MiLift: Efficient Smartwatch-Based Workout Tracking Using Automatic Segmentation

Chenguang Shen, Bo-Jhang Ho, and Mani Srivastava, *Fellow, IEEE*

Abstract—The use of smartphones and wearables as sensing devices has created innumerable context inference apps including a class of workout tracking apps. Workout data generated by mobile tracking apps can assist both users and physicians in achieving better health care, rehabilitation, and self-motivation. Previous approaches impose extra burdens on users by requiring users to select types of exercises or to start/stop sessions. In this paper, we propose MiLift, a practical end-to-end workout tracking system that performs automatic segmentation to remove user burdens. MiLift uses commercial off-the-shelf smartwatches to accurately and efficiently track both cardio and weightlifting workouts without manual inputs from users. For weightlifting tracking, MiLift supports both machine-based and free weight exercises, and proposes a lightweight repetition detection algorithm to ensure efficiency. A research study of 22 users shows that MiLift can achieve above 90 percent average precision and recall for cardio workout classification, weightlifting session detection, and weightlifting type classification. MiLift can also count repetitions of weightlifting exercises with an average error of 1.12 reps (out of an average of 9.65). Our empirical app study on a Moto 360 watch suggests that MiLift can extend watch battery lives by up to $8.25\times$ (19.13h) compared with previous approaches.

Index Terms—Context-aware inferences, mobile sensing, wearable computing

1 Introduction

The emergence of mobile sensing devices such as smartphones and wearables has enabled ubiquitous and continuous context inferences, including various types of health monitoring and workout tracking apps. With rising obesity and cardiovascular diseases linked to physical inactivity [1], such workout tracking can provide quantitative data on users' everyday activities and assist both users and physicians in achieving better health care [2], [3], rehabilitation [4], and self-motivation [5]. Compared with self-reporting, the use of mobile devices for workout tracking can provide more accurate summaries for both *cardio* and *weightlifting* exercises and avoid over- or under-estimations. Visualizing personal workout history from sensor data can help motivate users to maintain or improve workout plans and states of health. In health care and rehabilitation, physicians can also maintain progressive data of patients with the help of mobile sensors [6], [7].

Although there are a variety of approaches for mobile workout tracking, they lack the ability to *automatically* segment workout activities and therefore impose substantial burdens on users, such as requiring users to manually start/stop tracking or to select exercise types. Failure to provide timely inputs may lead to inaccurate tracking results and/or excessive energy consumption. These burdens have made prior approaches less attractive compared with simple self-reporting. For example, smartphone apps such as RunKeeper [10] and Strava [11] can only track specific types of exercises (e.g., running and biking) and require users to manually start and stop tracking. Mobile health monitoring frameworks such as Apple HealthKit [12] and Google Fit [13] leverage both smartphones and wearable devices but require users to specify the type of workouts. While most previous work focus on monitoring cardio workouts, a few have considered tracking weightlifting exercises [14], [15], [16]. However, these approaches introduce extra user burdens such as the use of multiple sensors and energy-hungry algorithms while still requiring certain degrees of user inputs for accurate tracking. Recently, there have been initial attempts to perform automatic exercise segmentation, as seen in the Pocket Track feature of RunKeeper [17], the VimoFit app [18], and RecoFit [19]. Nevertheless, they focus on limited types of exercises or do not quantify the energy consumption of continuous tracking on commercial mobile and wearable devices.

In this paper, we describe the design and implementation of MiLift, a workout tracking system that uses *automatic segmentation* to eliminate the burden on users. MiLift leverages a new generation of Android smartwatches such as the Moto 360 [20], which benefit from powerful hardware resources, Bluetooth Low Energy radios, and a rich set of sensors. MiLift can accurately and efficiently track both cardio and weightlifting exercises without requiring inputs from users. Additionally, MiLift applies context-aware optimizations and a lightweight repetition detection algorithm to efficiently track exercises on smartwatches.

- *C. Shen is with the University of California, Los Angeles, Los Angeles, CA 90095 and Facebook, Menlo Park, CA 94025. E-mail: cgshen@ucla.edu.*
- *B.-J. Ho and M. Srivastava are with the University of California, Los Angeles, Los Angeles, CA 90095. E-mail: {bo-jhang, mbs}@ucla.edu.*

Manuscript received 3 Aug. 2016; revised 30 Aug. 2017; accepted 4 Nov. 2017. Date of publication 22 Nov. 2017; date of current version 1 June 2018. (Corresponding author: Chenguang Shen.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TMC.2017.2775641

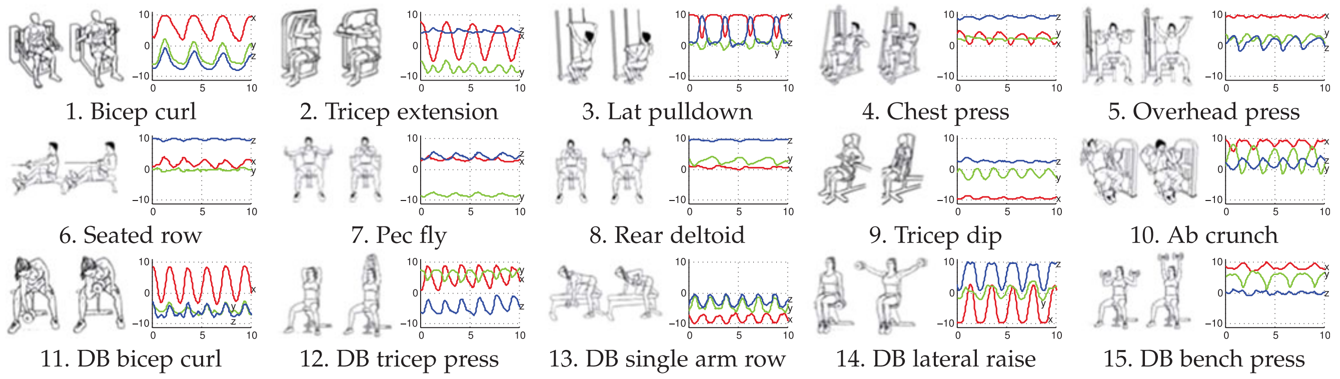


Fig. 1. Illustration of 15 weightlifting exercises considered in this paper (image sources [8], [9]) and repeating patterns in gravity sensor data traces collected from our user study. x-axis shows time in s and y-axis shows 3-axis gravity readings in $\mathrm{m/s^2}$, and x, y, z-channel are encoded in red, green, and blue, respectively (same for the rest of this paper). Type 1-10 are machine exercises, and 11-15 are dumbbell (DB) free weight exercises.

1.1 Research Contribution

We highlight the research contributions of MiLift as follows, achieved by using only a single smartwatch:

First, MiLift can automatically segment both cardio workouts and weightlifting exercises from non-workout activities using a two-stage classification model. It is the first system to apply and fully evaluate such an automated algorithm in workout tracking. Unlike previous tracking approaches, users do not need to provide any manual inputs to MiLift, such as selecting types of exercises or starting/stopping exercise sessions. MiLift runs in the background on a smartwatch and also provides a UI to visualize the workout summary of users for management. From our user study, MiLift's automatic segmentation feature is proven valuable to individuals who regularly perform gym exercises.

Second, MiLift can track both weightlifting machine and free weight (dumbbell) exercises, as shown in Fig. 1. MiLift meets real-world user requirements during weightlifting exercises, including (1) automatically detect the start and stop of weightlifting sessions (sets); (2) count repetitions (reps) of exercises; (3) classify the type of exercises. Our evaluation on a dataset of 2528 sets of weightlifting exercises (24408 reps) collected by 22 users shows that MiLift can achieve above 90 percent average precision and recall for both weightlifting session detection and exercise type classification. The average error of rep counting in MiLift is 1.12 reps (out of an average of 9.65). Weightlifting detection in MiLift requires no model training and is user-independent.

Finally, to achieve efficient resource usage, MiLift employs two novel techniques on wearable context inferences: context-aware optimization, and a lightweight revisit-based weightlifting detection algorithm. Our experiments on a Moto 360 smartwatch indicate that watch battery lives can be extended by up to $8.25\times$ (19.13h) by running MiLift instead of unoptimized tracking apps. Even with a continuous execution of MiLift, the watch battery can last for more than a day and therefore will not require extra charging by users.

The MiLift app is open-source and available online at https://github.com/nesl/WorkoutTracking.

2 Challenges and Design Choices

We discuss four key challenges towards an autonomous and efficient workout tracking system and highlight the design choices made in MiLift.

Although prior works have tackled each individual challenge to a certain extent, none of them can address the four challenges simultaneously. For example, previous workout tracking systems often need multiple body sensors, require manual start/stop by users, only track cardio exercises but not repeating weightlifting exercises, and often significantly reduce the battery life of devices. In contrast, MiLift proposes a comprehensive workout tracking system that addresses the four challenges altogether and provides a practical end-to-end system to users.

2.1 C1: Single-Device Sensing

Workout tracking apps running on smartphones cannot accurately sense user activities when the phone is placed away from the user. Moreover, workout exercises typically involve movements of different body segments such as hands, arms, waists, and legs and cannot be accurately monitored by a single smartphone. Prior tracking algorithms either placed more than one sensing device (e.g., a phone and a watch or multiple wearable sensors) on a user [14], [21], [22] or required instrumentation of weightlifting equipment [16]. In contrast, smartwatches are less intrusive since most users wear them for the majority of the day. Watches can sense wrist orientations and partial torso movements whereas smartphones, typically carried in pockets, can only capture body postures. Therefore MiLift uses a single smartwatch to replace smartphones and other sensors previously used for workout tracking.

2.2 C2: Automatic Segmentation

Most workout tracking apps require users to manually choose workout types and start/stop the tracking of each session. If a user fails to mark the session end in time or even forgets to do so, the tracking algorithm could overestimate the current session and/or consume excessive energy. Although prior work proposed different classification models to recognize the presence of exercises, to identify exercise types, and/or to quantify the amount of exercises, none of them combined these models in an end-to-end system on commercial devices to automatically segment user activities. To eliminate user burdens, MiLift can detect a user's activity transitions and automatically segment different workout exercises using a two-stage classification model: it first applies a lightweight classifier on low-power inertial

TABLE 1
Summary of Prior Workout Tracking Approaches and Whether They Meet Our Design Challenges

Category	C1	C2	C3	C4
Mobile workout tracking apps	●			○
Mobile health frameworks	●	○		●
Weightlifting tracking systems		○	●	
Emerging apps (VimoFit [18])	●		●	
MiLift (this paper)	●	●	●	●

○ *denotes partial fulfillment.*

sensor data to determine high-level user activities such as non-workout, walking, running, and weightlifting, and only starts detailed weightlifting analysis upon detection of weightlifting exercises. The multi-stage model is also motivated by prior work on hierarchal activity classification [23], [24], [25].

