [2]. Rigorous machine learning and deep learning algorithms have been designed in conjunction with wearable motion sensors for automatic detection of various ADLs [3]–[6]. However, performance of all these machine learning algorithms highly depends on the availability of a sufficient amount of precise labeled training data and are sensitive to the level of activities being targeted [7]. That is, activities may be organized into a hierarchy where long-term activities are often composed of multiple short-term activities. Abundant precise annotations for this scenario becomes more critical when working with data hungry deep learning models [7], [8].

Accurate logging and annotating of ADLs is required in health monitoring applications too [9]. The process of data collection and annotation is usually burdensome and time consuming, and it is prone to human errors especially in real-world scenarios where the users are required to annotate their own data. The aim of this study is to build an automated system for timely solicitation of high-level activity labels by designing a context-aware change point detection algorithm that runs on a smartwatch. This system assists the users in providing convenient and precise annotations and is extremely helpful for the researchers in the field of activity recognition as well as health monitoring.

In controlled experiments, researchers are able to monitor participants and annotate activities manually; however, in real-world data collection scenarios, it is very challenging to continuously monitoring of users' activities to assign labels. In this case, the users must annotate their own data. This is typically achieved through retrospective self-reporting. However, this method is limited since the details of previous activities cannot always be recalled precisely via retrospective memory [10]. Even when users are asked to annotate each activity on their handheld devices – which we will refer to as unprompted self-reporting for the rest of this paper – whenever they switch from one activity to another, the quality of the provided labels are low as the users frequently forget to annotate either the beginning or end of an activity [11]. Moreover, this becomes cognitively challenging for the users to remember to annotate their activities throughout the day. This is more challenging when the set of activities that the user should annotate increases, which is the case in assisted living scenarios where a variety of high-level ADLs should be monitored [3]. Therefore, there is a need for an automated system that can request labels from the users at the appropriate time when a transition in activity occurs. Such a system could alleviate three challenges associated with the unprompted self-reported annotation approach: 1) it relaxes the users cognitive load since they no longer have to remember to provide the label at the time that they start a new activity, 2) it reduces inaccurate annotation due to human error, and 3) it further facilitates annotation by notifying the users and displaying the list of activities to choose from.

In contrast to unprompted self-reported annotation, which relies on the users' memory, automatic solicitations of labels requires some sub-processes to help determine when to prompt the user. While the unprompted self-reporting approach has been previously used in real-world scenarios [3], [11], there are not many studies that attempt automatic solicitation. Certain studies have built active learning systems to solicit labels from new users in order to personalize recognition models [12]. These systems, however, still initially require a set of labeled training data to train classifiers, and through the active learning scheme they only receive new data for personalization of the models. Therefore, there is still a need for collecting labeled data for these systems to begin with. Reminding the users for annotation at random or constant intervals is another solution that reduces the users' cognitive load but cannot achieve high quality labels since there is no guarantee that it will alert a user at the proper start and end time of activities.

Soliciting the labels at the exact time of activity transitions is the best approach to both reducing the users' burden and maximizing the quality and usability of the labels for training machine learning models [13], [14]. This can be achieved through change detection algorithms by identifying the transitions in the motion signals that are associated with transitions from one to another activity. Specifically, unsupervised change point detection algorithms can identify activity transitions independently of the exact activity that is performed, and they do not require any prior knowledge or training data. Most of the current change point detection algorithms focus on detecting the exact sample at which the change occurs by estimating the distribution of the data [15]. These algorithms: 1) typically require heavy computations that makes them inappropriate for implementation on small wearable devices such as a smartwatch. 2) Many of the existing algorithms cannot be easily applied to multi-dimensional data. 3) Using merely motion data for change detection for identifying the activity transition can cause lots of false positives [16]. For instance, some people move their hands frequently while they are talking. Using only the motion sensor worn on the wrist, multiple change points will be detected in this scenario. However, adding contextual information such as change in the users' environment could help to identify activity transitions more accurately. 4) Finally, in the application of detecting ADL transitions for the goal of label solicitation, in contrast to typical change point detection algorithms, impulse and short-term changes (e.g., when a running person suddenly stops to avoid an obstacle

for a moment and then continues) are not of interest; instead, we are interested in persistent changes from one to another activity.

In this paper, we propose a context-aware, nonparametric, unsupervised change point detection algorithm for motion sensors' data that can be implemented and run on commercial smartwatches with small computational power. The proposed system detects the activity transition moments and notifies the users to annotate their data. It should be noted that in this paper, we do not focus on detecting the activities; instead we detect the changes in an unsupervised manner. This ensures that our system is sufficiently generic for a range of applications without requiring prior knowledge. Through the experiments in real-world data collection, we show that this system obtains high quality and precise activity labels from the users while it facilitates the annotation and significantly reduces their cognitive load compared to the unprompted self-reporting approach. We perform analysis by segmenting (or windowing) the motion data and extracting informative features from each segment. To detect if a change occurs within a segment, we treat the segments before and after that segment as instances of different classes. Based on the ratio between the intra-class and inter-class variability of the features, the algorithm decides if there is any change in the mentioned segment. In addition, we leverage the Bluetooth low energy (BLE) data broadcasted by various devices around the user, to identify changes in the context in order to improve the motion-based change detection. There is no need for any extra device or infrastructure for this context detection. Moreover, this context is a nice-to-have component of the proposed algorithm, meaning that this algorithm can work merely with motion sensors data even when there is no contextual information available although the rate of false positives may increase. In summary, the contributions of this paper are as follows:

- Proposing a light nonparametric yet accurate change point detection algorithm for motion sensor data that can efficiently run on a smartwatch.
- Adding contextual information obtained from BLE packets to enhance the capability of the system in detecting changes. To best of our knowledge, this is the first work that uses freely available BLE data broadcasted by devices around the users for identifying changes in the environmental context.
- Using the proposed context-aware change detection technique to design a smartwatch app to facilitate the ADL data annotation in real-world data collection. The app provides precise labels while reducing users' cognitive load.
- Validating the performance of the system through real-world experiments.

2 Related Works

In this section, we will start by reviewing the studies that focused on data annotation for activity recognition, and then we will explain research work on change point detection.

2.1 Data Annotation

Data annotation is a critical step in training machine learning algorithms. In the case of activity recognition, the most common approach for collecting labeled data is to conduct controlled experiments in which scientists supervise the data collection process. This supervision could be either in person or through recording the activities and performing offline annotation [17], [18]. This approach, however, is not feasible in real-world data collection scenarios where the data is collected by users who go through their normal daily life [19]. There are many works that focus on personalizing activity recognition models for new users by facilitating a data annotation process for them. Active learning is a widely used technique for this as it attempts to identify important samples and acquire labels from the users for those samples with the aim to minimize the interactions [20]–[23]. All these methods, however, need to have access to a model that has already been trained on some labeled data so it can identify the important samples. In this case, still the challenge is to collect the initial labeled data for training those models.

