This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

FallDroid: An Automated Smart Phone based Fall Detection System using Multiple Kernel Learning

Ahsan Shahzad, *Student Member, IEEE,* and Kiseon Kim, *Senior Member, IEEE*

***Abstract*—Common fall occurrences in the elderly population pose dramatic challenges in public healthcare domain. Adoption of an efficient and yet highly reliable automatic fall detection system may not only mitigate the adverse effects of falls through immediate medical assistance, but also profoundly improve the functional ability and confidence level of elder people. This paper presents a pervasive fall detection system developed on smart phones (SPs) namely, FallDroid that exploits a two-step algorithm proposed to monitor and detect fall events using the embedded accelerometer signals. Comprising of the threshold based method (TBM) and multiple kernel learning support vector machine (MKL-SVM), the proposed algorithm uses novel techniques to effectively identify fall-like events (such as lying on a bed or sudden stop after running) and reduce false alarms. In addition to user convenience and low power consumption, experimental results reveal that the system detects falls with high accuracy (97.8% and 91.7%), sensitivity (99.5% and 95.8%), and specificity (95.2% and 88.0%) when placed around the waist and thigh, respectively. The system also achieves the lowest false alarm rate of 1 alarm per 59 hours of usage, which is best till date.**

***Index Terms*—Fall detection, smart phone-based application, multiple kernel learning, accelerometer, power consumption, Android App.**

I. Introduction

EXPONENTIAL increase in falls are recognized as a major factor causing physical, psychological, and economical concerns within the growing elderly population worldwide. Statistical reports from the World Health Organization (WHO) indicate that $28\%-35\%$ of seniors over 64 are subject to a fall event each year, which further elevates to $32\%-42\%$ for those over 70 years of age [1]. These falls are responsible for majorly 90% of hip and wrist fractures and 60% of head injuries [2]. Besides these injuries, long-lie situation (i.e. remaining on the ground for long time) is another outcome of fall that has serious consequences such as dehydration, hypothermia and even death. Moreover, frequent incidence of falls in the elderly may provoke fear of falling (FoF) which in turn, deteriorates their confidence in living independently and being socially active [3]. Thus, the development of automated, reliable, and prompt fall detection systems is vital to guarantee immediate assistance in case of falls, especially those involving long lies, and minimize severe health complications [4].

Contemporary techniques employed for automatic detection of imminent real-life falls can be broadly classified into two categories: (i) context-aware systems and (ii) wearable systems [5]. The former category concerns the deployment of sensory gadgets such as cameras, microphones, infrared, and pressure sensors to track the movement of people in limited environments. The main strength of these systems lies in usability amongst the elderly as no dedicated device is needed to be worn. Nonetheless, such systems are vulnerable to issues such as limited coverage, high installation cost, high false alarms due to other mobile entities, and privacy (especially in video-based systems). Fall detection methods based on wearable motion sensors that rely on kinematic signals, like tri-axial accelerometers and gyroscopes, fall under the latter category. While such body-worn systems offer several advantages over video-based systems, the bearer is still required to carry at least one device which may be intrusive and raise usability concerns. Moreover, the cost incurred by wearable commercial/customized fall detection devices such as LifeCall [6] and LPFD [7] is another issue of fundamental importance.

Momentum in the advancement of micro electro-mechanical systems (MEMS) technology has however, enabled the development of very light, compact, low power, and inexpensive wireless inertial sensors that eliminate concerns regarding their portability and user inconvenience [8]. Consequently, smart phones (SPs) integrated with dedicated detection hardware serve as potential candidates that have widely been accepted as a daily life commodity [9]. The remarkable penetration of smart phones (SPs) as a mobility and safety tool amongst the elderly has therefore, led the surge for cost-effective, efficient, and commercially viable fall detection applications [10].

Standalone body-worn or SP-based fall detectors have been extensively surveyed in [5], [11]–[13]. Earlier studies were predominantly grounded on threshold based methods (TBM) that rely on certain decision thresholds applied on features extracted from the inertial sensor signals [14], [15]. These detection algorithms are easy to implement, offer less computation cost, and are power efficient. The authors of [12] compared four famous TBM algorithms with a commercial device and showed that the TBM techniques fail to avoid false negatives (falls that remain undetected) and false positives (activities of daily living (ADL) classified as falls) simultaneously. Since fine-tuning the thresholds invokes a trade-off between number of false negatives and false positives, it is difficult to attain optimal thresholds ensuring performance consistency for everybody.

More recent studies reported in the literature use sophisticated machine learning (ML) techniques such as support vector machines (SVMs) [16], artificial neural networks (ANNs) [9], [17], and k-nearest neighbor (k-NN) [18] to classify falls from ADL. A comprehensive Matlab-based comparison of

This research was supported by the GIST Research Institute(GRI), and by the Brain Research Program through the National Research Foundation of Korea funded by the Ministry of Science and ICT 2016M3C7A1905477. Paper no. TII-17-0035. (Corresponding author: Kiseon Kim.)