2.3 C3: Weightlifting Exercise Tracking

Medical researchers have shown that weightlifting exercises (or weight training, strength training) can help improve metabolic function and muscle strength [26], [27]. Although *free weight exercises*, such as those using dumbbells and barbells, can sometimes lead to greater muscle activities than *machine-based weightlifting exercises* [28], both should be combined to maximize training outcome [26], [29]. Most workout tracking apps monitor cardio activities such as walking and running, but few can efficiently track weightlifting exercises, including the following metrics:

- Number of *sets*: each set is a workout session that includes several repetitions of the same weightlifting exercise.
- Number of repetitions (*reps*) in a set: each rep is an instance and the basic unit of a particular weightlifting exercise.
- *Type* of the exercise: for example, dumbbell bicep curl.

With noises such as those caused by improper exercise forms or non-workout activities in-between exercises, it remains challenging to accurately and efficiently capture repeating human movements during exercises. MiLift exploits repeating patterns of human arms during weightlifting exercises as demonstrated in Fig. 1, and performs weightlifting classification include set detection, rep counting, and exercise type classification. MiLift considers 10 types of weightlifting machine exercises (#1-#10) and 5 types of dumbbell-based free weight exercises (#11-#15).

2.4 C4: Efficient Resource Usage

The limited battery capacity of mobile devices calls for a detailed study and optimization of energy consumption of workout tracking services. We propose that the battery life of a smartwatch should last for at least 16 hours (a full day except sleeping) even with continuous workout tracking, so that users do not need to charge the watch during the day. Previous approaches can rapidly drain out device batteries because of the continuous nature of inference executions and the use of complex algorithms for weightlifting tracking such as Dynamic Time Wrapping (DTW) [30]. MiLift applies two techniques to achieve efficient resource usage on watches: 1) context-aware optimizations, and 2) a lightweight algorithm for weightlifting detection.

3 Related Work

Mobile apps explore a variety of human contexts including transportation modality [6], social interactions [31], [32], and physical and mental healthiness [33], [34]. Recently the emergence of Apple Watch [35], Android smartwatches (e.g., the Moto 360), and Microsoft Band [36] also prompts the development of context inferences on smartwatches and wearable devices [15], [37], [38].

We group prior approaches on workout tracking and management using mobile devices into the following categories (summarized in Table 1):

Mobile workout tracking apps including RunKeeper [10], Strava [11], and MapMyRun [39] focus on cardio exercise tracking and management using a single smartphone. Due to the limited sensor coverage when using phones, these applications only support specific types of cardio exercises and require users to manually start and stop workout sessions.

Mobile health frameworks such as Google Fit [13], Apple HealthKit [12], and Microsoft Health [40] provide APIs for both app developers and data scientists. Each framework also provides an app for users to manage workout tracking. Typically built-in as part of the mobile operating systems, the resource usage of these frameworks are well optimized. However, they mostly focus on cardio exercises and do not support tracking of weightlifting exercises. The front-end apps require manual selection of workout types as well.

Weightlifting tracking systems. The weightlifting classification system in MiLift is motivated by several prior work in this space. myHealthAssistantt [41] and Chang et al. [14] employ multiple wearable sensors and a smartphone as a hub to track weightlifting exercises. MyoVibe [22] embeds wearable sensors in fitness clothing and leverages muscle vibrations to estimate fatigues during exercises. FEMO [16] tracks weightlifting exercises by instrumenting dumbbells with RFID sensors and measuring frequency shifts caused by Doppler Effect. Repeating sensor signals in workout sessions can be recognized by instance-based classifiers [15], and Dynamic Time Wrapping [30]. NuActiv [21] develops a zero-shot attribute-based learning algorithm for recognizing and memorizing unseen exercises. MyoBuddy [42] leverages electromyograms for detecting weights being lifted. RecoFit [19] supports labeling exercise periods and counting repetitions, but does not study the feasibility of running continuous tracking on mobile devices. In contrast, MiLift is the first end-to-end system on commercial smartwatches that uses automatic segmentation to track both cardio and weightlifting exercises without requiring users to start/stop tracking and/or select workout types. MiLift also tackles challenges such as single device sensing and efficient resource usage described in Section 2.

Emerging apps and web services including the VimoFit app [18] and the Atlas wristband [43] aim at autonomous workout tracking. Although some of them support rep counting for guided workouts, they still require certain manual inputs from users. Section 6.3 shows the high

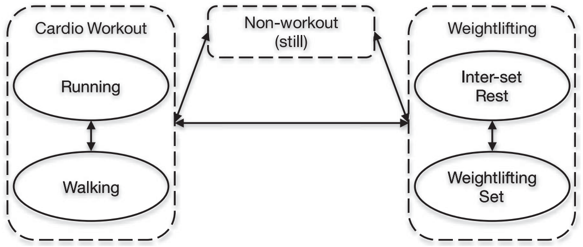


Fig. 2. State transitions of a user's workout activities.

energy consumption of VimoFit and compares it with MiLift. Other services such as JEFIT [44], WorkoutLabs [8], and Gymwolf [9] provide workout management and guidance from self-report workout data.

4 System Architecture

In this section, we propose the system architecture of MiLift, describe the implementation of a two-stage classification model, and discuss optimization techniques.

4.1 Overview

MiLift aims to track workout activities of a user using only a smartwatch (C1). Exercises of a user can be categorized into three groups: non-workout (still), cardio workouts such as walking and running, and weightlifting. Fig. 2 describes state transitions of these exercises. A user's state can transit among the three exercise groups, for example, a user can start weightlifting or cardio workouts from the non-workout state. State transitions also take place within each exercise group, for example, a user can switch between walking and running during a cardio session, take rests between weightlifting sets, or perform different weightlifting exercises in consecutive sessions. Motivated by the state transitions of user activities, MiLift develops a two-stage classification model to accurately and efficiently track workout activities, shown in Fig. 3. The model contains two stages:

S1: High-level activity classification. S1 aims at detecting high-level activity transitions. It implements a lightweight algorithm to label a data window with activities including non-workout, walking, running, and weightlifting. If walking or running is detected, session duration and step counts are logged. Once weightlifting is detected, MiLift wakes up the weightlifting classification module described below which involves more complicated computations.

S2: Weightlifting classification. This module analyzes inertial data and performs detailed weightlifting classification (C3) including set detection, rep counting, and type classification. We have implemented an autocorrelation-based algorithm and a lightweight revisit-based algorithm to achieve efficient resource usage (C4).

To perform automatic segmentation on user activities and eliminate the burden on users to manually start/stop tracking of sessions (C2), MiLift transits between S1 and S2 based on current user contexts. The state transition also helps preserve battery energy of smartwatches (C4).

4.2 High-Level Activity Classification

In MiLift, inertial data sampled from smartwatches are first labeled by a high-level activity classifier. Motivated by prior

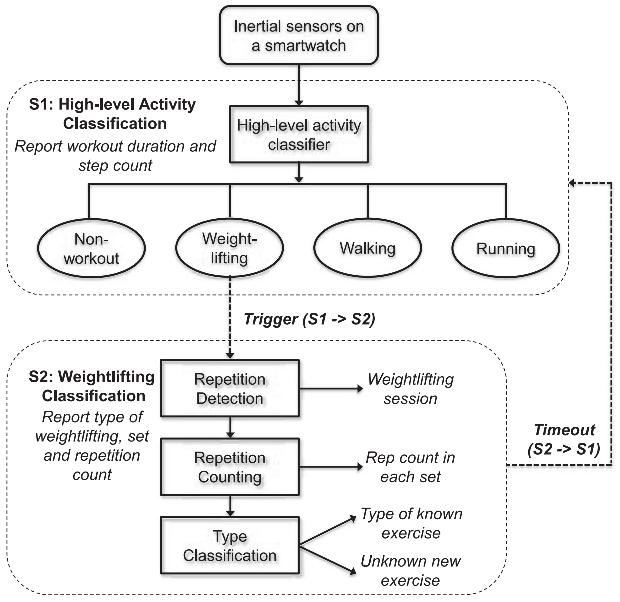

Inertial sensors on a smartwatch
S1: High-level Activity Classification
Report workout duration and step count
High-level activity classifier
Non-workout
Weight-lifting
Walking
Running
Trigger (S1 -> S2)
S2: Weightlifting Classification
Report type of weightlifting, set and repetition count
Repetition Detection
Weightlifting session
Repetition Counting
Rep count in each set
Type Classification
Type of known exercise
Unknown new exercise
Timeout (S2 -> S1)

Fig. 3. MiLift architecture and state transitions.

work on mobile sensing [6], [45], [46], MiLift first segments 3-axis accelerometer data into 1-second windows, and classifies the high-level activity label corresponding to each window. The activities include *non-workout, walking, running, weightlifting*. The classification pipeline consists of:

Sampling and preprocessing. MiLift uses a 1- second classification window on accelerometer data sampled at 50 Hz. The choice of 1- second window size is also seen in previous activity recognition work [6], [46] so that the data window contains enough samples for feature functions but also keeps only one type of activity in a single window. Data is buffered for each second and then sent to the next stage for feature extraction.

Feature extraction. This submodule takes a 1-second window of accelerometer data and reduces its dimension by applying a set of feature functions in both time and frequency domains. Features considered in our system are mean, variance, range, root mean square (RMS), mean absolute deviation (MAD), magnitude, skewness, kurtosis, quartiles, median, and energy coefficients between 1-5 Hz from Discrete Fourier Transform (DFT). For each feature, MiLift fuses three accelerometer axes by computing on each axis separately and then taking an average. To improve performance and reduce feature calculation workload, we apply two feature selection algorithms implemented in scikit-learn [47] including the univariate statistical test and the tree-based feature ranking. These two selection algorithms rank features based on their significances and select 7 of them for use in MiLift, including mean, standard deviation, MAD, range, the 1st quartile, the 2nd quartile, and DFT at 5 Hz. The feature vector is then sent to the next stage for classification.