There are only a few studies in the literature that have focused on facilitating the whole data annotation process. CLAP is a system that enables the collection and labelling of data via an Android based mobile app [24]. This system trains an activity classifier for both stationary and non-stationary activities. Upon detecting a transition from an activity to 'standing still' activity, the app prompts the user to enter the label. This system still needs a set of labeled data to train the initial activity classifier. Moreover, this requirement results in potentially inter-activity transitions, such as walking to running, being lost. The change point detection approach was used to deliver ecological momentary assessment [13]. The authors of this paper showed that the rate of responding to ecological momentary assessment increases significantly if the system can ask the users to respond during the times of transition between activities. They used separation distance as the measure of dissimilarity between two windows of data gathered from motion sensors in smart-homes to detect transitions. Another study proposed a change point detection method for label solicitation implemented on a smartphone [14]. This system slides a window over the data and leverages the F-statistics calculated from mean and variance of each window to detect change points and then asks for users' feedback upon detection of change points. This system only relies on the motion data, which results in lots of false positives in real-world data collections. In addition, relying only on the summary of statistics, i.e., mean and variance, decreases the accuracy of the system in the presence of complex activities.

2.2 Change Point Detection

Change point detection has been approached with both supervised and unsupervised learning methods depending on the availability of labeled training data. Multi-class classifiers are able to both identify and explain change points [25]; however, they require prior knowledge of all possible data behaviors. To overcome this, one-class classifiers have been proposed to learn to identify only one type of data behavior and then determine everything that is quantitatively unfamiliar to the model as a change [26]. But this also presents a challenge as it highly depends on both the amount of variance and the number of instances obtained in a training set. On the contrary, unsupervised learning methods enable change detection without any given prior knowledge of the data set. This provides a scalable resource that may be applied to various scenarios. Among these unsupervised methods are those referred to as Likelihood Ratio Methods [27] which have achieved state-of-the-art performances for change point detection. This approach involves the estimation of data distributions and monitoring how they evolve over time.

Likelihood Ratio Methods are especially desirable when monitoring patterns of multi-step windows of data. The CUSUM (cumulative sum) algorithm is very popular as its detection scheme is based on accumulating the sum of changes in a monitored metric and identifies a change when a pre-defined threshold is passed [28]. Although an intuitive mechanism to implement, this approach first requires the determination of the metric to be used to summarize a window of data (typically a mean) and a hard threshold to be set. Furthermore, when analyzing a multivariate time-series, each feature of data is often treated as independent and correlations are not considered. So, in a sense, this mechanism is limited by the statistical measure that is being used to summarize a window of data. On the other hand, other approaches propose the distribution estimation of time-series windows and through hypothesis testing compare their density functions over time [29]. While effective, as feature dimensionality grows, the computational complexity of the density estimation step will also grow. So, researchers propose to handle this concept by first applying a dimensionality reduction step, with techniques such as principal component analysis (PCA) [30]. This also has shown to better distinguish clusters of data from each other in the transformed space. However, such representations are still limited by the linear nature of the projection technique.

Another valuable extension to this concept involves directly estimating the density ratio between two time-series windows. This helps to overcome the computational cost of standard density estimation methods and the challenge in estimating the likelihood of each sample independently in an online fashion. Kullback-Liebler Importance Estimation (KLIEP) was proposed for this and later extended to handle the streaming time-series setting [31]. Although effective, this approach considers two consecutive stream samples as independent and identically distributed which may not best capture the sequential nature of the stream. This approach was again extended by the Least-Squares Importance Fitting (RuLSIF) method which aims to

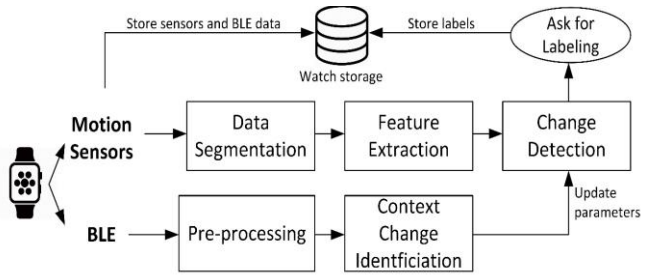


Figure 1: Overview of the proposed platform for change point detection based label solicitation on smartwatch

reduce computation cost of ratio estimation [32]. However, both methods still suffer from a major challenge share by most unsupervised change detection approaches. The first being, careful consideration and setting of model hyperparameters for static thresholds. And second, both approaches have shown to decline in performance when in presence of a noisy environment – partially due to the former challenge. Although some work has strived to overcome this by the development of somewhat dynamic threshold settings based on percent of change [33], such thresholding still does not appear capable of fully adapting to the unique characteristics of the data being analyzed.

3 Methods

Figure 1 shows the overview of the app designed for detecting activity transitions and soliciting the activity labels from the user. The data collected by the accelerometer and gyroscope sensors, which are embedded in the smartwatch, are analyzed by the change point detection module to identify persistent changes in the activities. In order to reduce the false positive rate in detection the true transition between the activities we process all the BLE packets observed around the user to detect changes in the environmental context. In this study, the environmental context could be the physical location of the users or people and/or devices around them. The parameter of the motion-based change detection are set and updated based on the changes in the environmental context. Please note that, the motion-based change detection algorithm can work independent of the context although the contextual information can enhance the accuracy of the system. In Section 4.2, we show the impact of the context identification for reducing the amount of false positives in detecting activity transitions. The label is solicited from the user upon detection of a change in the activity. Please note that this system does not leverage any training or a priori labeled data. The proposed algorithm is non-parametric and unsupervised in this study, which significantly improves the usability of the system. The app stores all the sensors data along with the observed BLE packets continuously on the smartwatch, and it stores the activity labels whenever they are provided by the user in response to the app's prompt. In the following we will explain the details of each component as shown in Figure 1.

3.1 Motion-based Change Point Detection Algorithm

In this section we explain the details of our unsupervised change point detection algorithm for motion sensor data. There are a few considerations that drive the needs for our proposed algorithm:

- This algorithm needs to run on a smartwatch with limited computational capabilities. Therefore, it is important for the algorithm to be light and fast.
- In contrast to typical change point detection algorithms, here short-term activities and impulse changes like stretching the hand in the middle of working with computer is not of interest. Instead, persistent switching between activities must be detected because these data and their labels will be eventually used to train an activity recognition system. Hence, very rapid and short changes in the activities and/or impulse motions, which usually last less than a few seconds, are not useful, and they should not be taken into account.

- Working with motion sensors with high sampling rates, single samples do not provide enough information for detecting changes. Thus, the algorithm should be able to look over patterns of longer segments of the data to detect changes.
- The motion data is dynamic meaning that there is always a certain level of variations in the signals during particular activities where those changes do not correspond to activity transitions. Hence, the system needs to detect changes in the patterns of the signals over long periods of time instead of looking at local amplitude changes.

To address the aforementioned issues, we proposed a simple yet effective window-based change point detection algorithm that detects whether there is a change occurred in the segments of motion data. In fact, the proposed algorithm does not look for the exact sample at which the change happens. Given the application of label solicitation, the exact sample of transition within the segment is not critical because 1) the length of the windows is small enough such that users won't forget the correct label, and 2) to use this data for training activity recognition models later, we can always remove few samples of data around the start and end time of each activity to avoid mislabels. However, working with windows of data instead of individual samples will 1) significantly reduce the computational complexity of the algorithm, and 2) increase the accuracy of the system by looking at patterns of signals over larger segments and detecting persistent changes in activities rather than short-term activities and impulse changes [6].

To detect changes, we segment the sensor data into windows of t seconds with no overlap in real time to reduce the computational complexity. In this method, instead of working with the raw data of the sensors, we extract several informative and critical features from each segment of data. Those features are powerful in terms of explaining the general morphology of the signal and discriminating different activities. The features include a set of statistical features that are proven to be useful for activity recognition [34]. The features that are extracted from each segment of data for change detection are: mean, standard deviation, mean crossing rate, zero crossing rate, area under the curve, skewness, kurtosis, root mean square, maximum, and entropy. It should be noted that these features are simple enough to be calculated in real-time on the smartwatch, yet they are effective regarding capturing the important patterns of the signals. For the same reason we avoid extracting frequency domain and spectral features. Each feature is extracted from each axis of each sensor (X, Y, Z) separately as well as from the magnitude of the signals which makes p = 40 features in total.