The authors are with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju 500-712, South Korea (e-mail: ahsanshahzad@gist.ac.kr; kskim@gist.ac.kr)

1551-3203 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

the offline classification performance of six ML approaches is given in [19], wherein 1,404 features extracted from accelerometer, gyroscope and magnetometer sensing were used by the authors to obtain an average accuracy of around 99%. In spite of the profound results, usage of multiple sensors, especially gyroscope, along with the high computational cost of ML techniques exploiting large feature sets demands high power consumption thus, making them unfeasible for SPs.

With over 86% of the market share in the third quarter of 2016 [20] and due to its open source approach, Android OS stands out as the most widely used programming environment for SP-based fall detection solutions [21]. Though many SP-based fall detection algorithms have been reported, yet only few have actually been tested in real-life. Moreover, to our best knowledge, Smart Fall Detector (SFD) [17] and iFall [14] are the only two applications that have been released for public use. Other fall detection Android apps available on Google Play Store include Fade: Fall detector, Emergency Fall detector, and T3Lab [22]. Nonetheless, information on neither the underlying algorithm nor their performance exists.

Abbate *et al.* proposed an SP-based fall detector that uses a combination of TBM and ANN [9]. In spite of the reported 100% classification performance in offline analysis, the data set used for training and testing the ANN was very small (86 samples in total). Also, the performance of the application (false alarms rate and battery consumption) in real-life scenarios, commonly known as online analysis, was not presented.

More recently, Kerdegari *et al.* developed an Android application, SFD, built on the notion of multi-layer perceptron (MLP) neural network for fall event detection [17]. During offline analysis, their algorithm achieves 92.03% sensitivity, 91.07% specificity, and 91.06% accuracy on data recorded around the waist. However, when applied for online analysis, the system performance slightly degrades in terms of specificity (93.18% sensitivity, 88.88% specificity, and 91.25% accuracy). Additional to the long system decision time (at least 30 s for algorithm decision + 60 s default time for alert cancellation), no specific information on the false alarm rate with respect to the placement of the SP is provided. Furthermore, more battery power is drained due to the continuous use of ANN.

Successful deployment of a fall detection system among elderly population depends on various factors: usability, battery lifetime, privacy issues, cost, and reliability. System usability hugely depends on the number of sensors used, their prefix orientation, locations, battery life, and the like. The system should be reliable, i.e., it should detect almost all the fall events with 100% sensitivity while keeping the false alarm rate at its minimum. High false alarm rates can be very annoying and ultimately reduce user compliance with the system. To decrease false alarms, some researchers utilize postural information by either employing multiple sensors or by using a single tri-axial sensor with prefix orientation. As older people tend to easily forget, neither of the two options serve convenient in terms of usability. As the existing literature relating to these issues is lacking, there is no widely accepted system among elderly until now. Hence, it is of great significance to develop a reliable fall detection system to fill this gap while accounting for the existing practical issues.

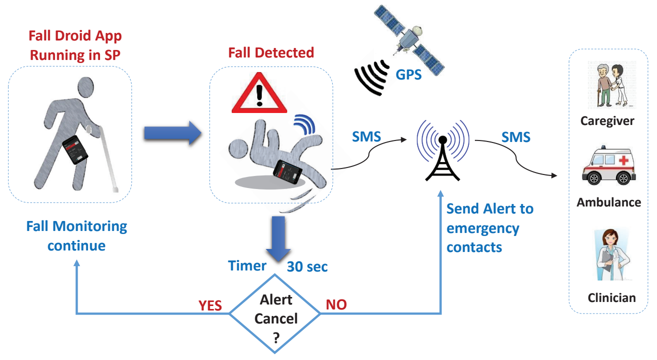

Fig. 1: Overview of fall detection and emergency alert system.

Accordingly, we present an automated high performance SP-based fall detection system focusing on practical issues such as user convenience and power consumption. The proposed standalone fall detector is developed as an Android app, namely FallDroid, which uses the accelerometer sensor embedded in SPs. The designed application provides an elder-friendly GUI and supports the two most convenient SP carrying locations: waist (belt/pouch) and thigh (pant pocket). In comparison with ML techniques, the proposed two-step algorithm is shown to be more power-efficient. In the first step, a low computational cost approach based on TBM is used, followed by the pattern recognition technique, multiple kernel learning support vector machine (MKL-SVM) in the second step which is rarely invoked. The battery consumption was analyzed and reported for different scenarios. The recorded data sets were acquired from human trials conducted systematically in both, laboratory and free living environments. Finally, we report the offline and online classification results on fall-like ADLs such as lying on the floor, sudden stop after walking, accidentally hitting the sensor etc. to demonstrate the better performance of the presented system.

The remainder of the paper is structured as follows: Section II presents the design, implementation, and algorithm used in the proposed system. The experimental settings and criteria for various scenarios are detailed in Section III, followed by results and discussions in Section IV and Section V, respectively. Finally, Section VI concludes the paper by pointing out the important observations and guidelines for potential future directions.