Classification. The classification module takes a feature vector as input and generates an activity label for each window (i.e., every second). We have implemented two categories of classification models. The first approach uses Conditional Random Fields (CRF) to continuously label each data window represented by the feature vector. CRF is commonly used for sequence labeling tasks such as part-of-speech tagging and image segmentation [48]. Prior studies

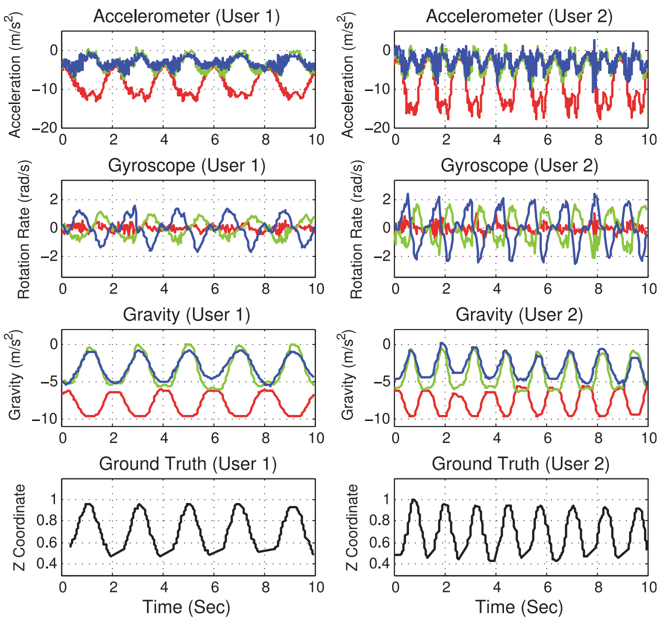


Fig. 4. Sensor traces comparison of dumbbell single arm row performed by two users, showing that gravity sensor can best demonstrate the repeating patterns.

also use CRF for tasks on sensor data such as room occupancy inference from motion sensors and human activity recognition from wearable accelerometers [49], [50], [51].

In MiLift, CRF is used to exploit the temporal correlation among workout activities. Each state y_t in CRF corresponds to a ground truth activity label at time t, and each accelerometer feature vector is used as an observation x_t. The joint probability of a state sequence $\mathbf{y}$ and an observation sequence $\mathbf{x}$ is modeled as

$$p_\lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\lambda(\mathbf{x})} \cdot \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i \mathbf{f}_i(\mathbf{y}_j, \mathbf{x}_j)\right),$$

where n is the total time considered, $\mathbf{f}$ is a set of m feature functions internally used by the CRF (not to be confused with our accelerometer features), λ is a weight vector for all feature functions, and Z serves as a normalization term. Because it is much more difficult to learn $\mathbf{f}$ when it is a continuous function over a multi-dimensional variable, we use K-means algorithm to discretize the feature space into clusters and use corresponding cluster IDs as the input observation x_t to CRF. We tested different numbers of clusters ranging from 4 to 20 and picked the one with the best CRF training performance.

In addition to CRF, our second approach of high-level activity classification is to label each 1-second window by an instance classifier, and then apply Hidden Markov Models (HMM) to smooth the label series. The instance classifiers we considered are Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM). The parameters of all models are tuned in the training stage, such as maximum depths for DT, number of trees and size of feature subsets used for each split for RF, and kernel types for SVM.

These instance models label each 1-second window independently. However, adjacent windows are temporal-correlated as workout activities are continuous and would not transit frequently within a short period, which potentially causes high false positive because these models treat all $1s$ windows independently. For example, our system should not classify transient movements, such as raising one's arm while sitting in an office, as weightlifting activities. To capture temporal correlation and filter out unlikely activity transitions, we apply an HMM classifier to smooth output activity labels of instance classifiers and filter out unlikely activity transitions for better accuracy. To generate the HMM model, we use ground truth activity labels as hidden states and output labels from instance classifiers as observations.

4.3 Weightlifting Classification

The second state in the two-stage classification model of MiLift is a weightlifting classification module. It is waken up by the high-level activity classifier when users are performing weightlifting exercises. This module achieves three tasks: (1) detecting weightlifting sessions and labeling boundaries of *sets*, (2) counting number of *reps* (repetitions) in each set, and (3) classifying type of current weightlifting exercise. The weightlifting classification module must conserve battery energy of smartwatches (C4).

4.3.1 *Sensor Choice*

There are several available inertial sensors on a smartwatch that can capture human motions, including but not limit to accelerometers and gyroscopes. A gravity sensor is a low-power software sensor fusing both accelerometer and gyroscope [52]. Gravity sensor data can reflect wrist orientations of users and can be a good indicator of repeating weightlifting exercises with wrist movements. Using the dumbbell single arm row exercise as an example, Fig. 4 demonstrates the ability of different sensors to capture weightlifting exercises. Sensor data traces collected from two users each performing one set of exercises are compared with ground truth motion traces captured by an OptiTrack Prime 13 tracking system [53]. For both users, the gravity data is less noisy than the accelerometer and gyroscope data and can better highlight the repeating pattern in signal traces generated by weightlifting exercises. Moreover, amplitudes of the gravity data across two users are more consistent compared with the other two sensors, indicating that gravity sensor can better identify types of weightlifting exercises across different users. Therefore MiLift uses the Android gravity sensor for weightlifting detection.

Fig. 5a demonstrates a trace of gravity sensor data during a user's weightlifting exercise session (bicep curl machine). This session can be clearly identified by repeating patterns in the gravity sensor trace. Moreover, since the corresponding gravity data are cyclical, each rep in this session can be identified and the number of reps can be counted. In contrast, we see arbitrary patterns recorded during non-workout periods shown in Fig. 5b. Therefore, MiLift quantifies repeating patterns in gravity data to detect weightlifting sessions and count reps, using two approaches including an autocorrelation-based algorithm and a revisit-based algorithm. MiLift then classifies the type of the detected weightlifting activity.

4.3.2 *Autocorrelation-Based Weightlifting Detection*

Weightlifting session detection. Autocorrelation is a technique for examining the periodicity of a signal series. It calculates

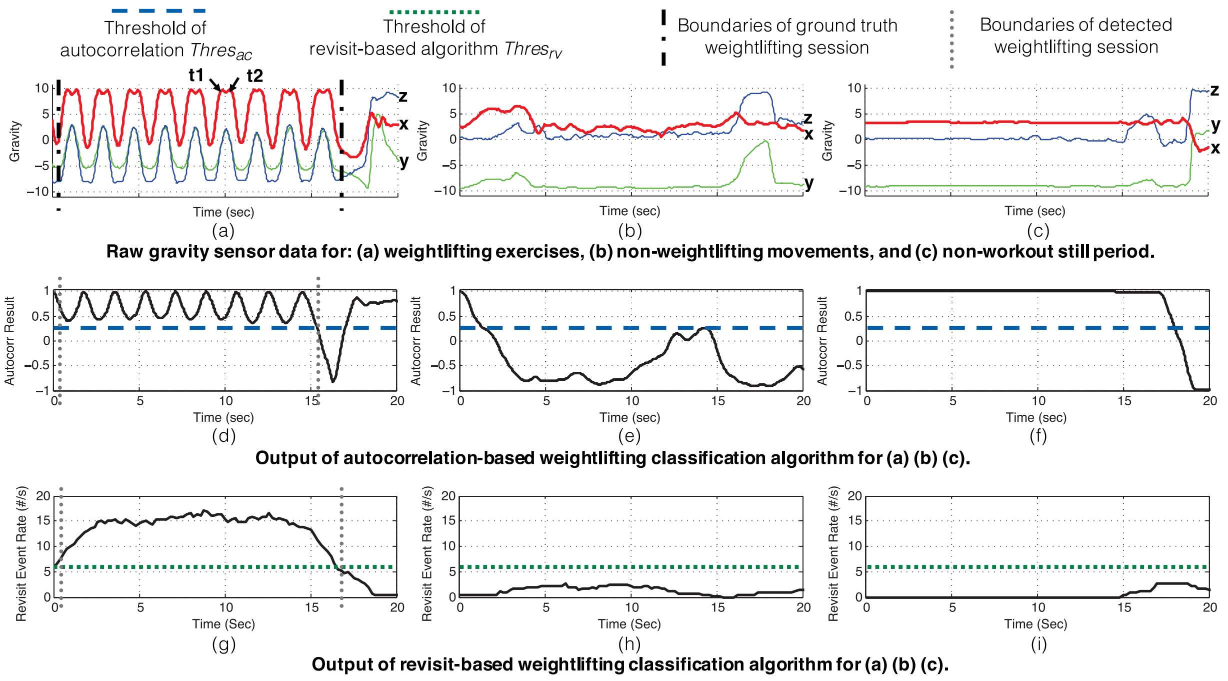


Fig. 5. Results of applying autocorrelation-based algorithm and revisit-based algorithm on gravity sensor data for three cases: weightlifting (bicep curl), non-weightlifting movements, and non-workout still period.

correlation between a sample window at time t and another same-size window with an offset in time known as the *lag* ℓ. The output of an autocorrelation series can be represented as a function of ℓ:

$$\mathbf{R}_\ell = \frac{S_t^{\mathbf{T}} S_{t-\ell}}{|S_t| \cdot |S_{t-\ell}|},$$

where S_t is a vector representing a signal window with w elements starting from time t. If a sample window and another window with lag ℓ have similar signal patterns, the autocorrelation value $\mathbf{R}_\ell$ will be close to 1 indicating significant similarity between these two windows. We apply a threshold value $Thres_{ac}$ on $\mathbf{R}_\ell$ to identify weightlifting sessions. In order to capture periodicity of weightlifting exercises, we choose a small window size but longer than the typical duration of a rep ($6s$ from our analysis) to cover all possible rep lengths.

For weightlifting session detection, we apply autocorrelation using $1s$ sliding windows on gravity data to capture all possible sessions. Figs. 5d, 5e, and 5f plot autocorrelation results when applied on three different cases of sensor readings shown in Figs. 5a, 5b, and 5c. The figures also show the threshold $Thres_{ac}$ used for repetition detection as well as both ground truth and detected weightlifting sessions. When autocorrelation is performed on a weightlifting session, the output values are close to 1 and demonstrate spike patterns (shown in Fig. 5d). A peak in autocorrelation results is aligned with a valley in raw data because this is the beginning of a new rep and autocorrelation discovers the maximum similarity between current window and the previous one. MiLift applies $Thres_{ac}$ on peaks from autocorrelation results to identify a weightlifting session. In contrast, when the input signal trace indicates no repeating wrist motion, for example when a user is adjusting the weights, the autocorrelation output values show no peaks and are relatively low (shown in Fig. 5e). However, one exception exists that can lead to constant high autocorrelation values. When a user's wrist is stationary during breaks and gravity readings are relatively constant, the output values can remain close to 1 (shown in Fig. 5f). Our algorithm filters out such cases by removing sessions with consistently high autocorrelation values but no spike patterns.