After extracting the features for each segment of data in real-time, we look back at m previous windows, where m must be an odd number, and aim to detect whether the middle window contains a change point as shown in Figure 2. To be formal, we call this window at which we are looking to identify a change (i.e., the green window in Figure 2) as the target window. It should be noted that this will generate a delay of $\frac{m-1}{2} \times t$ seconds in detecting the changes. In this study, we empirically set m = 7 and t =15 seconds through experiments with our collected data. With those values, delay of the system will be 45 seconds, which is acceptable in our application since people will not forget the label during such short period of time.

The main idea to detect changes is that if the target window contains a change point, then the windows after that target window and the ones before it belong to two different activities. Please note that this is a valid assumption given that we are looking for persistent changes in the user's activity, but not short-term and impulse changes. Therefore, the features extracted from the windows to the right hand side of the middle windows should be similar to each other; the same is true for all the windows to the left. However, the features from the windows to the right are different from the features to the left. Hence, by looking at the intra-group and inter-group variability of the features we should be able to detect windows that contain a change. Each window W_i^k, where $i \in \{1,2, \dots, \frac{m-1}{2}\}$ and $k \in \{a, b\}$, is a vector of p features and p is the total number of features. The superscript b denotes the windows before the target window and superscript a denotes the

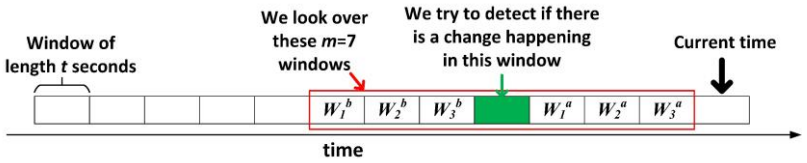


Figure 2: Illustration of the segments of data for the proposed change point detection algorithm with m = 7

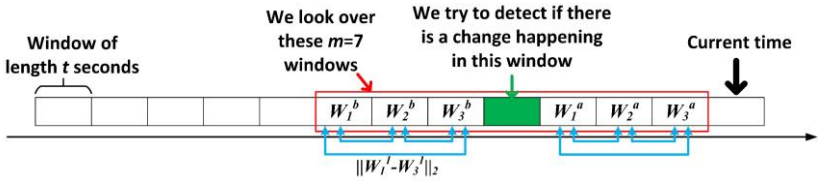


Figure 3: Calculating the intra-group variability

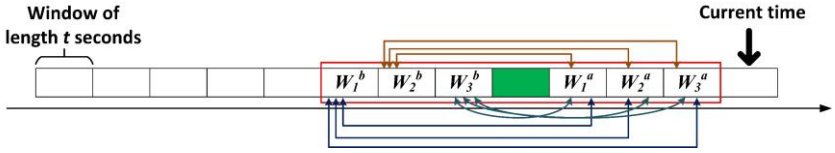


Figure 4: Calculating the inter-group variability

windows after that. The subscripts show the window index in each side. The total Euclidian distance between the features of all the windows either before or after the target window is called intra-group, aka within-group variability S_W, as shown in Figure 3 by blue lines and is calculated by Equation 1.

$$S_W = \frac{1}{2\times \mathrm{P}_2^{(m-1)/2}} \sum_{k\in\{a,b\}} \sum_{i=1}^{u} \sum_{j=1}^{u} ||W_i^k - W_j^k||_2 \quad (1)$$

where $u = \frac{m-1}{2}$ and is equal to 3 in this study given Figure 2, $\mathrm{P}_2^{(m-1)/2}$ is the permutation, which is equal to the number of ways to choose a sample of 2 elements from a set of $\frac{m-1}{2}$ distinct objects where order does matter. In fact, Equation 1 computes the average distance between every two windows in the feature space which shows the similarity between the them. and $||X, Y||_2$ is norm-2 which is calculated as:

$$||X, Y||_2 = \sqrt{(X_1 - Y_1)^2 + (X_2 - Y_2)^2 + \cdots (X_p - Y_p)^2} \quad (2)$$

where the subscript *{1,…,p}* point to each feature in windows *X* and *Y*.

The total distance between the features of every window before the target window and every window after the target window is called inter-group variability, aka between-group variability S_B, as shown in Figure 4 and is calculated by Equation 3.

$$S_B = \frac{1}{\left(\frac{m-1}{2}\right)^2} \sum_{i=1}^{u} \sum_{j=1}^{u} ||W_i^b - W_j^a||_2 \quad (3)$$

This value is the average distance between every window before and after the target window and indicates the similarity between the activities occurring before and after that window.

The ratio between the inter-group and intra-group variability is called F. Using Equation 4 the F value is continuously calculated on the smartwatch for each window of data in real-time as they become available.

$$F = \frac{S_B}{S_W} \quad (4)$$

Theoretically, the F value meets its maximum whenever the target window contains a change. The reason is that when there is a change in the target window, all the windows before/after that window are from the same activity and they will have minimum intra-group variation. Whereas the inter-group variability is the maximum; therefore, the F ratio would meet its maximum. To check if there is a change in i^{th} target window, we wait for the F value to become available for $(i+1)^{th}$ and $(i+2)^{th}$ windows and then check if the F value at window i is a local maximum or not. This will increase the total delay from 45 seconds to 75 seconds which is still reasonable in our application. To account for this delay, we cut the first and last 75 seconds of the signals when we use these data to train activity recognition models. Again, it should be noted that in this application, short term activities and changes are not of interest; instead, we look for annotating activity data when they are steady and useful for future machine learning training.

Due to noisy nature of the motion sensors embedded in commercial smartwatches, merely checking for peaks of F value is not enough since sometimes small fluctuations in the signals will be confused with real changes. To address this problem, we compare the F value, in case it is detected as a peak, with a single threshold Th and if it is less than the threshold it will be ignored. To account for the normal variations in the signals, which could happen during certain activities, we adaptively update the threshold as a constant multiplication of the average F values over last l windows given Equation 5.

$$Th^i = c^i . \frac{1}{l} \sum_{k=1}^{l} F^{i-k} \quad (5)$$

Where Th^i is the threshold used for i^{th} window, c^i is the constant multiplier at i^{th} window, l is the number of previous windows over which we calculate average of F, and F^{i-k} is the F value at $(i-k)^{th}$ window. We set l = 10 in our study. Determining c is critical because it controls the sensitivity of the change point detection algorithm. With smaller c the system will detect more changes, which increases both true and false positives. On the other hand, larger values of c leads to less detected changes, which reduces both true and false positives. One of the most important properties of our proposed method is that the threshold, more specifically the multiplier c is not constant and there is no need to set that manually. In fact, we leverage environmental information obtained from BLE packets (see next section) to adaptively update the threshold based on users' context.