II. FallDroid System Design

FallDroid is designed as a standalone and user independent fall detection system that actively runs in the background and uses a two-step algorithm (described in sub-section II-B) to analyze subject movement. Upon fall detection, the application triggers the SP to vibrate and an alert cancellation page appears on the screen. Unless canceled within a specified time period (default setting of 30 s) by the user, a sound alarm is activated followed by a help text message containing location information being sent to specified emergency contacts. This process is illustrated in Fig. 1.

1551-3203 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

A. *FallDroid Implementation*

FallDroid has been developed using Android Studio IDE with min API 17 and target API level 23. The GUI of application consists of four screens: the main page, settings, fall alert cancellation, and feedback. The layout is designed to facilitate usability by the elderly with an overall focus on reducing battery power consumed particularly for unnecessary computations. In what follows, we highlight the main services provided by the application.

1) Configuration and Control: The main page of the application serves three functions: (i) start/stop the fall detection process, (ii) application configuration settings, and (iii) summarized display of critical/crucial settings. Once the user starts the sensing process, a notification icon will appear on the top left corner of the screen in the notification bar. The icon remains visible as long as the application is running in the background.The settings page can be used to customize personal details, location of the SP being carried, fall detection service priority, and settings related to fall alert notification. Settings for fall alert notification comprise of alarm sound and duration, countdown timer for alert cancellation, and entries for emergency contacts. Changes made to any user-specified setting can be seen immediately under the settings summary section on the main page. The settings are saved using the `Shared Preferences` class which permits previous settings to persist over multiple sessions as well as after SP reboot.

2) Fall Detection Service: The algorithm proposed for fall detection is implemented as an Android service, using the `Intent Service` class, that can run continuously in the background irrespective of the application. The intent service runs the algorithm in its own separate worker thread without blocking the main UI thread which otherwise can make the application non-responsive. Once the service is activated, it instantly acquires the PARTIAL_WAKE_LOCK, which prevents the CPU from going into sleep mode when the phone is idle. Subsequently, the algorithm checks the device accelerometer and its sensing capabilities. Once found, it registers the `Sensor Event Listener` to receive motion data from the accelerometer at a desired sampling rate. The incoming sensor data values are stored in a linked-list queue following the First-In-First-Out (FIFO) discipline, and maintains data history of up to 6 s. As further explained in sub-section II-B, initial steps of the TBM algorithm use this data to detect fall like events. Upon detection, the rest of the algorithm is then executed. The parameters associated with pre-trained MKL-SVM are not hard-coded but instead provided via a file. In this manner, application personalization or update is facilitated by replacing the parameters file with a new one.

3) Notification System: When a fall event is detected by MKL-SVM, the SP starts vibrating followed by an automatic launch of the fall alert cancellation page. The user has a default period of 30 s to cancel the false alert in case an ADL event is mis-classified as fall. If the user cancels the alert, the application then requests for feedback through the feedback page. The entered data is stored in a file which can later be used for further analysis to improve the algorithm performance. However, in case of a real fall event, the sound alarm is triggered once the alert countdown timer expires. The system then retrieves the last known geographical position of the user based on the available location providers (like GPS, Network) and sends the most accurate result in the form of an alert text message to the pre-specified emergency contact number. If the last known location was not too accurate (>100 meters) or not recent (>10 min), the application then tries to acquire the latest/current location prior to sending the alert message.

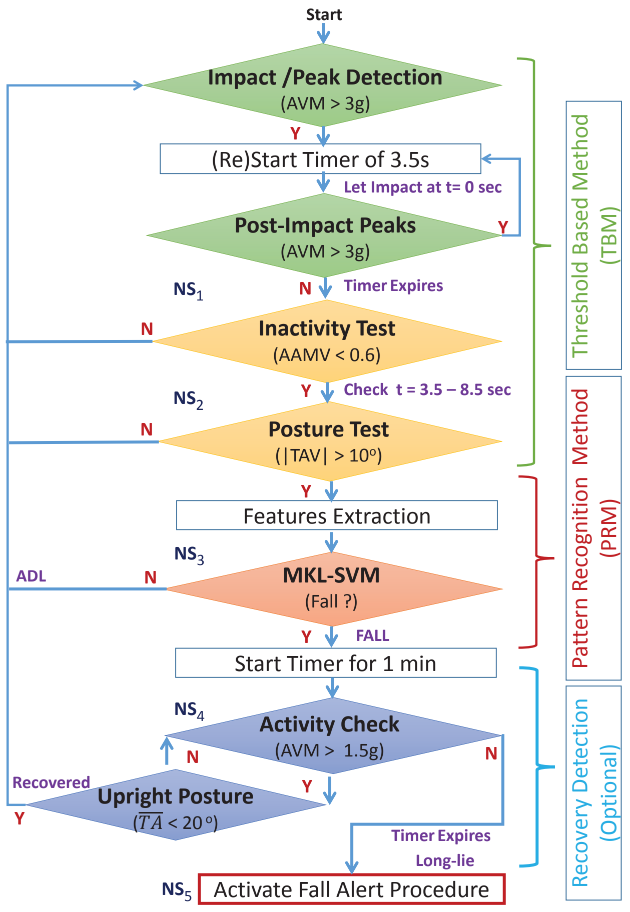

Fig. 2: Fall detection algorithm (proposed), comprising of two main steps: Threshold Based Method and Pattern Recognition Method, with an optional Recovery Detection part.