Another issue in this algorithm is which axis should be consider for autocorrelation. From the gravity signals shown in Fig. 1, there is no universally dominant axis with the most obvious peak-valley patterns. In MiLift, we explored three techniques to select the best gravity axis: (1) summing up absolute values of each axis; (2) taking absolute values of sums of each axis; (3) performing autocorrelation on each axis separately and choosing the one with the best result. Our experiment indicates the last strategy is the most robust against false positives and demonstrates the best accuracy among the three.

Rep counting. After weightlifting sessions are detected and labeled, the rep counting part is achieved by simply performing a naive peak detection on autocorrelation results. Because non-repeating signals are already filtered out, the number of reps can be derived by simply counting peaks.

4.3.3 Revisit-Based Weightlifting Detection

Although the autocorrelation-based algorithm can effectively detect repetitions in time-series data, it can incur $O(wL)$ overhead upon arrival of each new gravity sample where L is the maximum lag of interest and w is the window size, leading to heavy computations (challenge C4). We propose a lightweight algorithm which quantifies revisit events in gravity sensor data to detect weightlifting sessions and count reps with less computation and better efficiency.

Weightlifting session detection. Fig. 5a shows that gravity signal demonstrates repeating patterns during weightlifting sessions. This implies that gravity sensor readings (i.e., $(grav_x, grav_y, grav_z)$) with similar values can be found

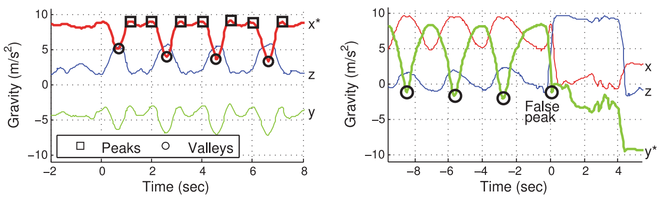


Fig. 6. Two cases that can lead to failures of naive peak detection. Left: For bicep curl, if we only consider upper peaks, each rep will be counted twice. Right: For ab crunch, the weightlifting session boundary looks similar to a rep leading to a false count. In both graphs, the dominant axis is marked with stars.

within a short time, typically no longer than the period of a rep. For example, the gravity reading marked as $t1$ has a similar value to sample $t2$, with a time difference about one second. A necessary condition of a weightlifting session is that repeating patterns of sensor readings are seen within a short time window whose length is similar to the typical length of one rep (6 seconds from our analysis), indicating that the majority of gravity samples in this window have a similar sample within the same window. However, the strategy of comparing sensor data windows whenever a new sample arrives can cause high overhead, as seen in autocorrelation. To cope with this, we present a heuristic algorithm to detect repeating patterns in the signal stream, called the *revisit-based approach*.

First, each sample $(grav_x, grav_y, grav_z)$ is discretized at an interval of I, i.e.,

$$D = \left(\lfloor\frac{grav_x}{I}\rfloor, \lfloor\frac{grav_y}{I}\rfloor, \lfloor\frac{grav_z}{I}\rfloor\right).$$

Two similar sensor readings will share the same discretized D values. Our experiments indicate the best choice of I is 0.6.

Second, we maintain a hash table $\mathcal{H}$ to store incoming samples where D is used as the key and timestamp t of current sample is used as the value

$$\mathcal{H}[D] = t.$$

Next, we define that a *revisit event* occurs when a hash collision happens. For instance, when a second sample with a key D arrives at time t, but D already exists in $\mathcal{H}$ with a value of t' ($\mathcal{H}[D] = t'$). The *revisit time frame* $T_{revisit}$ of this event is defined as the time difference between two samples with the same discretized value D

$$T_{revisit} = t - t'.$$

We only keep the revisit events whose $T_{revisit}$ is than 6 seconds, which is the duration of the longest rep identified in our dataset. $\mathcal{H}[D]$ is then updated with value t.

Finally, we define *revisit event generation rate* to represent the number of revisit events generated in the past 1 second at a given time. A high revisit event generation rate suggests that revisit events occur frequently within a short period of time and thus indicating the occurrence of repeating patterns, which can then identify an ongoing weightlifting session. We apply a threshold value $Thres_{rv}$ on the revisit event generation rate of each second to select significant (high) values.

The above revisit-based algorithm only takes $O(1)$ time to update the revisit event generation rate when a new sensor reading arrives and can significantly reduce computational cost compared with the $O(wL)$ overhead incurred for each new sample by the autocorrelation-based algorithm.

Figs. 5g, 5h, and 5i plot results of our revisit-based algorithm on three different input traces shown in Figs. 5a, 5b, and 5c. The figures also show the threshold $Thres_{rv}$ used for repetition detection as well as both ground truth and detected weightlifting sessions. If input signals have cyclical patterns due to repeating motions, the revisit event generation rate raises and remains high until the end of current weightlifting session (shown in Fig. 5g). In contrast, revisit event generation rates are much lower if there is no repeating pattern in the input trace (shown in Fig. 5h). However, whenever a user is stationary and the gravity readings are relatively constant, revisit events will be generated at a high rate equal to the sensor sampling rate. Our algorithm excludes this situation by discarding any sensor reading that has the same discretized value D as its immediate predecessor. As shown in Fig. 5i, this effectively prevents stationary motions from being detected as weightlifting.

Rep counting. After the boundaries of a weightlifting session are detected, our revisit-based algorithm performs peak detection on raw gravity sensor data for rep counting. However, it is a more challenging task than the peak detection in the autocorrelation-based algorithm because raw data are noisy and simply counting peaks can lead to overestimation of the number of reps.

There are three issues preventing an accurate rep counting using naive peak detection on raw gravity data. First, we do not know which axis of gravity data has the most distinct repeating patterns. Second, neither upper peaks nor bottom peaks (valleys) always better indicate repeating patterns. As shown in Fig. 6 left, simply counting peaks yields double-counting errors because the upper part in the sensor signal of a bicep curl rep sink in the middle (i.e., showing reversed w-shapes), resulting in counting each rep twice due to the two peaks. Third, the weightlifting session boundaries detected above may not be precise. Fig. 6 right shows that if we consider bottom peak (valley) values, the valley at $t = 0$ satisfies all constraints even if it is not a rep but the end of this session.

To address the first issue, we select the axis with the largest range between the maximum and minimum value within the session to reduce ambiguities during rep counting. Next, to decide whether the number of reps should be derived from counting peaks or valleys, we use vertical displacements within a small time delta (i.e., second derivatives) to capture spikiness of peaks/valleys where a larger displacement indicates a spikier peak/valley. For both peaks and valleys, we compute the average vertical displacement V and count the one with larger V as number of reps to improve the counting accuracy. In the example of Fig. 6 left, valleys should be considered since they have larger vertical displacements than peaks. Finally, to reduce false rep counts from weightlifting session boundaries, we consider all the three axes when ambiguity occurs close to the boundaries. Taking Fig. 6 right as an example, the z value at $t = 0$ is different from all other detected valleys, allowing us to remove it as a false rep count.

Both the weightlifting detection algorithms require no model training. Because neither algorithm considers any user-specific feature, our proposed weightlifting algorithms are user-independent as well. Note that weightlifting exercises in real life could lead to noises to the sensed gravity signal, such as those caused by tiredness or improper exercise forms, as discussed in Section 6.1.2, and therefore lead to errors of both algorithms.

4.3.4 Back-to-Back Weightlifting Sessions

Although there is usually a transition period in-between sessions, for example, taking a one-minute break or walking to a different machine/equipment, MiLift does not rely on the transition period to detect consecutive workout sessions, such as performing dumbbell tricep press right after doing bench press. When a user switches to a different type of exercise, both the autocorrelation- and revisit-based detection algorithms can recognize the difference in gravity signal (shown in Fig. 1). MiLift can then decide the boundary of the two sessions and correctly segment consecutive sessions.

4.3.5 Weightlifting Type Classification

In routine weightlifting exercises, a user is not only interested in statistics about sets and reps, but also the type of weightlifting exercises performed in each set. Once weightlifting sessions are detected and numbers of reps are calculated, MiLift starts weightlifting type classification and labels each detected session with an exercise type.

Our intuition for weightlifting type classification is that wrist positions of a user during different weightlifting exercises will lead to unique orientations of a smartwatch. To quantify watch positions, MiLift aggregates 3-axis gravity sensor readings from the current weightlifting session and computes a set of features for each axis, including mean, standard deviation, minimum, maximum, and range. MiLift then applies an SVM classifier to label the type of the current session. We choose SVM here because of the relative simplicity of this classification problem and the ability of SVM to generate confidence scores for each class of types, which can be used to determine if an exercise type is unseen. To tune performance of the SVM classifier we have considered different kernels such as linear, Gaussian (with different radius r), and polynomial (with different degree d).

Another significant aspect of type classification is the ability to determine whether a new sample instance belongs to known exercise types during training or a new type of exercise. To identify new types, MiLift takes advantage of confidence scores reported by the SVM classifier. For each incoming instance, a vector of confidence scores is calculated by the SVM, representing the probability that this instance belongs to each known type class. If the maximum probability in this vector falls below a certain threshold $Thres_{conf}$, MiLift determines this instance belongs to a new type of weightlifting exercise and asks the user for a correct label.

4.4 Context-Aware Optimization

To achieve both automatic segmentation (C2) and efficient resource usage (C4), MiLift takes advantage of available user contexts and applies a context-aware state transition mechanism. Fig. 3 shows the transition among two classification states, *S1* and *S2*.

S1: MiLift executes the high-level activity classification in this state. It involves sampling of low-power accelerometer and incurs only lightweight computation and low energy consumption. Because a user typically keeps the same activity for a period of time and will not switch activities frequently within seconds, MiLift duty-cycles the execution of S1 by a 20 percent schedule. MiLift classifies a 1-second window every 5 seconds so that it is able to capture most activity transitions while conserving battery energy. Note we design the 20 percent duty-cycle schedule as a system parameter and it can be changed based on user preferences and current battery states.