As the window length in this study is 15 seconds, and one change per window could be detected, the resolution of detecting the start/end of activity is 15 seconds. Moreover, if an impulse change happens and it does not last for more than 15 seconds (at least more than one window) and the activity does not change after that point as well, most likely our system will ignore that as a change, which would be a desired property for the application of ADL annotation given that most of complex ADLs last more than 15 seconds. This value, however, could be changed based on the requirements of end-applications (see Section 3.4). However, if two consecutive activities are very similar regarding the pattern of hand motion, and the transition between them is very fast (less than 15 seconds given our window size), then there is a likelihood of missing that change in our algorithm. Enforcing the system to detecting those kinds of impulse changes will increase the rate of false positives, due to random hand movements in the middle of a long activity (e.g., scratching the head while typing). Therefore, the user burden created by those false positives would be much more significant than missing a few activity labels that could rarely happen in very specific cases.

3.2 Context Modeling via BLE Data

Using merely motion sensors on the wrist for change detection may generate many false positives since there are several changes that do not correspond to real activity transitions. For instance, imagine a person sitting at his desk and typing on the computer; if he pauses for a few seconds, stretches his hand, and then continues typing, there would be no real activity transition. However, from the perspective of the hand motion, it will be seen as a change in the signal captured by the motion sensors. These may also be considered as micro-activities, which are not of interest for the current application of data labeling where the labels will eventually be used for training an ADL recognition models. To address this issue, we need to take extra contextual information into account. In this study, we define the notion of environmental context as location, nearby

people, and nearby objects. We hypothesize that as long as the environment of the user is constant, there could be less chance of change in activities. On the other hand, when the environmental context changes, most likely the activity changes as well. To reflect this in our change point detection algorithm, we set the multiplier c in Equation 5 with respect to the changes in environmental context. As long as the context is constant, the value of c would be large meaning that we care for intense motion changes and ignore slight motions changes. However, when there is a change in the context, the value of c decreases and the system will take small motion changes into account. It should be noted that, the activity change is mainly detected by the motion data and the contextual information is just used to update the threshold. As a result, even if the environment around the user suddenly changes, the system will not detect a change unless the motion pattern of the user changes. Furthermore, even when there is no enough contextual information available around the user, e.g. there is no BLE data observed around the user, the system continuous working with the default threshold and detects the changes based on the motion data; hence, absence of context data does not make the system stop working although it impacts false positive rate.

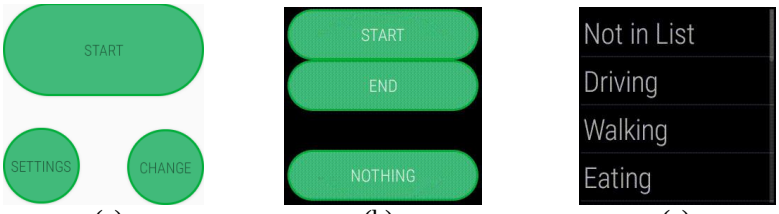
For an unobtrusive and data-driven context detection we use freely available Bluetooth low energy (BLE) data broadcasted by various devices. It is also known as "nearables" [35]. These nearables are effectively smart objects, an integral component of the IoT as such, these are equipped with Bluetooth such that they are passively detectable from the wireless signals they transmit. The BLE packets transmitted in the air can come from static devices, such as a TV in the conference room that indicates a certain location; it could also come from handheld devices, such as a colleague's cellphone that indicates presence of certain people. We call the devices who transmit these BLE packets as BLE beacons in this section.

The smartwatch used in this study is capable of sniffing the BLE packets around it. The BLE packets include the MAC address of the devices by which they are transmitted, and they are freely available. With the progress of IoT there could be found abundant of devices in each environment that are equipped with BLE technology. We leverage no prior knowledge about these MAC addresses meaning that we do not know what device is transmitting a specific BLE packet. However, since we are only interested in identifying changes in the context, we do not need to know the exact type of devices. Instead, we only look at the changes in the set of observed devices around the user over short periods of time to identify context changes. Since the BLE packets could come from devices that are sporadically present around a user, e.g., the cellphone of random people who are passing nearby the user, usually a big portion of the observed BLE beacons are not relevant to the user's context. Thus, the first step of identifying the change in the context is to filter out the set of BLE beacons that have not been seen consistently. To achieve this, we look at windows of one minute and remove all the BLE beacons that have been seen for less than 45 seconds during the one minute scanning window.

After filtering out irrelevant beacons, we look at the set of remaining BLE bacons over the last two one-minute windows, and we measure the amount of similarity between the observed devices within those two windows. To measure the similarity, we use intersection-over-union (IoU), also known as the Jaccard index as shown in Equation 6 [36].

$$IoU = \frac{\#\ of\ unique\ beacons\ present\ in\ both\ windows}{total\ \#\ of\ unique\ beacons\ observed\ in\ two\ windows} \quad (6)$$

Higher IoU means that there is not much change in the set of observed BLE beacons from one window to the next one, which indicates that the context has not been changed. On the other hand, smaller IoU means dissimilar set of BLE beacons were present in the two consecutive windows, which shows the context has been changed. We categorize the BLE change based on the value of IoU and use that to determine the constant multiplier c of the threshold in Equation 5. When $IoU > 0.6$ we consider no change in the context and set $c = 2$ which leads to detect less changes; when $0.2 < IoU < 0.6$ we consider fair chance of change in the context and set $c = 1$ which is the default value; and finally, when $IoU < 0.2$ we consider a definite change in the context and set $c = 0.01$ which leads to detect more changes.



(a) (b) (c)

Figure 5: The data collection and annotation app interface

3.3 Smartwatch App

In this study, we used a Polar M600 watch (an Android-based Wear OS watch) running the Android's smart watch operating system, Wear OS 2.0, based on Android 8.0 [37]. The motion sensors including three-axis accelerometer and gyroscope embedded in this smartwatch are sampled at 20 Hz and the BLE packets are scanned every 5 seconds.

We designed an app that: 1) collects and stores all accelerometer, gyroscope, and BLE data on device's memory, 2) analyzes the context and motion data in real-time to detect changes, and 3) notifies the users when a change is detected, shows them the list of activities to choose from, and saves the label entered by them on the device's memory. Figure 5-a shows the first page of the app interface. The users need to hit "Start" to begin; the app starts storing all the sensors data and analyzing change points in real-time when "Start" is pressed. When a change is detected, the watch vibrates and the screen in Figure 5-b is shown to the user. On this page, the users can specify if the detected change corresponds to the start or end of an activity, or it could be a false positive where they can hit "Nothing". If the user does not respond to the notification after one minute it will be automatically considered as a false positive too. If the detected change is a true positive then the user can hit either "Start" or "End" and then the list depicted in Figure 5-c will be shown. This list contains a set of high-level activities of daily living that the user can choose from. The first option on this list is "Not in List" which the users can choose when their performed activity is not in the desired list of the end-application. The chosen label with its timestamp will then be saved on the watch. In case the user chooses "End" of an activity, the watch will also ask about the activity that has been started right after that. The app will ignore any change points detected within the first minute after start of an activity since persistent changes in activities are of interest for the application of activity data annotation, and it will also ignore the changes during the time that the user is responding to the app to avoid false positives.

With the app running on the watch and detecting changes in real-time, given the Polar M600's ARM Dual Core 1.2 GHz Cortex A7 CPU, no lag or delay in terms of computation was observed. The battery lasts for 12-16 hours in this case considering constant running of BLE scanning, data storage, and change point analysis. The device's memory can save all the data for up to a week using the watch for around 12 hours per day. The watch can be connected to the PC to offload the data including sensors readings and labels. The labeled data can be later used for training activity recognition algorithms or other monitoring purposes.