B. *Fall Detection Algorithm*

As shown in Fig. 2, FallDroid uses a two-step fall detection approach composed of TBM and MKL-SVM. The first step of the algorithm relies on TBM for determining fall-like events and efficiently discarding most fall-like ADLs using inactivity and posture tests. As falling down is a single incident that occurs instantaneously, a fall-like event does not exhibit the traits of repetitiveness and is basically characterized as an acceleration peak of magnitude greater than the 3 g threshold, followed by a period of 3.5 seconds without further peaks exceeding the threshold [1] (Fig. 3). The threshold value 3g is selected as it is widely reported in the literature [23] and is small enough to avoid false negatives, as even low impact

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

falls have peak acceleration greater than 3g value. When the user is static (i.e. no SP or sensor movement), the acceleration vector magnitude (AVM) shows a value of 1 g representing gravitational acceleration. However, during free fall, this value falls below 1 g and upon having impact with the ground, a high value (> 3 g) appears. The AVM signal used to detect fall-like events is:

$$AVM[i] = \sqrt{(A_x[i])^2 + (A_y[i])^2 + (A_z[i])^2}, \quad (1)$$

where A_x, A_y and A_z represent respectively, the acceleration signals along x, y, and z axes of the sensor. When a fall-like event is detected, the user is sensed for any activity over a default period of 5 s. The average absolute acceleration magnitude variation (AAMV) measure defined below is then compared with pre-defined threshold to analyse the inactivity event:

$$AAMV = \frac{1}{\#of samples} \sum_{i \in win} \left| AVM[i+1] - AVM[i] \right|. \quad (2)$$

In case of little or no movement after impact, (2) yields values close to zero ($AAMV < 0.6$). If the user was inactive, then his/her posture variation is evaluted using pre- and post-impact (peak) signals. If the user shows postural change, as tested by tilt angle variation ($TAV > 10°$), then it is most likely a fall. Hence, decisions on such fall-like events that are not discarded by TBM are made in the second step of the algorithm which exploits the powerful MKL-SVM pattern recognition technique, explained later on in this sub-section. Some examples of acceleration patterns generated by thigh located SP that cause false alarms are shown in Figure 4. An optional long-lie or recovery detection step can also be applied at the end of the algorithm to further alleviate the false alarms. If the user shows significant movement after the fall event and regains an upright posture within one minute, then it will be treated as recovery from fall and no fall alert will be generated. The extended test serves more efficient in discarding fall-like ADL events or to detect recovery from (non-injurious) fall as the user is more likely to show movement. On the other hand however, such prolonged tests would delay decision-making and medical assistance in case of an actual fall.

The thresholds are obtained from heuristic analysis using the training data set samples. Furthermore, the thresholds are chosen such that no actual fall event is mistakenly classified and discarded as ADL in the TBM stage. This ensures that the algorithm can achieve 100% sensitivity depending on the performance of MKL-SVM. In other words, by choosing higher threshold values for AAMV and lower values for TAV, more number of fall-like ADL events are likely to be passed to the computationally expensive MKL-SVM. However, owing to the fact that the number of fall-like events occurring daily are usually very few, the impact on power consumption will be insignificant. Fig. 5 shows the receiver operating characteristics (ROC) curves of AAMV and TAV measures. It can be seen that each of them can achieve 100% sensitivity while discarding almost 60% of the ADL events.

1) Feature Extraction: A set of 15 features listed in Table I were extracted from each simulated fall and fall-like ADL

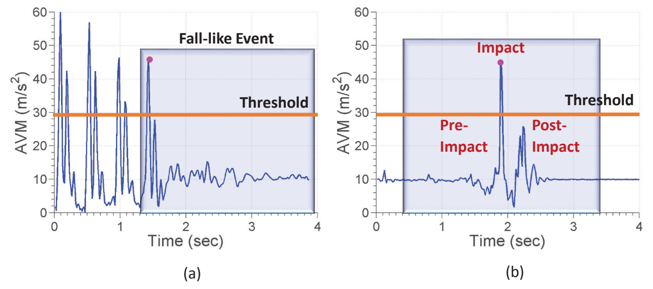

Fig. 3: (a) Fall-like ADL event generated by running and sudden stop activity. (b) Forward simulated fall pattern with a 3 sec window centered at impact point.

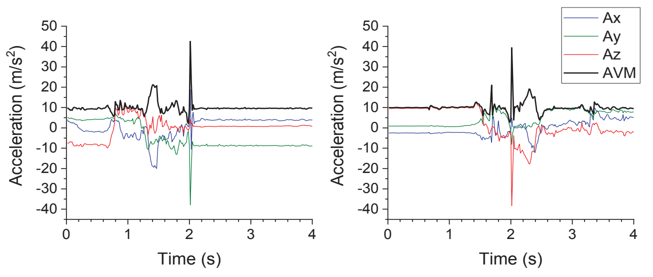

Fig. 4: Examples of thigh acceleration patterns during ADL that triggered false alarms on the proposed (TBM + MKL-SVM) algorithm.