Moreover, to prevent everyday activities from being misclassified as gym exercises (i.e., reducing false positives), S1 opportunistically leverages coarse location contexts such as those inferred from network accesses or WiFi signatures. For a given user, workout exercises mostly take place near similar locations (e.g., a gym) or WiFi networks (e.g., gym WiFi hotspots). MiLift can automatically learn the typical exercise location of a user after the first few app uses and from WiFi connection histories. If a coarse location is known from recent queries or WiFi scans initiated by the OS or other apps, it can be compared with the user's typical exercise locations. Once weightlifting is detected by S1, only when the two locations match will MiLift transit to S2. The opportunistic use of coarse locations as opposed to querying exact GPS coordinates ensures no additional energy is consumed.

S2: MiLift performs the more sophisticated weightlifting classification in this state. Although S2 is designed to be lightweight and efficient, the complexity of this state is relatively higher than S1 because of the computation incurred by each new sensor reading. Therefore it is triggered by S1 and only starts executing upon detection of weightlifting exercises. When no new weightlifting exercise is detected for a certain timeout (e.g., 5 min), MiLift transits back to S1 and performs the simpler high-level activity classification.

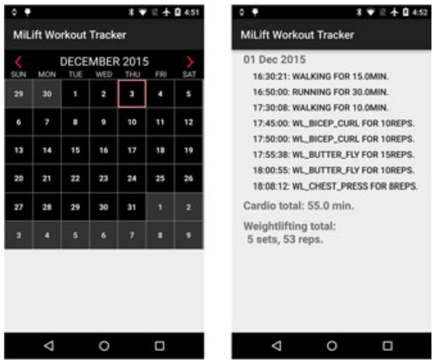
Although performing duty-cycled classification will not prevent the watch from being waken up frequently, i.e., the watch will still have to perform sensing and classification in S1 every 5 seconds, Sections 6.2 and 6.3 have proven such an optimization to be effective since the watch can last for more than a full day with this two-stage classification model running continuously. This is because most of watch battery energy is spent on data sampling and computation rather than simply remaining power-on. Nevertheless, we acknowledge that the use of low-power co-processors (e.g., DSPs and dedicated sensor hubs) for always-on sensing and computation tasks will further reduce the energy consumption of context inferences, as discussed in Section 7.

5 IMPLEMENTATION

Hardware device. The commercial off-the-shelf watches we used for data collection include Moto 360, Moto 360 Sport [54], LG G Watch R [55], and ASUS ZenWatch 2 [56]. For experiments on energy profiling we used the Moto 360 which has a 1 GHz OMAP3 CPU, 512 MB RAM, 4G flash storage, and a 3.8V/320 mAh Lithium-ion battery. We have implemented a smartwatch data recording app that logs accelerometer and gravity sensors from the watch. Users were also assisted by Google Nexus 5 phones during data collection. Although we used four different watch models for data collection, the recording app implements the same sampling frequency even when running on different models.

TABLE 2
Summary of Data Collection in Our User Study

# of participants	22	Time of non-workout	$14.88h$
# of weightlifting types	15	Time of walking	$9.22h$
# of weightlifting sets	2528	Time of running	$7.95h$
# of weightlifting reps	24408	Time of weightlifting	$36.15h$
Total duration: $68.20h$			



SUN	MON	TUE	WED	THU	FRI	SAT
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

01 Dec 2015
16:30:21: WALKING FOR 15.0MIN.
16:50:00: RUNNING FOR 30.0MIN.
17:30:08: WALKING FOR 10.0MIN.
17:45:00: WL_BICEP_CURL FOR 10REPS.
17:50:00: WL_BICEP_CURL FOR 10REPS.
17:55:38: WL_BUTTER_FLY FOR 15REPS.
18:00:55: WL_BUTTER_FLY FOR 10REPS.
18:08:12: WL_CHEST_PRESS FOR 8REPS.
Cardio total: 55.0 min.
Weightlifting total:
5 sets, 53 reps.

Fig. 7. Screenshots of the MiLift Android app. Left: a calender for workout management. Right: a workout activity summary of a given date.

User study. Our data collection campaign was performed as part of a user study that involves 22 participants, approved by UCLA IRB. Participants are aged from 19 to 27 and consist of both males and females. All participants regularly engage in gym exercises. We have collected 68.2 hours of workout data in total, including 2528 weightlifting sets (24,408 reps), as summarized in Table 2. For ground truth recording, we used the Moves app [57] to log cardio activities and let participants manually record set/rep counts in weightlifting exercises. In addition, we asked each participant to finish a post-completion survey on workout tracking (Section 6.3).

Offline analysis. With the collected data, we first conducted an offline analysis to train classification models. The high-level activity classifier and the weightlifting type classifier were implemented using the scikit-learn library [47] in Python. The HMM smoothing of high-level activities and weightlifting classification algorithms including session detection and rep counting were implemented in Matlab.

Android implementation. We implemented MiLift as an Android app using Android 5.1.1 (API 22). MiLift contains two components: a watch app that performs workout tracking and a phone app that provides visualization and assistance for workout management. The watch app executes real-time classifications as a background service and does not require user inputs. Multiple open-source projects were used to port trained classification model to Android, including CRF++ [58] and jahmm [59]. Fig. 7 left shows the MiLift workout calendar where users can choose a particular day to view their progresses. Fig. 7 right displays workout activities performed in a given day showing durations for cardio workouts and rep counts for weightlifting exercises.

6 Evaluation

We first evaluate MiLift using a set of micro-benchmarks including accuracy (Section 6.1) and power (Section 6.2) profiling of classification models. We then use a macro-benchmark to analyze the effect of MiLift on required user tasks and smartwatch battery lives compared with previous approaches (Section 6.3).

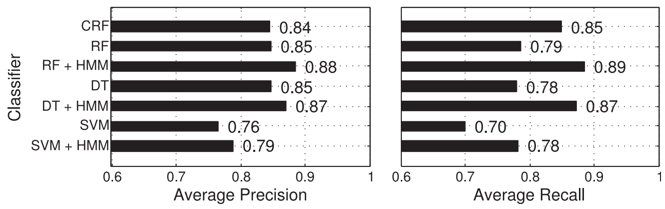


Fig. 8. Weighted average precision and recall of high-level activity classification (10-fold cross validation).

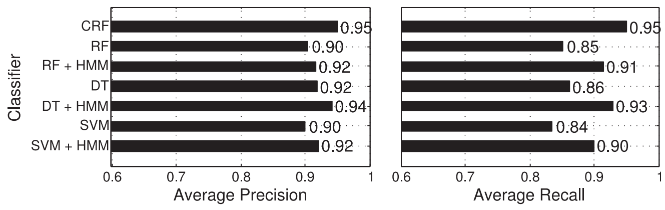


Fig. 9. Weighted average precision and recall of high-level activity classification (15-hour all-day trace).

6.1 MiLift Tracking Accuracy

6.1.1 *Accuracy of High-level Activity Classification*

The high-level activity classifier is the first state of the two-stage classification model and aims at detecting high-level activities of users before triggering the weightlifting classifier. To select the best model for this classification task, we first used a 10-fold cross validation with data from all users to examine the performance of the four classifiers, including the continuous graph model CRF and three instance classifiers RF+HMM, DT+HMM, and SVM+HMM. The best parameters selected for each classifier include maximum depth of 8 for DT, 64 classifiers and 4 features used at each split for RF, and linear kernel for SVM. For CRF, input feature values are first transformed to integers using cluster centers trained from k-means clustering algorithm, and our parameter tuning has chosen the use of 16 clusters for best CRF performance. Fig. 8 plots the weighted average precision and recall for all four models from the cross validation, including results before and after HMM smoothing for the three instance classifiers. Among the four models DT+HMM and RF+HMM has the best overall performance (nearly 90 percent) but CRF only performs slightly worse (around 85 percent). As discussed in Section 4.2, the results suggest that HMM smoothing is crucial in reducing false positives and increasing the overall performance of all three instance classifiers.

We then used the best model parameters selected from cross validation to showcase real-world performances of the high-level activity classifiers. We had one user collected an all-day data trace consisting of 15 hours of continuous accelerometer data. This trace includes the user walking around school, sitting during classes and study, running in a gym, and performing weightlifting exercises. Our entire dataset described in Section 5 is divided into two parts: all but this 15-hour data trace is used to train the high-level classifiers and this trace itself is used as the testing set. Fig. 9 plots the

TABLE 3
Memory Footprints of High-Level Classifiers

CRF	RF+HMM	DT+HMM	SVM+HMM
1.6 MB	105.1 MB	45 KB	5.6 MB

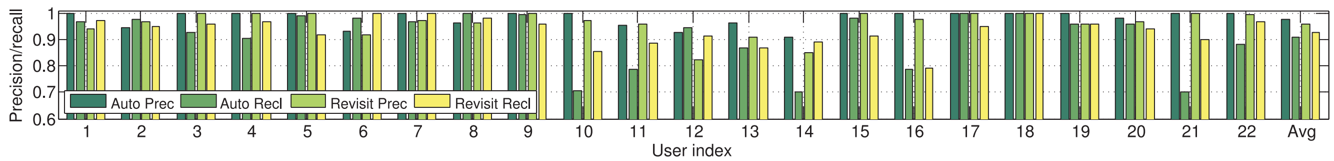
resulting weighted average precision and recall for classifications on the all-day trace. All four models achieve above 90 percent average precision and recall. Among them, DT +HMM and CRF have the best overall accuracy. The results above have proven that the high-level activity classification models in MiLift can successfully separate non-workout activities from gym exercises in real-world user scenarios. Moreover, continuous graphical models (e.g., CRF, HMM) can effectively increase the accuracy of time-series classification tasks by exploiting temporal correlations in data.

Finally, with random-access memory (RAM) becoming another power-hungry component in mobile SoC, even consuming more power than CPU in certain workloads [60], memory footprint is another factor to consider when choosing classification models. Table 3 compares the size of the four trained models. The sizes of DT+HMM and CRF are much smaller because of the simplicity of the model structures. Considering both their classification precision/recall performances and the memory footprints, we have chosen to implement both DT+HMM and CRF in MiLift.