3.4 Parameter Adjustment

In this section we explain in more detail the effect of each parameter introduced in this algorithm and the strategy to adjust them. It must be noted that the proposed algorithm aims to detect changes in a wide range of activities regardless of the exact type of activity in an unsupervised manner. However, parameters such as window size could be further adjusted to meet the properties of activities desired for end applications. Apparently one promising approach to adjust these parameters is to set them empirically based on some supervised data. However, it will contradict the primary purpose of this study which is to help end-users to provide supervision required for training machine learning models. Therefore, we do not rely on any labeled data to adjust the parameters; instead, here we explain the effect of each parameter on the performance of the algorithm. Considering the role of each parameter, they can be easily changed in the smartwatch app's settings to meet the properties of end-applications.

We start by the window size *t*; this parameter determines the sensitivity of the algorithm to impulse and rapid changes. With smaller *t*, the algorithm will be more sensitive to rapid changes. However, reducing this parameter will increase the number of false positives since any small variation in the signal could be detected as a change point due to ignoring the long-term morphology of the signals. On the other hand, increasing *t* will increase the system's latency. We set *t=15* sec to balance between the latency and false positives due to impulse changes. The parameter *m*, which defines the number of windows to consider for change detection (see Section 3.1) affects the latency; higher values of *m* introduce longer latency. Therefore, it is better to set it to the smallest value possible. To be able to calculate inter and intra group variability of features, the minimum value of *m* should be five to have two windows in each group (see Figure 2). However, with two windows in each group, the calculation of intra-group variability will be susceptible to noise. That is why *m=7* was chosen to have three windows in each group. It is better to keep this value fixed and change window length *t* if lower latency is required in a particular application.

The duration of BLE scans is set to one minute which is the minimum number limited by the hardware used in this study. This also gives enough time to the smartwatch to complete a scan of all nearby BLEs. The threshold on IoU determines the sensitivity of the system to changes in environmental context. Lower threshold on IoU means that most of the observed BLE devices should change over two consecutive scans (i.e., low intersection) to consider it as a context change. For example, threshold of 0.6 means a context change is identified if 60% of the BLE devices change over two consecutive scans. By lowering the threshold, the number of detected changes in context will be reduced which can impact both false and true positives. It also depends on the location that the system is working in. In crowded locations with high traffic and lots of BLE devices, smaller thresholds should be applied since IoU will be normally low, whereas in less crowded places with more static BLE devices higher threshold is appropriate. The values of *F* ratio, and intra and inter group variability S_B and S_W depend on the range of features calculated from raw signals. However, we update the threshold based on the baseline of *F* ratio (Equation 5) for each user over time, so there is no need for manually selecting this parameter. Finally, the coefficient *c* used in Equation 5 to adjust the threshold based on the context changes are set based on the following logic: when a change in environmental context is detected, most likely the activity has changed, for example the user leaves their location and a context change is detected; in this case even if the motion-based change detection misses that change point, by setting the threshold close to zero (with changing *c* in Equation 5) we increase the likelihood of detecting that change. The app allows the users to change all these parameters based on their preferences although we kept them fixed during our study. Scientists using this system could change those parameters based on the aforementioned explanation to adjust it for their specific end application. An important future study could be to design unsupervised learning algorithms for automatically determining the best set of parameters such as threshold on IoU for each user leveraging a few days of their data, with the goal of maximizing accuracy and minimizing false positives.

4 Experiments and Results

To validate the performance of our context-aware change point detection algorithm and evaluate its efficacy in collecting high quality and precise activity labels in uncontrolled environments, we have conducted a real-world experiment with the designed app. In this section we start by explaining the details of the experiments and data collection. Afterwards, we investigate the performance of the proposed change detection algorithm. We will also assess the importance of context identification on the performance of change detection. Then, we compare the performance of activity recognition models trained on the data collected by our system versus unprompted self-reported labeling to represent the amount and quality of labeled data collected by our system. We also quantify the overhead of for running the app on the watch. Finally, we will evaluate the efficacy of the proposed system in facilitating the process of data annotation by exploring users' experience.

4.1 Data Collection

In this study, six participants were given the Polar M600 smartwatch to collect data of their high-level activities during normal daily life with no restriction. The participants were instructed to annotate the data on

TABLE 1: List of activities

Activity Label			
Driving	Making a phone call	Walking	Eating
Meeting	Running	Biking	Working with computer
Attending a class	Exercising	Relaxing	Shopping

the app but there was no specific instruction about the details of activities and how they should be performed. Table 1 represents the list of activities that the participants were asked to annotate in this study. Each participant was enrolled in the study for 20 days.

To compare the efficacy of our proposed change detection-based data annotation app with the conventional data annotation strategy, we asked the participants to annotate the data based on unprompted self-reporting approach for the first 10 days of data collection. In fact, during this period, the app did not notify the user for labeling; instead, they were asked to keep in mind to open the app and annotate their activities whenever they switch from one activity to another. For the next 10 days, we enabled the change point detection-based label solicitation and asked the users to only respond to the app's prompts. Please note that with this approach, users' experience during the first 10 days of data collection does not affect the second period since during the second period the users are asked to only respond to the watch prompts. All the participants were asked to wear the watch on their dominant hand. The data were stored on the watch and were transferred to a PC once a week. All the data were collected under an IRB protocol approved by the Texas A&M University.

4.2 Performance of Change Point Detection Algorithm

In this section, we will assess the performance of the change point detection algorithm. We mainly look for precision as shown in Equation 7 that tells us what percentage of detected changes are real changes.

$$precision = \frac{TP}{TP+FP} \quad (7)$$

where TP is true positives and FP is false positives. Moreover, we are interested in recall since it tells us about the amount of missed change points as shown in Equation 8.

$$recall = \frac{TP}{TP+FN} \quad (8)$$

where FN is false negatives. In fact, lower recall shows more missed change points. Finally, we look at false positive rate as the algorithm is trying to minimize that. The false positive rate is defined using Equation 9 and shows how many of non-change windows are detected as a change by the algorithms.

$$FP\text{-}rate = \frac{FP}{FP+TN} \quad (9)$$

where TN is the true negatives. To measure FN and TN we need to have access to the true labels for the missed changes; however, in the last 10 days of our data collection, where the users were asked to provide labels only in response to the app requests, we have the labels only when the system detected a change. As a result, we do not have the FN and TN for last 10 days of data collection. Therefore, the recall and FP-rate are only calculated and reported for the first 10 days of data collection.

Figure 6 shows the precision, recall, and FP-rate of our change point detection algorithm. We have also compared our method with two well-known change detection algorithms including CuSum [28] and RuLSIF [32], which were explained in Section II-B, in Figure 6. As the figure shows, our proposed algorithm outperforms CumSum by achieving 8.9% more precision and 31.9% more recall. It also achieves slightly better results than RulSIF(6.7% more recall and 26.1% more precision) while our algorithm is much simpler to be implemented on the smartwatch compared to RulSIFwhich is not appropriate to run on wearables due