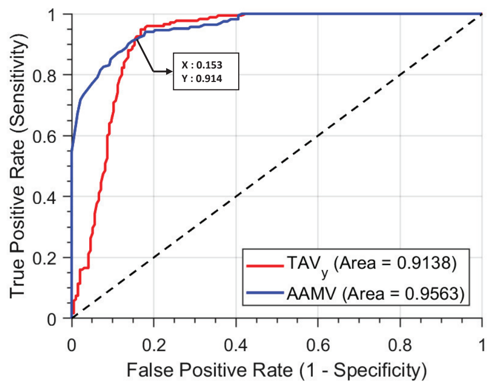

Fig. 5: ROC curves of AAMV (Inactivity test) and TAV$_y$ (Posture Test) features using thigh location dataset.

record. These features can be categorized as postural (13 − 15), energy-related (9 − 12), statistical (6 − 8), and reference-based (1 − 5) features [9]. The statistical and energy-related features were extracted from the AVM signal using a 3 s long data window centered at impact point (peak > 3 g) as depicted in Fig. 3. The duration of window was considered to cover pre- and post-impact signal patterns. The impact start and end points were found initially and were used to extract other features. More details on features 1 − 5, impact start (IS) and impact end (IE) points can be found in [9], while the remaining postural and energy-related features are described as follows:

- Tilt Angle (TA): This feature approximates the orientation of the SP along with the body of the user carrying it.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

TABLE I: Summary of the 15 candidate features derived from the accelerometer signals.

Feature No.	Feature Name
1	Avg. Absolute Accel. Magnitude Variation (AAMV)
2	Impact Duration Index (IDI)
3	Maximum Peak Index (MPI)
4	Minimum Valley Index (MVI)
5	Step Count Index (SCI)
6-7	Mean and Median
8	Range
9-11	Summed Axis Accel. along x, y, z axes (SAA$_{x/y/z}$)
12	Summed Magnitude Area (SMA)
13-15	Tilt Angle Variation (TAV$_{x/y/z}$)

Keeping in mind that gravity always acts vertical to the ground, the x and y axes of the SP sensor are mainly responsible for capturing the gravitational acceleration component (assuming standing position) for the waist (belt/pouch) and thigh pocket locations, respectively. As the posture changes from standing position to lying down, the TA value, calculated as below, goes from 0° to 90°:

$$TA[i] = \arccos \frac{A_{lp}[i]}{AVM_{lp}[i]}, \tag{3}$$

where $A_{lp}[i]$ and $AVM_{lp}[i]$ represent respectively, the low-pass filtered axis acceleration (x, y, or z axis) and the resultant acceleration.

- Tilt Angle Variation (TAV): The TA value may vary for the same posture depending on the alignment difference between the axis of the SP and the user vertical axis. To overcome this issue, we considered the change in TA during a fall-like event rather than using the TA value directly. Therefore, TAV is defined as the absolute value of the difference between the average TA values computed within intervals [IE + 0.5 s : IE + 1 s] and [IS − 1 s : IS − 0.5 s].
- Summed Axis Acceleration (SAA): This feature characterizes the intensity of activity along a particular axis and is defined as the total absolute acceleration along each axis.

$$SAA_{x/y/z} = \sum_{i \in win} \left| A_{x/y/z}[i] \right|. \tag{4}$$

- Summed Magnitude Area (SMA): The sum of the absolute values of the x, y and z-axes accelerations over the window length is represented by this feature and is computed as:

$$SMA = \sum_{i \in win} \left(\left|A_x[i]\right| + \left|A_y[i]\right| + \left|A_z[i]\right| \right). \tag{5}$$

2) Proposed MKL-SVM : To recognize the pattern of falls and ADL events, we used state-of-the-art multiple kernel learning (MKL) support vector machine. The MKL-SVM is an effective machine learning tool which allows the optimal combination of multiple kernels to build a single kernel support vector machine (SVM). The MKL problem involves learning both, the coefficients (a_i^* and b^*) and the kernel weights (d_m) in a single optimization problem. For kernel algorithms such as SVM, the decision function is of the (dual) form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} a_i^* \, K(\mathbf{x}, \mathbf{x_i}) + b^*, \tag{6}$$

where a_i^* and b^* denote the optimized support vector and bias coefficients that were learned using N labeled training instances $\{\mathbf{x_i}, y_i\}_{i=1}^{N}$, respectively. In our case, $\mathbf{x_i} \in \mathbb{R}^{(1 \times 15)}$ is the feature vector that represents the i^{th} event and the label $y_i \in \{-1, +1\}$ refers to ADL and fall classes. In the case of MKL-SVM, the kernel $K(\mathbf{x}, \mathbf{x_i})$ is basically a convex combination of basis kernels:

$$K(\mathbf{x}, \mathbf{x_i}) = \sum_{m=1}^{M} d_m K_m(\mathbf{x}, \mathbf{x_i}); \quad d_m \geqslant 0, \sum_{m=1}^{M} d_m = 1, \tag{7}$$

where M symbolizes the total number of kernels, K_m are the basis kernels for $m \in \{1, 2, \ldots, M\}$ and the constraint $d_m \geqslant 0, \sum_m d_m = 1$ tends to result in a sparse solution of d_m i.e. forcing some d_m to be zero. The widely-adopted kernels such as linear, 2^{nd} and 3^{rd} degree polynomials, and Gaussian with 5 different σ^2 values (0.1, 0.5, 1, 2, 3) were selected as candidates for basis kernels (M = 8). All kernels were build using the same set of input features and were normalized to unit trace before training. In this study, we considered the following formulation of (primal) MKL problem:

$$\begin{aligned}
&\min_{\mathbf{d}, \mathbf{w}, \boldsymbol{\xi}, b} && \frac{1}{2} \sum_{m=1}^{8} \frac{1}{d_m} ||\mathbf{w}_m||^2 + C \sum_{i=1}^{N} \xi_i \\
&\text{s.t.} && y_i \Big(\sum_{m=1}^{8} \big\langle \mathbf{w}_m, \phi_m(\mathbf{x_i}) \big\rangle + b \Big) \geqslant 1 - \xi_i, \quad \forall_i \\
& && \xi_i \geqslant 0, \quad \forall_i \\
& && \sum_{m=1}^{8} d_m = 1, \quad d_m \geqslant 0, \quad \forall_m,
\end{aligned} \tag{8}$$

where $\mathbf{w}_m$ is the m^{th} weight vector, b represents bias, ξ_i is the i^{th} slack variable, and hyperparameter C controls the regularization between training errors and an optimal separating hyperplane. The l_1-norm constraint on the kernel weights d_m will remove redundant kernels by forcing their weights (d_m) to be 0. The primal MKL problem (eq. 8) can also be transformed into a following min-max optimization problem which is based on standard SVM dual formulation (see [24] for details),

$$\begin{aligned}
&\min_{\mathbf{d}} \quad \max_{\boldsymbol{\alpha}} && \sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \sum_{m=1}^{8} d_m k_m(\mathbf{x}_i, \mathbf{x}_j) \\
&\text{s.t.} && \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& && 0 \leqslant \alpha_i \leqslant C, \quad \forall_i \\
& && \sum_{m=1}^{8} d_m = 1, \quad d_m \geq 0, \quad \forall_m.
\end{aligned} \tag{9}$$

Here $\{\alpha_i\}_{i=1}^{N}$ are the lagrange multipliers. The above mentioned min-max problem can be solved using a gradient

1551-3203 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

TABLE II: The types of simulated falls and fall-like ADLs events considered in our study.

Event Type	No.	Description
Fall events	1	(Forward) Improper weight shift, arms used for protection, end up lying facing downwards
	2	(Forward) Trip while walking forward, end up lying facing downwards
	3	(Forward) Walking with improper weight shift, rotate 90°, end up with right lateral position
	4	(Forward) Trip while walking and turning, rotate 180°during fall, end facing upwards
	5	(Backward) End up sitting with imaginary wall
	6	(Backward) Walking and slip, end up lying
	7	(Lateral) Left side, end up lying
	8	(Lateral) Right side, end up lying
	9	(Vertical) Collapse or loss of consciousness, syncope etc.
	10	(Experience) Fall as you experienced in real life
Fall-like ADL events	11	Sitting on (i) sofa (ii) chair (iii) floor
	12	Standing from (i) sofa (ii) chair (iii) floor
	13	Lying down on (i) bed (ii) floor
	14	Rising up from (i) bed (ii) floor
	15	Bending down to pick up an object from floor
	16	(Level walking) Bending to pick something, continue walking and make sudden stop
	17	Walking up/down the stairs
	18	(Running) Jogging, jumping
	19	(Sensor Hit) Hitting of sensor or SP with something unintentionally

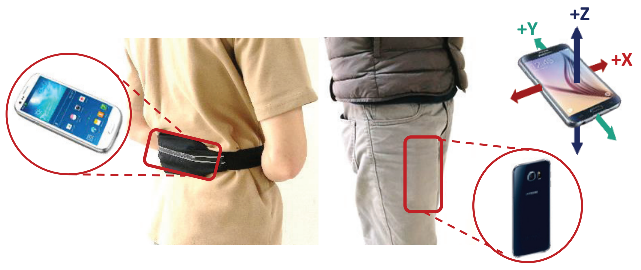

Fig. 6: Smartphone placed around the waist or at thigh location and the smartphone accelerometer axes.

descent based algorithm named *SimpleMKL* [24]. By keeping $\mathbf{d}$ fixed, the maximization problem over $\boldsymbol{\alpha}$ is solved via any standard SVM solver. After plugging the $\boldsymbol{\alpha}^*$ values, the remaining minimization problem over $\mathbf{d}$ is optimized using gradient descent method. These two steps keep iterating until the stopping criteria is reached i.e. either the duality gap (eq. 15 of [24]) reduced to less than 0.01 or a maximum number of 500 iterations is reached.