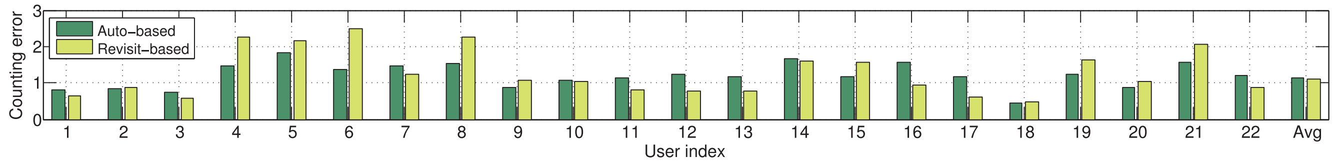
6.1.2 *Accuracy of Weightlifting Classification*

Session (set) detection and rep counting. We first evaluate the accuracy of weightlifting session detection and rep counting using our two proposed algorithms. Fig. 10a illustrates the weighted average precision and recall of session (set) detection based on users. In general, both algorithms demonstrate high overall accuracy, as autocorrelation-based approach achieves 97.5 percent precision and 90.7 percent recall while revisit-based approach yields 95.7 percent precision and 92.6 percent recall. Fig. 11 shows three common cases seen in our user study that lead to errors in weightlifting set detection: (a) some users tend to take short pauses within a set, possibly because of tiredness; (b) users can sometimes finish the last rep in an incorrect posture; (c) some users may adjust their body postures (including wrist orientations) during a set. All three cases can cause irregular patterns in sampled gravity data and set detection errors from both approaches.

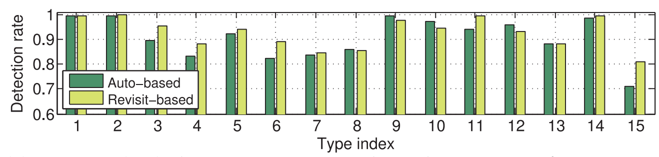
For rep counting, both the autocorrelation-based and the revisit-based approach have similar average errors, shown in Fig. 10b. Here the rep counting error is defined as the difference between the true number of reps and the number of reps counted by our algorithms in each set. On average the autocorrelation-based algorithm and the revisit-based algorithm report errors of 1.125 and 1.122 reps per set, respectively. Some rep counting errors are caused by cases shown in Fig. 11, because inaccurately detected session (set) boundaries can lead to rep miscounts as well. Other errors are results of unconventional postures and actions performed by some users during weightlifting exercises. Participants can sometimes incorrectly log ground truth leading to errors as well. Nevertheless, the rep counting errors are insignificant considering a user performs 9.65 reps on average within each set. In addition, the errors are only slightly greater than user expectations from our survey (Table 5).



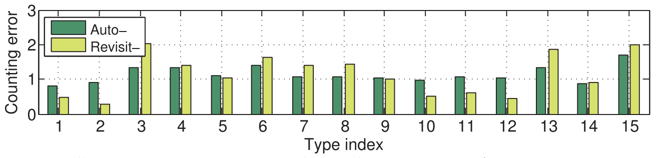
(a) Session (set) detection accuracy based on users.



(b) Rep counting error based on users.



(c) Session (set) detection accuracy based on types of exercises.



(d) Rep counting error based on types of exercises.

Fig. 10. Weightlifting detection performance, reported by users and by types of exercises (as shown in Fig. 1).

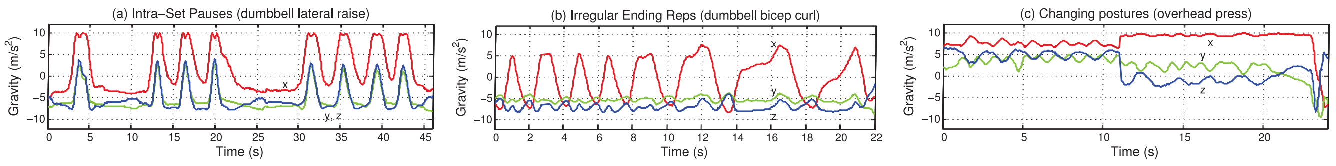


Fig. 11. Three common error sources of weightlifting session (set) detection seen in our user study.

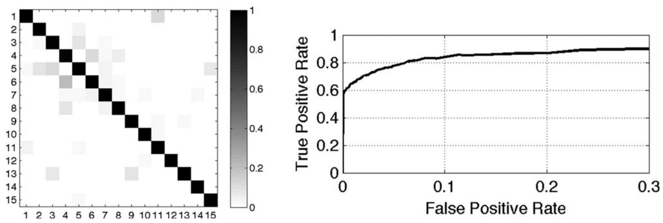


Fig. 12. Left: Confusion matrix of weightlifting type classification. Right: ROC curve of leave-one-type-out experiments.

To study the root cause of errors, we report session (set) detection and rep counting results based on the type of weightlifting exercises. Fig. 10c plots the recall or detection rate of each exercise type shown in Fig. 1. Note that the precision metric is not applicable here since a workout trace may contain multiple types of exercises and therefore we cannot identify false positive sessions for each exercise type. Both algorithms can nearly perfectly identify all sessions of bicep curl (#1), tricep extension (#2), tricep dip (#9), and dumbbell lateral raise (#14). However, seated row (#6), pec fly (#7), and rear deltoid (#8) are not well captured by either approach. This is because the vertical displacements in gravity data, i.e., the range between peaks and valleys, are small in all three axes for these exercises, making both algorithms susceptible to noises. Seated row does not involve substantial wrist motions and will not affect gravity readings much. Though both pec fly and rear deltoid require large forearm movements, the wrist trajectories of users fall into a horizontal plane and the watch rotates around z-axis in the global frame (perpendicular to the ground). Such rotation will not cause significant changes in gravity readings.

Although free weight exercises generally can lead to more complicated body movements than machine-based exercises, MiLift's performance on session detection and rep counting for free weight exercises are similar to those for machine-based exercises. For free weight exercises, MiLift has high errors detecting dumbbell bench press (#15) due to similar reasons stated above. It is also worth noting that keeping body motions stable while performing dumbbell bench press exercises is known to be difficult even for experienced gym users, which adds more noises to the sampled gravity data.

Fig. 10d reports rep counting errors by exercise types and shows similar error trends as seen in Fig. 10c: type #6-#8 and #15 have higher rep counting errors than others. Lat pull-down (#3) and dumbbell single arm row (#13) also have relatively high errors due to unclear repetition patterns shown in gravity data traces from some users.

TABLE 4
Average Power Consumption and Battery Life Benchmarks of Moto 360

State	Description	Power (mW)	Battery Life (h)
S0	Sleep	13.10	97.47
S1-D	High-level activity classification (DT+HMM)	47.08	25.83
S1-C	High-level activity classification (CRF)	58.83	20.67
S2-A	Weightlifting classification (autocorrelation)	159.58	7.62
S2-R	Weightlifting classification (revisit)	112.91	10.77

Showing battery life estimations if executing each state continuously.

TABLE 5
Survey Questions and Answers

Linear Scale Question (1 to 5)	**Score**
Do you find it useful that MiLift can automatically detect ongoing exercises (cardio and weightlifting)?	4.13 ± 0.83
Do you find it useful that MiLift can detect the type of exercises you are performing?	4.47 ± 0.74
Do you find it useful that MiLift can automatically detect and count weightlifting sessions (sets)?	4.67 ± 0.62
Do you find it useful that MiLift can count number of reps (repetitions) in weightlifting exercises?	4.73 ± 0.59
Short Answer Question	**# of Rep**
Max rep counting error you can tolerate.	0.8 ± 0.49

For scale questions, 1 (5) stands for strongly disagree (strongly agree).

Weightlifting Type Classification. We have chosen a Gaussian SVM with $r = 0.05$ for MiLift to determine the type of weightlifting exercises in a session and identify unseen exercise types. We used a 10-fold cross validation on the entire weightlifting dataset to test the performance of this classifier, which shows a precision of 89.71 percent, a recall of 89.53 percent, and an F1-score of 89.48 percent (all weighted average) for all 15 exercise type classes. Fig. 12 left plots the confusion matrix of all exercise types from cross validation, suggesting that our SVM model has good performance on all classes and does not bias towards certain weightlifting types.

Another important benchmark of weightlifting type classification is the accuracy of identifying newly unseen types. Section 4.3.5 proposes our algorithm in MiLift to identify new types by applying a threshold value $Thres_{conf}$ on confidence scores generated by the SVM. We evaluated the performance of this approach by conducting a leave-one-type-out cross validation. For each weightlifting type, we trained an SVM model using data from all other 14 types, and used this model to classify the entire dataset including instances of the 14 known types and the 1 unknown type. Because an instance of an unknown type can be similar to instances of one or more of the known types, the accuracy of identifying unknown types depends on the value of $Thres_{conf}$. A smaller $Thres_{conf}$ will allow more instances of unknown types to be correctly identified but will also lead to more false positives. Fig. 12 right plots the ROC curve (true positive rate over false positive rate) of identifying unknown types obtained by adjusting $Thres_{conf}$ from 0.1 to 0.8. The results illustrate our SVM model can achieve a true positive rate of 85 percent with about 10 percent false positive rate indicating an effective classifier in determining new types.

The micro-benchmark results show that MiLift can accurately track both cardio and weightlifting workouts (C3).

6.2 Energy Efficiency of MiLift Models

Table 4 compares the average power consumptions of MiLift at different classification states and the battery life estimations if each state is executed continuously on a Moto 360 smartwatch. We also measured the sleeping power of

TABLE 6
User Task and Watch Battery Life Comparisons of MiLift and Baseline Approaches

Approach	User Tasks	Battery Life
Baseline 1	(1) Manually start/stop weightlifting sessions; (2) Manually select workout types; (3) Count weightlifting sets and reps.	$18.14h$
Baseline 2	(1) Manually select workout types; (2) Count weightlifting sets and reps.	$6.28h$
Baseline 3	(1) Manually input the end of weightlifting sessions.	$2.32h$
MiLift	None	$21.45h$

the watch (S0) for comparison. There are two conclusions we can draw from the results:

First, the two high-level classification models DT+HMM and CRF are energy efficient after the 20 percent duty-cycle optimization discussed in Section 4.4. If only executing these two models, the watch battery can last for more than 16 hours meeting the efficiency challenge (C4). This is achieved even with the duty-cycle executions frequently waking up the watch, indicating that sampling and classification are more energy hungry for the watch than remaining awake. Out of the two models, DT+HMM is about 25.96 percent more energy efficient than CRF because of DT's lower classification complexity than CRF.

Second, in terms of watch battery life, our revisit-based weightlifting classification algorithm is 41.34 percent more energy efficient compared with the autocorrelation-based approach, due to less computation incurred by each new sensor sample ($O(1)$ compared with $O(wL)$ as discussed in Section 4.3). Note that our micro-benchmarks estimate watch battery lives assuming each state is executed continuously. During real-world executions, weightlifting classification S2-A or S2-R does not need to be executed continuously but will only be triggered by the high-level activity classifier S1-D or S1-C. Section 6.3 presents an empirical study showing how our two-stage model improves the battery life of smartwatches.