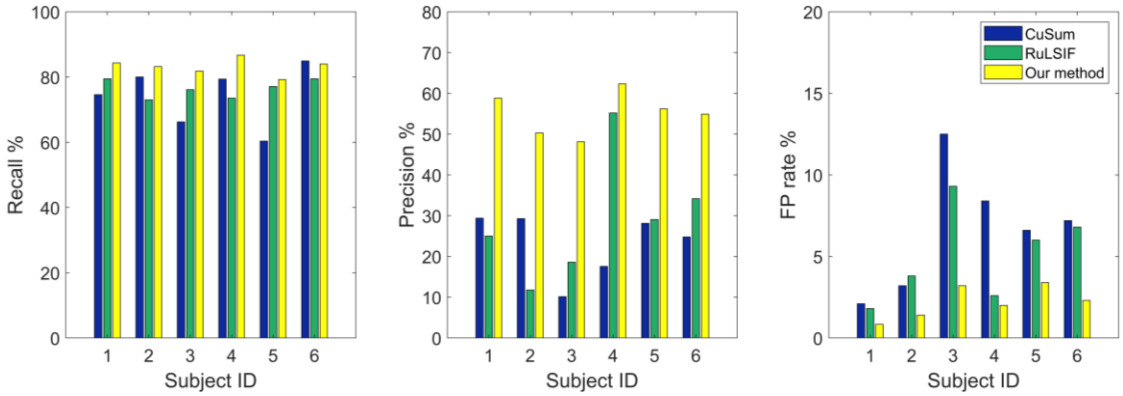


Figure 6: Comparing change point detection algorithms

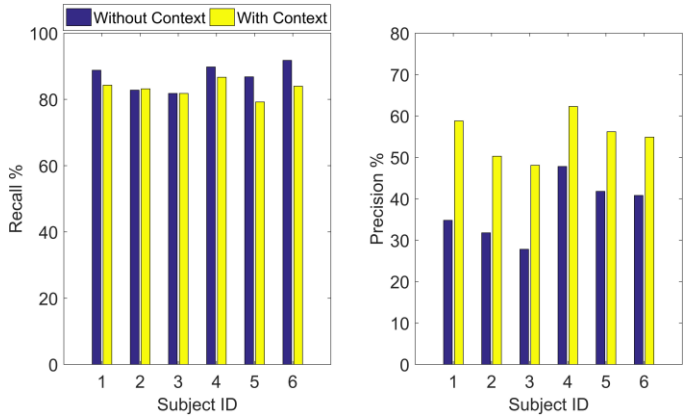


Figure 7: The effect of context identification on change point detection performance

to its complexity. Per this figure, the difference in precision is more significant due to detecting lots of false positives by the algorithms that use constant thresholds. However, our algorithm updates the threshold adaptively by leveraging the user's context which in turn reduces the false positives significantly. The FP-rate for all algorithms is relatively low (always less than 12.5%) due to having lots of TN belonging to the 15-second windows that contain no change; the average FP-rate of our algorithm is only 2.2% which is 2.8% and 4.4% less than RulSIF and CumSum respectively. Finally, it is worth mentioning that standard deviation and mean crossing rate of the magnitude of acceleration signal followed by mean of acceleration around *Z* axis (see Section 3-1) were the features that show maximum inter group distance and minimum intra group distance around the change points in our data.

In Figure 7 we compared the effect of context identification, as explained in Section 3.2, on the overall change detection performance. As the figure shows, eliminating the context and using a constant threshold (see Equation 5) increases the false positives significantly (17.6% less precision). Although the difference is more significant in the precision, the change detection with context achieves 8.0% more recall as well. These results indicates the importance of context-aware change detection in this application. In fact, by leveraging the freely available information from smart devices in the environment, which are significantly increasing due to development of IoT technology, we can understand environmental context that helps to identify transitions between activities more precisely. Reducing the false positives is vital in this application since it directly affects user compliance [14]. However, as Figure 7 illustrates, the change detection still works based on motion sensors data even without using context that shows the context is not a must-have component, but

TABLE 2: CPU, memory, and energy usage of running the app on the watch

	Context Enabled		Context Disabled	
	Data Collection with Change Detection Enabled	**Data Collection with No Change Detection**	**Data Collection with Change Detection Enabled**	**Data Collection with No Change Detection**
CPU Usage	Min: 0-10% Max: 50-60% Average ~ 40%	Min: 10-15% Max: 50-60% Average ~ 25%	Min: 0-5% Max: 40-50% Average ~ 30%	Min: 0-5% Max: 50-60% Average ~20%
Energy	Medium	Medium	Light	Light
Memory	Average: 7.4 MB	Average: 5.4 MB	Average: 5.4 MB	Average: 4.9 MB

rather a nice-to-have component of the system that helps reduce the false positive rate of the change detection algorithm, which mainly works with motion data.

Finally, we evaluate the CPU, memory, and energy usage of the proposed system when it runs on the smartwatch. For quantifying those parameters, we used Android Studio's tool 'Android Profiler'. Table 2 shows the parameters when BLE sniffing is enabled versus it is disabled. The comparison has also been done between running data collection and storage with and without change point detection algorithm running on the watch. Based on Table 2, running the change detection algorithm increases total CPU usage by 10% without context identification; this increase is more significant (15%) when the change detection algorithm leverages context identification for updating the threshold. This increase in CPU usage, which is the resource required for running change detection algorithm, is almost half of the CPU usage required for storing sensors data on the watch. We also observed that the energy consumption is dominantly affected by data reading and storage, and running the change detection algorithm slightly increases that.

4.3 Quantity and Quality of Labels Collected by Change Point Detection-based System

In this section, we evaluate the quality and usability of the labeled data collected by the change detection system. To better understand this performance, we compared the change detection based label solicitation (i.e., our proposed system) with the extensive labeling through unprompted self-reports. As mentioned in Section 4.1, all the participants were asked to perform unprompted self-report annotation, which required them to remember to annotate their data, for the first 10 days of data collection, and then they used our change-point detection based label solicitation system for next 10 days. The two metrics that we will compare here are the amount of labeled data acquired by each method and the performance of the activity recognition model trained on these data, which is associated with the quality and accuracy of the labels.

Figure 8 shows the total number of labels, number of hours of labeled data, and their averages per day obtained from unprompted self-reports versus the change point detection-based label solicitation system. As the figure depicts, the amount of labels acquired by our change point detection based label solicitation system is higher than the unprompted self-reports. The *p*-value statistic is 0.047 using t-test that shows this difference is significant and it is not random. The reason behind this difference is the fact that with unprompted self-reports we rely on the users to remember to annotate the data every time they switch between activities. However, the users may easily forget to annotate their data especially when they are busy with heavily engaged cognitive tasks. As a result, with the self-reporting approach there are many instance of desired activities that the users completely forget to annotate. It is worth mentioning that all the six participants in this study included employees who have a rather fix routines throughout the week; thus, comparing the amount of labeled data could indicate that the change point detection-based label solicitation can solve the issue with relying on unprompted self-reports, which is forgetting to annotate the activities.