For model selection, 10 repetitions of 5-fold cross-validation (CV) was used to evaluate the MKL-SVM models with different mis-classification penalty hyper parameter C values. For each repetition of 5-fold CV, the data set was divided into 5 equal sized stratified partitions. Each MKL-SVM model (C) was tested for 50 times and the average performance was calculated. With increase in C, the classification performance (i.e. accuracy, sensitivity, specificity) kept improving until a certain point, beyond which it became constant. However, the number of selected kernels also increases with C resulting in increased computational cost. Therefore, we chose the minimum possible value of C that yields the best performance. Eventually, the parametric values set for the model deployed in SP are $C = 400$ for waist and $C = 250$ for thigh locations.

III. Experimental Design

Two user trials, according to III.A and III.B, respectively, were conducted to develop and evaluate FallDroid with the help of young volunteers. The first systematic trial (III.A) was carried out in a laboratory setting which included fall-like ADL events and simulated falls. This data was mainly used for model selection, classifier (MKL-SVM) training, threshold and feature selection in the fall detection algorithm. The second trial (III.B) involved a week long monitoring of the application in real-world scenarios so as to analyse the false alarm rate. Finally, the power consumption of the application was also evaluated in multiple ways as described in III.C.

A. Protocols for Fall-like ADL and Simulated Falls

Twenty volunteers (17 male and 3 female, age: 28.45 ± 2.72, weight: 66.15 ± 10.83, height: 170.7 ± 7.68) were asked to perform the scripted set of fall-like ADLs and simulated falls given in Table II as realistically as possible in our experiment. During the trials, the subjects carried two LG G2 smart phones: one placed in a pouch around the waist and the other at thigh position in the trouser pocket as in Fig. 6. The application was configured to sample at the rate of 64 Hz and save only fall-like events in a file. An informed consent was taken before the experiments and the protocols were approved by the ethics committee of our institute.

The volunteers performed four different types of falls common to elderly people: forward, backward, lateral, syncope or vertical down fall with different body dynamics. The experiments were designed based on the previous simulated falls reported in literature and the video recordings of real-world falls [2]. The fall events were performed over a soft mat to avoid potential injuries. The subjects were asked to maintain the final posture for at least 10 s after each fall event.

With regard to non-fall actions, we mainly focused on fall-like ADLs that can be easily mis-interpreted as falls [9] as listed in Table II. In addition, we also deliberately monitored the ADL of three subjects over a continuous period of five days

1551-3203 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

to cover other possible fall-like ADLs. As a result, additional fall-like events were also recorded (cycling, sitting in car) and included in our final data set. As a result, 209 simulated falls and 137 fall-like ADL events were recorded from around the waist, while 175 simulated falls and 196 fall-like ADL events were detected for the thigh location.

B. Protocols for Free-Living Trials

In this setting, four (new) young volunteers (3 male and 1 female, age: 29.5 ± 3.12, weight: 68 ± 11.6, height: 169.25 ± 3.4) were asked to carry one SP pouched around the waist and the other in the pant pocket. They were instructed to keep the phones with them during the day for an entire week and perform their everyday routine. The application was configured to record the number of times each step of the proposed algorithm executed. Additionally, each fall-like event, its corresponding inactivity and posture test values, and the fall alarms (if generated) during the monitoring period were also logged by the application. In absence of any real fall events, all recorded fall alarms are treated as false alarms.

C. Protocols for Power Consumption Analysis

A crucial factor in SP-based fall detectors is power consumption. The continuous use of multiple sensors and computationally complex functions substantially drain the battery power thus, making applications practically unfeasible to use. In general, SP battery consumption largely leans on various factors including the number of running applications, active radio services (Wifi etc.), and user activity. To have a better estimate of the power consumed by FallDroid, and to avoid any possible interruptions, the phone was reset to factory settings, all running applications were closed, and put to airplane mode during Scenarios I and II. The location services were however, kept active for all scenarios and phone was charged to its maximum capacity prior to every experiment. The battery state was monitored and logged via the Battery Log application. All the experiments were conducted using LG G3 with a 3000mAh battery, Invensense embedded acceleromter sensor, Quad-core 2.7 GHz Krait 450 CPU, Adreno 420 GPU, 3GB RAM and Android OS v4.4.2. The scenarios are distinguished as below:

- *Scenario I (Wake-lock and sensor usage):* In this scenario, the application is configured to sense the accelerometer data without processing it. The application holds the Partial-Wake-Lock to prevent the CPU from sleeping and the fall detection algorithm is not running.
- *Scenario II (Running algorithm):* The user carries the SP with only the proposed algorithm (complete FD App) running on it. Assuming that the SP is not used for any other purpose, this scenario includes the battery consumption of the algorithm.
- *Scenario III (SP usage in real-life):* This scenario realizes our everyday routine where the user carries the SP and uses it as in real-life. The fraction of battery drained by FallDroid is reported using the Android OS battery diagnostic tool.