6.3 User Task and Battery Life Analysis

To evaluate MiLift in real-world scenarios, we conducted a macro-benchmark by empirically comparing MiLift with previous approaches using two metrics: (1) number of tasks a user needs to perform for accurate workout tracking, and (2) the battery life of a smartwatch running continuous tracking. We assumed the watch is used only during the day ($16h$), and the user performs weightlifting exercises for 12.5% of the time every day ($2h$). Each approach is described as follows:

Baseline app 1: To continuously track cardio workout activities, app 1 runs an always-on cardio activity classifier during the entire day. However, users have to manually start and stop weightlifting classification before and after each weightlifting session (12.5% of the day). Users need to manually select workout types and manually count sets/reps for weightlifting exercises as well. This app is similar to previous mobile workout tracking apps.

Baseline app 2: To reduce manual input from users, app 2 continuously executes both the cardio activity classifier and the weightlifting classifier all day. Users still have to manually select workout types and manually count sets/reps for weightlifting exercises. This app is similar to existing weightlifting tracking systems.

Baseline app 3: The VimoFit workout tracking app [18] executes continuously and can detect when a weightlifting session starts, but requires explicit user input to mark the end of workout sessions. The type of exercises and set/rep counts are automatically determined.

MiLift: MiLift automatically segments exercises from non-workout activities and does not require any manual input from users. It uses a two-stage classification model to duty-cycle the executions for efficient resource usage.

We used power profiles from Section 6.2 to estimate watch battery lives for baseline app 1, app 2, and MiLift. For app 3 (VimoFit) we performed a separate experiment to log its battery usage. For app 1, app 2 and MiLift, we used the more efficient DT+HMM approach as the high-level (cardio) activity classifier. App 1 and 2 implemented the more traditional autocorrelation-based algorithm for weightlifting detection, while MiLift considered the revisit-based algorithm.

User tasks: We first present the results of our user survey (Section 5) in Table 5. Users find the features of MiLift valuable, including automatic exercise segmentation, exercise type detection, weightlifting set detection, and weightlifting rep counting. This proves that tracking approaches without automatic exercise segmentation can be less attractive to active exercisers. Next, Table 6 compares the user tasks required in different approaches, suggesting that all three baseline apps rely on certain user tasks to accurately track exercises. Although baseline app 3 can automatically detect types of weightlifting exercises, the detection accuracy was poor. In our experiments, VimoFit was only able to correctly classify 3 of the 15 exercise types. In contrast, MiLift removes the burden on users and can provide fully autonomous workout tracking for both cardio and weightlifting exercises.

Energy efficiency: MiLift can extend watch battery life by 18.25 percent (3.31h), 241.56 percent (15.17h), and 824.57 percent (19.13h) compared with these three baseline apps, respectively. For baseline app 2 and app 3 the watch battery will run out before the end of the day forcing users to charge the watch. The VimoFit app used for baseline app 3 currently also keeps the watch screen on during executions therefore greatly exacerbating its power consumption. In contrast, a smartwatch running MiLift can last 21.45 hours without requiring charges that can interrupt exercise tracking. Such a battery life is sufficient for most use cases where users require the watch battery to last more than a day minus sleep time (16 hours). The improved battery life is achieved by MiLift's low-power weightlifting detection algorithm and the context-aware optimization.

The macro-benchmark suggests that MiLift can significantly better preserve battery power of smartwatches than previous approaches (C4) and remove user burdens (C2).

7 Discussion

We discuss potential improvements for future work:

Sensing scope and system generalization. Although MiLift can detect a variety of weightlifting exercises including both machine workouts and free weights, exercises which do not involve wrist motions cannot be tracked. This includes both single-arm exercises not performed on the watch-wearing hand and other types of exercises such as leg-based ones.

We argue that incorporating other non-intrusive sensors such as shoe motion sensors can cover more exercise types. Beyond cardio and weightlifting exercises, algorithms proposed in MiLift can be generalized to detect repeating patterns, for example, exercises and activities that involve repeating motions, including rock climbing, hiking, and racket sports. MiLift's algorithms can also work on clinical signals such as converting ECG signals to heart rate.

Energy optimization. We plan to leverage strategies proposed by prior work to further optimize the energy consumption of MiLift, such as offloading sensing and prepossessing from the main processor in mobile SoCs to low-power co-processors [61], [62], [63], exploiting the coordination of multiple mobile devices and the cloud [64], [65], and exploring the correlation among user contexts [66].

Quality assessment. Prior work has shown that incorrect uses of weightlifting equipment can lead to ineffective training or even injuries. Researchers have proposed several metrics to indicate exercise quality [16], [41]. Currently, MiLift does not provide exercise feedback, but some heuristics such as peak-to-peak intervals and noise patterns in the accelerometer/gravity data traces can be considered for quality assessment.

Data privacy. Using smartwatches as sensing devices for context inference apps can create new sources of privacy leakage. Personal workout history detected by MiLift is normally considered as sensitive information of a user. Without proper access control, such information can be exploited maliciously by third-party apps [67]. A framework to manage sensor access control across watches and phones is required to protect the privacy of users.

8 Conclusion

In MiLift the combination of a two-stage classification model and a lightweight weightlifting detection algorithm has enabled autonomous and efficient tracking of both cardio and weightlifting workouts. MiLift automatically segments exercises from non-workout activities so that users do not need to manually start/stop tracking or select exercise types. Our evaluations indicate that MiLift can achieve above 90 percent precision and recall for tracking both cardio workouts and weightlifting exercises. MiLift can also extend the battery life of a Moto 360 watch by up to 8.25 $\times$ (19.13h) compared with previous approaches. Finally, our user study suggests MiLift's automatic segmentation ability is valuable to individuals actively engaging in gym exercises.

Acknowledgments

The authors would like to thank Paul Martin, Tahiya Salam, and Ryan Peterman for their help on this work. This material is based in part upon work supported by the NSF under award #1029030 and #1213140, by the NIH under award #U54EB020404, and by the U.S. ARL and U.K. Ministry of Defense (MoD) under Agreement #W911NF-06-3-0001. This work was conducted while Chenguang Shen worked toward his Ph.D. at UCLA. Chenguang Shen and Bo-Jhang Ho contributed equally to this work.

References

[1] I.-M. Lee, et al., "Effect of physical inactivity on major non-communicable diseases worldwide: An analysis of burden of disease and life expectancy," *Lancet*, vol. 380, no. 9838, pp. 219–229, 2012.

[2] K. Patrick, W. G. Griswold, F. Raab, and S. S. Intille, "Health and the mobile phone," *Amer. J. Preventive Med.*, vol. 35, no. 2, pp. 177–181, 2008.

[3] M. A. Case, H. A. Burwick, K. G. Volpp, and M. S. Patel, "Accuracy of smartphone applications and wearable devices for tracking physical activity data," *J. Ame. Med. Assoc.*, vol. 313, no. 6, pp. 625–626, 2015.

[4] G. Castelnuovo, et al., "Obesity and outpatient rehabilitation using mobile technologies: The potential mhealth approach," *Frontiers Psychology*, vol. 5, p. 559, 2014.

[5] J. J. Rodrigues, I. M. Lopes, B. M. Silva, and I. d. L. Torre, "A new mobile ubiquitous computing application to control obesity: Sapofit," *Informat. Health Soc. Care*, vol. 38, no. 1, pp. 37–53, 2013.

[6] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava,"Using mobile phones to determine transportation modes," *ACM Trans. Sensor Netw.*, vol. 6, pp. 13:1–13:27, 2010.

[7] J. M. García-Gómez, I. de la Torre-Díez, J. Vicente, M. Robles, M. López-Coronado, and J. J. Rodrigues, "Analysis of mobile health applications for a broad spectrum of consumers: A user experience approach," *Health Informat. J.*, vol. 20, no. 1, pp. 74–84, 2014.

[8] Workout Labs. [Online]. Available: http://workoutlabs.com/, accessed on 2017.

[9] Gymwolf. [Online]. Available: http://www.gymwolf.com/, accessed on 2017.

[10] RunKeeper. [Online]. Available: https://runkeeper.com/, accessed on 2017.

[11] Strava. [Online]. Available: https://www.strava.com/, accessed on 2017.

[12] Apple HealthKit. [Online]. Available: https://developer.apple.com/healthkit/, accessed on 2017.

[13] Google Fit. [Online]. Available: https://developers.google.com/fit/, accessed on 2017.

[14] K.-H. Chang, M. Y. Chen, and J. Canny, "Tracking free-weight exercises," in *Proc. 9th Int. Conf. Ubiquitous Comput.*, 2007, pp. 19–37.

[15] B. J. Mortazavi, M. Pourhomayoun, G. Alsheikh, N. Alshurafa, S. I. Lee, and M. Sarrafzadeh, "Determining the single best axis for exercise repetition recognition and counting on smartwatches," in *Proc. 11th IEEE Conf. Wearable Implantable Body Sensor Netw.*, 2014, pp. 33–38.

[16] H. Ding, et al., "FEMO: A platform for free-weight exercise monitoring with RFIDs," in *Proc. 13th ACM Conf. Embedded Netw. Sensor Syst.*, 2015, pp. 141–154.

[17] Step tracking made easy with the M7. [Online]. Available: http://blog.runkeeper.com/326/step-tracking-made-easier-with-the-m7/, accessed on 2017.

[18] VimoFit. [Online]. Available: http://www.vimofit.com/, accessed on 2017.

[19] D. Morris, T. S. Saponas, A. Guillory, and I. Kelner, "RecoFit: Using a wearable sensor to find, recognize, and count repetitive exercises," in *Proc. 32nd ACM SIGCHI Conf. Human Factors Comput. Syst.*, 2014, pp. 3225–3234.

[20] Moto 360. [Online]. Available: https://www.android.com/wear/moto-360/, accessed on 2017.

[21] H.-T. Cheng, F.-T. Sun, M. Griss, P. Davis, J. Li, and D. You, "NuActiv: Recognizing unseen new activities using semantic attribute-based learning," in *Proc. ACM MobiSys*, 2013, pp. 361–374.

[22] F. Mokaya, R. Lucas, H. Y. Noh, and P. Zhang, "MyoVibe: Vibration based wearable muscle activation detection in high mobility exercises," in *Proc. ACM Int. Conf. Ubiquitous Comput.*, 2015, pp. 27–38.