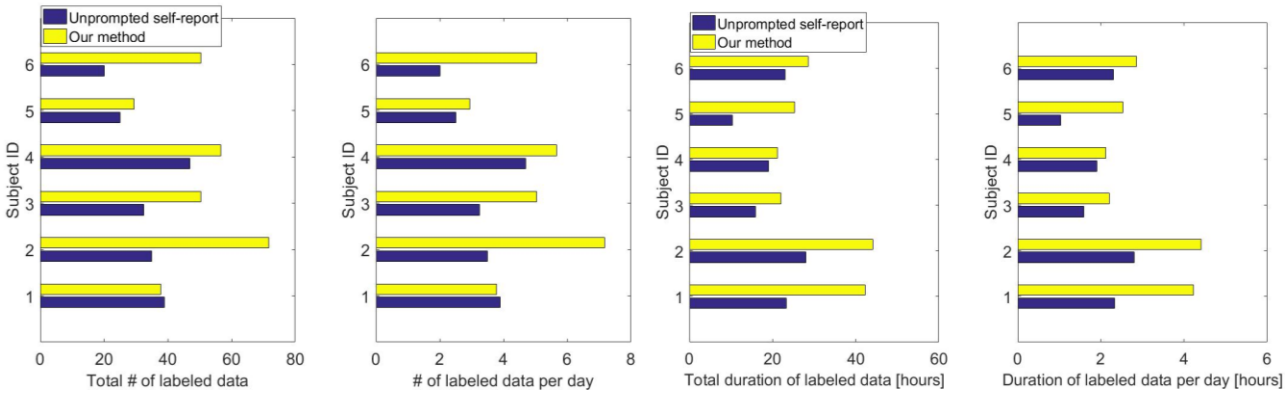


Figure 8. The amount of data obtained with unprompted self-reports vs. our method, which solicits the labels based on detected change points

To show the quality of the proposed labels, we take all the labeled data and use them to train a classifier for recognizing the activities. Note that this analysis was done offline on a PC, not the smartwatch. For this goal, we divided the data, collected with either self-reports or our system, into training and testing sets. From each dataset, i.e., the one collected with self-reports and the one collected with our change detection based label solicitation system, we used 10 hours of labeled data for training and the remaining for testing. It should be noted that we divided the data based on the number of hours because we wanted to remain fair when we compare our approach with self-reports regarding the accuracy of labels. In fact, with this, we eliminate the effect of amount of training data and we only focus on the quality and accuracy of labels. Moreover, we did not shuffle the data to remain realistic with the activity recognition results. As for training the activity recognition, we extracted a set of extensive features from the motion sensors and BLE data. The motion-based features include all the statistical features used for change detection (see Section 3.1), frequency domain features including maximum frequency and signal power in [0-0.5 Hz], [0.5-1 Hz], [1-2 Hz], and [2-5 Hz] bins, and coefficients of an autoregressive model of order 10 fitted to the acceleration signals. From BLE data we used number of unique devices as well as the rate of BLE change measured by IoU. In addition to the aforementioned features for BLE, we made a one hot encoded vector with the size of the total BLE beacons observed over each 10 days for each subject. In this vector, each element corresponds to one BLE beacon and at each sample it contains a one if that beacon is present and a zero if not. We feed this vector as an extra feature to our activity recognition model to incorporate the environmental context. We trained SVM, Random Forest, and Naïve Bayes classifier models with these features for each user.

Table 3 represents the accuracy of the activity recognition trained on these data. As the table shows, the models trained on the data that is collected with our change detection-based label solicitation system achieve 2.9% more accuracy on average compared to the ones that are trained on the data with self-reported labels.

TABLE 3: Accuracy of the activity recognition models trained on the data that is collected with unprompted self-reports versus change point detection-based label solicitation system

Model trained on the data collected with…	SVM	Random Forest	Naïve Bayes
Unprompted self-reports	74.4	76.1	70.2
Change point detection-based label solicitation	77.2	78.4	74.0

#	Question	Method 1 Unprompted self-report	Method 2 Automatic label solicitation
1	The data annotation added to my mental effort.	3.2	1.8
2	The data annotation was distracting my attention from my main task. Why?	3	1.8
3	Entering the labels was very easy.	2.6	2
4	It was hard to remember labeling.	3.4	1.8
5	I feel I missed many labels that I could have provided.	3.4	2.4
6	I found this data annotation not enjoyable.	3.4	2.2

Fig. 9. The survey questions and mean of responses (1 = strongly disagree, 2 = slightly disagree, 3 = slightly agree, 4 = strongly agree)

Moreover, this improvement is independent of the type of classifier. This shows the better quality of the data collected with change point detection-based label solicitation system. The quality or accuracy of the labels is related to that how accurate the start and end time of activities are annotated, where the end times are more important. To explain this, assume that a user forgets to enter a label at the exact moment of starting an activity, and then he remembers to enter that five minutes after the actual start time. In this case we miss a little bit of data but there is no mislabeling here. On the other hand, when the user forgets to enter the annotation at the exact time that the activity ends, and then he does that after a few minutes, the data of the next activity is mislabeled and this will negatively affect the accuracy of the machine learning algorithms that are trained on these data. The change point detection-bas ed label solicitation system helps the users to provide the annotation at the right time. Consequently, it improves the accuracy of the labels by eliminating the human errors which in turn improves the accuracy of the machine learning models that are trained on these data. An important future direction could be to combine this system with automated labeling as well as active learning methods [11], [12] in order to reduce the need for user annotation and asking them the most important questions to increase the efficiency of label annotation.

4.4 Comparing Change Point Detection-based System with Unprompted Self-reports regarding User Experience

Besides acquiring more high-quality labels with the proposed approach, an important significance of this system is to facilitate the data annotation process and decrease users' cognitive loads. In this section, we will look at the efficacy of the proposed change detection based label solicitation system regarding its objective, which is to facilitate the data annotation process, by comparing it to self-reporting approach from the users' experience perspective. For this goal, we asked the participants to fill out a short survey at the end of the data collection period. The survey contains a few questions as shown in Figure 9. The answer to the questions are a single number between 1 to 4 where 1 = strongly disagree, 2 = slightly disagree, 3 = slightly agree, and 4 = strongly agree. The participants answered all the questions for both self-reports and change point detection based label solicitation system. The mean of the responses is shown in Figure 9 too. As the figure illustrates, the self-report puts significantly more cognitive loads on the users compared to our system. People found both systems distractive but it should be noted that with the self-reporting approach the distraction come from the extra cognitive load while for the change point detection based label solicitation system the distraction is caused by the false positives. To better understand the difference between these two, we should look at the responses to Question 6 that shows the latter was significantly less annoying than the former, again emphasizing on the effectiveness of the change detection-based label solicitation system in terms of facilitating data annotation process. The current version of the algorithm ignores a window of one minute after each detected change in order to pass potential transient state and avoid bothering the user. A future study can focus on developing algorithms for reducing the likelihood of querying the user when a sequence of false positive is observed in quick succession to further address this issue. The idea of learning the pattern

of false positives and using that to adjust the threshold to optimize the user inquiry could be pursued as a future direction.

Overall, people found it much easier to provide labels with change detection-based label solicitation system since they did not have to keep in mind to provide the labels. With the self-reporting approach forgetting leads to miss the labels for activities while with the change detection based label solicitation system the algorithm may have false negatives. However, people found the rate of their own errors more than the change detection algorithm. These findings show that the proposed change detection based label solicitation system provides a great opportunity for collecting well labeled datasets of human activities in real-world scenarios as it removes the barriers related to human errors and compliance.

5 Conclusion

In this study, we proposed a new light and effective context aware, unsupervised change detection algorithm for detecting changes in human activities. The proposed algorithm was implemented on a smartwatch and could detect the moments of transition between activities. We implemented this system in the form of an app that allows the users to annotate their activity data very conveniently and accurately using a smartwatch. Through experiments in the real-world and uncontrolled scenarios we showed that the proposed system can acquire a larger amount of high quality labels that can be used for training accurate machine learning algorithms for activity recognition. We also showed that the proposed system provides a more positive user experience by facilitating the data annotation process and eliminating users' cognitive load. This system enables a new paradigm for collecting huge datasets of human activities in real-world with high quality labels. This provides a great opportunity for researchers, especially those who are working with data hungry deep learning methods. It could also be an alternative for self-reported ADL logging used in healthcare and assisted living applications.

ACKNOWLEDGMENT

This work was supported in part by the National Institute of Health, under grant 1R01EB028106. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations

REFERENCES

[1] M. Sandberg, J. Kristensson, P. Midlöv, C. Fagerström, and U. Jakobsson, "Prevalence and predictors of healthcare utilization among older people (60+): Focusing on ADL dependency and risk of depression," *Archives of gerontology and geriatrics*, vol. 54, no. 3, pp. e349--e363, 2012.

[2] D. Wang, J. Zheng, M. Kurosawa, Y. Inaba, and N. Kato, "Changes in activities of daily living (ADL) among elderly Chinese by marital status, living arrangement, and availability of healthcare over a 3-year period," *Environmental health and preventive medicine*, vol. 14, no. 2, p. 128, 2009.

[3] P. Bharti, D. De, S. Chellappan, and S. K. Das, "HuMAn: Complex Activity Recognition with Multi-modal Multi-positional Body Sensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 857–870, 2019.

[4] L. G. Clift, J. Lepley, H. Hagras, and A. F. Clark, "Autonomous computational intelligence-based behaviour recognition in security and surveillance," in *Counterterrorism, Crime Fighting, Forensics, and Surveillance Technologies II*, 2018, vol. 10802, p. 108020L.

[5] K. Davis *et al.*, "Activity recognition based on inertial sensors for ambient assisted living," in *2016 19th international conference on information fusion (fusion)*, 2016, pp. 371–378.

[6] A. Akbari, J. Wu, R. Grimsley, and R. Jafari, "Hierarchical signal segmentation and classification for accurate activity recognition," in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, 2018, pp. 1596–1605.

[7] A. Akbari and R. Jafari, "Transferring activity recognition models for new wearable sensors with deep generative domain adaptation," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, 2019, pp. 85–96.

[8] F. Ordóñez and D. Roggen, “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition,” *Sensors*, vol. 16, no. 1, p. 115, 2016.
[9] B. Hagsten, O. Svensson, and A. Gardulf, “Health-related quality of life and self-reported ability concerning ADL and IADL after hip fracture: a randomized trial,” *Acta orthopaedica*, vol. 77, no. 1, pp. 114–119, 2006.
[10] S. Aminikhanghahi, T. Wang, and D. J. Cook, “Real-time change point detection with application to smart home time series data,” *IEEE Transactions on Knowledge and Data Engineering*, 2018.
[11] R. Solis, A. Pakbin, A. Akbari, B. J. Mortazavi, and R. Jafari, “A human-centered wearable sensing platform with intelligent automated data annotation capabilities,” in *Proceedings of the International Conference on Internet of Things Design and Implementation*, 2019, pp. 255–260.
[12] A. Akbari and R. Jafari, "Personalizing Activity Recognition Models with Quantifying Different Types of Uncertainty Using Wearable Sensors," in IEEE Transactions on Biomedical Engineering, 2020. doi: 10.1109/TBME.2019.2963816
[13] S. Aminikhanghahi, M. Schmitter-Edgecombe, and D. J. Cook, “Context-aware delivery of ecological momentary assessment,” *IEEE journal of biomedical and health informatics*, 2019.
[14] T. Patterson *et al.*, “Sensor-based change detection for timely solicitation of user engagement,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2889–2900, 2016.
[15] Y. Kawahara and M. Sugiyama, “Sequential change-point detection based on direct density-ratio estimation,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 2, pp. 114–127, 2012.
[16] T. Patterson, S. McClean, C. Nugent, S. Zhang, L. Galway, and I. Cleland, “Online change detection for timely solicitation of user interaction,” in *International Conference on Ubiquitous Computing and Ambient Intelligence*, 2014, pp. 116–123.
[17] T. R. Bennett, H. C. Massey, J. Wu, S. A. Hasnain, and R. Jafari, “Motionsynthesis toolset (MoST): An open source tool and data set for human motion data synthesis and validation,” *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5365–5375, 2016.
[18] D. Roggen *et al.*, “Collecting complex activity datasets in highly rich networked sensor environments,” in *2010 Seventh international conference on networked sensing systems (INSS)*, 2010, pp. 233–240.
[19] H. Gjoreski *et al.*, “The university of sussex-huawei locomotion and transportation dataset for multimodal analytics with mobile devices,” *IEEE Access*, vol. 6, pp. 42592–42604, 2018.
[20] T. Diethe, N. Twomey, and P. A. Flach, “Active transfer learning for activity recognition.,” in *ESANN*, 2016.
[21] S. Sabato and T. Hess, “Interactive algorithms: from pool to stream,” in *Conference on Learning Theory*, 2016, pp. 1419–1439.
[22] A. Akbari, R. Solis Castilla, R. Jafari and B. J. Mortazavi, "Using Intelligent Personal Annotations to Improve Human Activity Recognition for Movements in Natural Environments," in IEEE Journal of Biomedical and Health Informatics (JBHI). doi: 10.1109/JBHI.2020.2966151
[23] T. Sztyler and H. Stuckenschmidt, “Online personalization of cross-subjects based activity recognition models on wearable devices,” in *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2017, pp. 180–189.
[24] I. Cleland *et al.*, “Evaluation of prompted annotation of activity data recorded from a smart phone,” *Sensors*, vol. 14, no. 9, pp. 15861–15879, 2014.
[25] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, “Using mobile phones to determine transportation modes,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2 , 2010.
[26] J. Muñoz-Mar\’\i, F. Bovolo, L. Gómez-Chova, L. Bruzzone, and G. Camp-Valls, “Semisupervised one-class support vector machines for classification of remote sensing data,” *IEEE transactions on geoscience and remote sensing*, vol. 48, no. 8, pp. 3188–3197, 2010.
[27] S. Aminikhanghahi and D. J. Cook, “A survey of methods for time series change point detection,” *Knowledge and information systems*, vol. 51, no. 2, pp. 339–367, 2017.
[28] D. R. Jeske, V. M. De Oca, W. Bischoff, and M. Marvasti, “Cusum techniques for timeslot sequences with applications to network surveillance,” *Computational Statistics & Data Analysis*, vol. 53, no. 12, pp. 4332–4344, 2009.
[29] X. Song, M. Wu, C. Jermaine, and S. Ranka, “Statistical change detection for multi-dimensional data,” in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 667–676.

[30] L. I. Kuncheva and W. J. Faithfull, "PCA feature extraction for change detection in multidimensional unlabeled data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 69–80, 2013.
[31] Y. Kawahara and M. Sugiyama, "Change-point detection in time-series data by direct density-ratio estimation," in *2009 SIAM International Conference on Data Mining*, 2009, pp. 389–400.
[32] S. Liu, M. Yamada, N. Collier, and M. Sugiyama, "Change-point detection in time-series data by relative density-ratio estimation," *Neural Networks*, vol. 43, pp. 72–83, 2013.
[33] A. A. Qahtan, B. Alharbi, S. Wang, and X. Zhang, "A pca-based change detection framework for multidimensional data streams: Change detection in multidimensional data streams," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 935–944.
[34] J. Wu and R. Jafari, "Orientation independent activity/gesture recognition using wearable motion sensors," *IEEE Internet of Things Journal*, 2018.
[35] E. R. Sykes, S. Pentland, and S. Nardi, "Context-aware mobile apps using iBeacons: towards smarter interactions," in *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*, 2015, pp. 120–129.
[36] M. Levandowsky and D. Winter, "Distance between sets," *Nature*, vol. 234, no. 5323, pp. 34–35, 1971.
[37] "Polar M600 | Sport smart watch for fitness | Polar Global."