[23] A. M. Khan, Y.-K. Lee, S. Y. Lee, and T.-S. Kim, "A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer," *IEEE Trans. Inform. Technol. Biomed.*, vol. 14, no. 5, pp. 1166–1172, Sep. 2010.

[24] S. Zhang, P. McCullagh, C. Nugent, and H. Zheng, "Activity monitoring using a smart phone's accelerometer with hierarchical classification," in *Proc. 6th IEEE Intell. Environ.*, 2010, pp. 158–163.

[25] J. Y. Xu, Y. Sun, Z. Wang, W. J. Kaiser, and G. J. Pottie, "Context guided and personalized activity classification system," in *Proc. 2nd Conf. ACM Wireless Health*, 2011, Art. no. 12.

[26] M. L. Pollock, et al., "Resistance exercise in individuals with and without cardiovascular disease benefits, rationale, safety, and prescription an advisory from the committee on exercise, rehabilitation, and prevention, council on clinical cardiology, american heart association," *Circulation*, vol. 101, no. 7, pp. 828–833, 2000.

[27] E. Cauza, et al., "The relative benefits of endurance and strength training on the metabolic factors and muscle function of people with type 2 diabetes mellitus," *Archives Phys. Med. Rehabilitation*, vol. 86, no. 8, pp. 1527–1533, 2005.

[28] S. T. McCaw and J. J. Friday, “A comparison of muscle activity between a free weight and machine bench press,” *J. Strength Conditioning Res.*, vol. 8, no. 4, pp. 259–264, 1994.
[29] M. L. Cotterman, L. A. Darby, and W. A. Skelly, “Comparison of muscle force production using the smith machine and free weights for bench press and squat exercises,” *J. Strength Conditioning Res.*, vol. 19, no. 1, pp. 169–176, 2005.
[30] I. Pernek, K. A. Hummel, and P. Kokol, “Exercise repetition detection for resistance training based on smartphones,” *Pers. Ubiquitous Comput.*, vol. 17, no. 4, pp. 771–782, 2013.
[31] C. Xu, et al., “Crowd++: Unsupervised speaker count with smartphones,” in *Proc. ACM Int. Conf. Ubiquitous Comput.*, 2013, pp. 43–52.
[32] S. Nirjon, et al., “Auditeur: A mobile-cloud service platform for acoustic event detection on smartphones,” in *Proc. 11th ACM Int. Conf. Mobile Syst. Appl Serv.*, 2013, pp. 403–416.
[33] H. Lu, et al., “Stresssense: Detecting stress in unconstrained acoustic environments using smartphones,” in *Proc. ACM Int. Conf. Ubiquitous Comput.*, 2012, pp. 351–360.
[34] R. LiKamWa, Y. Liu, N. D. Lane, and L. Zhong, “Moodscope: Building a mood sensor from smartphone usage patterns,” in *Proc. 11th ACM Int. Conf. Mobile Syst. Appl Serv.*, 2013, pp. 389–402.
[35] Apple Watch. [Online]. Available: https://www.apple.com/watch/, accessed on 2017.
[36] Microsoft Band. [Online]. Available: http://www.microsoft.com/microsoft-band/en-us, accessed on 2017.
[37] M. Shoaib, S. Bosch, H. Scholten, P. J. Havinga, and O. D. Incel, “Towards detection of bad habits by fusing smartphone and smartwatch sensors,” in *Proc. IEEE Int. Conf. Pervasive Comput. and Commun. Workshops*, 2015, pp. 591–596.
[38] N. D. Lane, P. Georgiev, C. Mascolo, and Y. Gao, “ZOE: A cloud-less dialog-enabled continuous sensing wearable exploiting heterogeneous computation,” in *Proc. 13th ACM Int. Conf. Mobile Syst. Appl Serv.*, 2015, pp. 273–286.
[39] MapMyRun.” [Online]. Available: http://www.mapmyrun.com/, accessed on 2017.
[40] Microsoft Health.” [Online]. Available: https://www.microsoft.com/microsoft-health/, accessed on 2017.
[41] C. Seeger, A. Buchmann, and K. Van Laerhoven, “myHealthAssistant: A phone-based body sensor network that captures the wearer’s exercises throughout the day,” in *Proc. 6th Int. Conf. Body Area Netw.*, 2011, pp. 1–7.
[42] B.-J. Ho, R. Liu, H.-Y. Tseng, and M. Srivastava, “Myobuddy: Detecting barbell weight using electromyogram sensors,” in *Proc. ACM 1st Workshop Digital Biomarkers*, 2017, pp. 27–32.
[43] Atlas Wristband. [Online]. Available: https://www.atlaswearables.com/, accessed on 2017.
[44] JEFIT Workout Tracker. [Online]. Available: https://www.jefit.com/, accessed on 2017.
[45] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, “Activity recognition and monitoring using multiple sensors on different body positions,” in *Proc. IEEE Int. Workshop Wearable Implantable Body Sensor Netw.*, 2006, pp. 113–116.
[46] S. Hemminki, P. Nurmi, and S. Tarkoma, “Accelerometer-based transportation mode detection on smartphones,” in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2013, Art. no. 13.
[47] F. Pedregosa, et al., “Scikit-learn: Machine learning in Python,” *J. Mach. Learning Res.*, vol. 12, pp. 2825–2830, 2011.
[48] C. Sutton and A. McCallum, “An introduction to conditional random fields for relational learning,” *Introduction Statist. Relational Learn.*, vol. 2, pp. 93–128, 2006.
[49] L. Yang, K. Ting, and M. B. Srivastava, “Inferring occupancy from opportunistically available sensor data,” in *Proc. IEEE Int. Conf. Pervasive Comput. Commun.*, 2014, pp. 60–68.
[50] S. Lee, et al., “Semi-markov conditional random fields for accelerometer-based activity recognition,” vol. 35, no. 2, pp. 226–241, 2011.
[51] D. L. Vail, M. M. Veloso, and J. D. Lafferty, “Conditional random fields for activity recognition,” in *Proc. ACM 6th Int. Joint Conf. Autonomous Agents Multiagent Syst.*, 2007, Art. no. 235.
[52] Composite sensor type summary - Android Developer. [Online]. Available: https://source.android.com/devices/sensors/sensor-types#composite_sensor_types, accessed on 2017.
[53] OptiTrack Prime 13. [Online]. Available: https://www.optitrack.com/products/prime-13/, accessed on 2017.
[54] Moto 360 Sport. [Online]. Available: https://www.android.com/wear/moto-360-sport/, accessed on 2017.
[55] LG G Watch R. [Online]. Available: http://www.lg.com/us/smart-watches/lg-W110-lg-watch-r, accessed on 2017.
[56] ASUS ZenWatch 2. [Online]. Available: https://www.asus.com/us/ZenWatch/ASUS_ZenWatch_2_WI501Q/, accessed on 2017.
[57] Moves app. [Online]. Available: https://www.moves-app.com/, accessed on 2017.
[58] CRF++. [Online]. Available: https://taku910.github.io/crfpp/, accessed on 2017.
[59] jahmm: An implementation of Hidden Markov Models in Java. [Online]. Available: https://code.google.com/p/jahmm/, accessed on 2017.
[60] A. Carroll and G. Heiser, “An analysis of power consumption in a smartphone,” in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2010, pp. 21–21.
[61] C. Shen, S. Chakraborty, K. R. Raghavan, H. Choi, and M. B. Srivastava, “Exploiting processor heterogeneity for energy efficient context inference on mobile phones,” in *Proc. ACM Workshop Power-Aware Comput. Syst.*, 2013, Art. no. 9.
[62] B. Priyantha, D. Lymberopoulos, and J. Liu, “Littlerock: Enabling energy-efficient continuous sensing on mobile phones,” *IEEE Pervasive Comput.*, vol. 10, no. 2, pp. 12–15, Apr.–Jun. 2011.
[63] P. Georgiev, N. D. Lane, K. K. Rachuri, and C. Mascolo, “DSP.Ear: Leveraging co-processor support for continuous audio sensing on smartphones,” in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, 2014, pp. 295–309.
[64] R. P. Singh, C. Shen, A. Phanishayee, A. Kansal, and R. Mahajan, “A case for ending monolithic apps for connected devices,” in *Proc. USENIX 15th Workshop Hot Topics Operating Syst.*, 2015, pp. 12–12.
[65] C. Shen, R. P. Singh, A. Phanishayee, A. Kansal, and R. Mahajan, “Beam: Ending monolithic applications for connected devices,” in *Proc. USENIX Conf. Annu. Techn. Conf.*, 2016, pp. 143–157.
[66] S. Nath, “ACE: Exploiting correlation for energy-efficient and continuous context sensing,” in *Proc. 10th ACM 10th Int. Conf. Mobile Syst. Appl. Serv.*, 2012, pp. 29–42.
[67] S. Chakraborty, C. Shen, K. R. Raghavan, Y. Shoukry, M. Millar, and M. Srivastava, “ipShield: Aframework for enforcing context-aware privacy,” in *Proc. 11th USENIX Conf. Netw. Syst. Des. Implementation*, 2014, pp. 143–156.

Chenguang Shen received the BEng degree in software engineering from Fudan University, Shanghai, China, and the PhD degree in computer science from the University of California, Los Angeles, in 2012 and 2016, respectively. He is a research scientist at Facebook. His research focuses on developing a mobile sensing framework for context awareness.

Bo-Jhang Ho is working toward the PhD degree in the Department of Computer Science at the University of California, Los Angeles. He is a research assistant in the Networked and Embedded System Lab supervised by Prof. Mani Strivastava. His research focus lies in the fields of pervasive and mobile sensing, but his interests also include machine learning algorithms and their applications. His work is published in ACM BuildSys, ACM UbiComp, and *IEEE Pervasive Computing*.

Mani Srivastava (F‘08) received the BTech degree from the Indian Institute of Technology Kanpur, Kanpur, India, and the MS and PhD degrees from the University of California at Berkeley, Berkeley, in 1985, 1987, and 1992, respectively. He was with Bell Laboratory Research, Murray Hill, NJ. He joined the University of California, Los Angeles, as a faculty member, in 1997, where he is currently a professor in the Electrical Engineering and Computer Science Department. His current research interests include wireless networking, embedded systems, sensor networks, mobile and ubiquitous computing, low-power and power-aware systems, and embedded technologies for health and sustainability. He is a fellow of the IEEE.