TABLE III: Offline performance analysis and comparison with existing applications. The average response time (fall event to alert generation) of FallDroid App is 8.9 sec.

App.	**Location**	**Features used**	**Sensitivity (%)**	**Specificity (%)**	**Accuracy (%)**
FallDroid	Waist	15	99.52	95.19	97.81
	Thigh	15	95.83	88.01	91.70
SFD [17]	Waist	28	90.10	80.50	86.24
Ref. [9]	Waist	8	89.62	80.53	86.01

IV. Application Performance Evaluation

In this section, the performance of online and offline classification as well as the power consumption of FallDroid is investigated and compared with relevant existing works.

A. Classification Performance Analysis

- *Offline Analysis:* The data set acquired through trials detailed in sub-section III-A was used for offline performance analysis of proposed MKL-SVM classifier in Matlab. The final model ($C = 400$ for waist and $C = 250$ for thigh) was evaluated using 10 repetitions of 5-fold CV. At each repetition, the data set was divided into stratified partitions. The results for the performance measures such as sensitivity, specificity, and accuracy have been averaged over 50 iterations and reported in Table III. To perform a fair comparison, the algorithms presented in the related works were implemented and tested using the same data set. It is apparent that our proposed fall detection algorithm outperforms the existing works, yielding 99.52% sensitivity, 95.19% specificity, and 97.81% accuracy.
  In order to find the root-cause of high accuracy, i.e. whether it is the classification algorithm (MKL-SVM) or the proposed features set, we compared the performance of several other machine learning techniques with MKL-SVM using the proposed features set. The classifiers tested were SVM with single kernel (RBF, linear, polynomial), ANN, k-NN and Naïve Bayes. The best performing model for each classifier (parameters optimization) was selected using 5-fold CV in WEKA software. The averaged performance using 10 repetitions of 5-fold CV of each classification algorithm on waist and thigh location datasets is presented in Table IV and Table V. Unlike ANN models of [9] and [17] (in Table III), a similar ANN classifier achieved much higher performance (accuracy = 96.99%) on proposed features set. It can also be seen in Fig. 7 and Fig. 8 that the proposed features SAA and TAV provide good discrimination of ADL and fall events. By comparing the results in Table IVand V, we can conclude that the major performance enhancement is due to the proposed features set. Whereas, the MKL-SVM classifier further improved the overall accuracy while outperforming all the other classifiers tested here.
- *Online Analysis:* The performance of FallDroid was also analyzed in the everyday life scenario in terms of

1551-3203 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

TABLE IV: Performance comparison (offline) of different machine learning algorithms using waist location dataset.

Model	Sensitivity (%)	Specificity (%)	Accuracy (%)
MKL-SVM	99.52	95.19	**97.81**
SVM	99.20	93.96	97.14
ANN	98.71	94.39	96.99
k-NN	99.50	93.12	96.99
Naïve Bayes	99.45	93.80	97.20

TABLE V: Performance comparison (offline) of different machine learning algorithms using thigh location dataset.

Model	Sensitivity (%)	Specificity (%)	Accuracy (%)
MKL-SVM	95.83	88.01	**91.70**
SVM	95.72	85.65	90.40
ANN	92.68	86.62	89.49
k-NN	97.02	84.30	90.30
Naïve Bayes	95.42	82.67	88.68

false alarm rate. The total monitoring time (SP carried duration), number of times each step of the algorithm was used (NS$_i$, see Fig. 2), and the false alarms triggered during the monitoring period for each volunteer is summarized in Table VI. Interestingly, the average time interval between two consecutive false alarms was calculated to be 59 h (16 h) for waist (thigh) location.

B. System On-time and Power Consumption Estimation

To report the battery consumption, two trials of Scenarios I and II each were conducted. For Scenario II, FallDroid and SFD [17] were both tested. The averaged results for the battery drain of LG G3 smart phone over a time period of 24 hours are illustrated in Fig. 9. The add-on power consumption of FallDroid with respect to the baseline scenario (Scenario I) is observed to be lesser than that of SFD. FallDroid consumes almost 24% of the battery charge in 24.32 hours in Scenario II. The estimated battery lifetime was almost 86 hours (24.32 h usage plus 62 h expected residual usage time).

V. Discussion

The better performance of our fall detection algorithm is evident in both, offline and online analysis. In addition, the simple GUI of FallDroid and support for multiple carrying locations ultimately improves user convenience. Furthermore, to minimize power consumption, the algorithm has been designed to reduce the average computational cost. This is achieved by avoiding power-intensive signal processing procedures and by largely exploiting the simple TBM phase of the algorithm most of the time.

Used in the second phase of the algorithm, MKL-SVM achieves excellent classification performance as compared to [9] and [17], which are both built upon the concept of two layer feed forward artificial neural network. Performance of the approaches mentioned in the aforementioned references on our data set however, was not appealing. This is mainly because our data set consists of merely fall-like events, i.e., confusing signal patterns which look much like fall signals, providing a robust performance analysis.

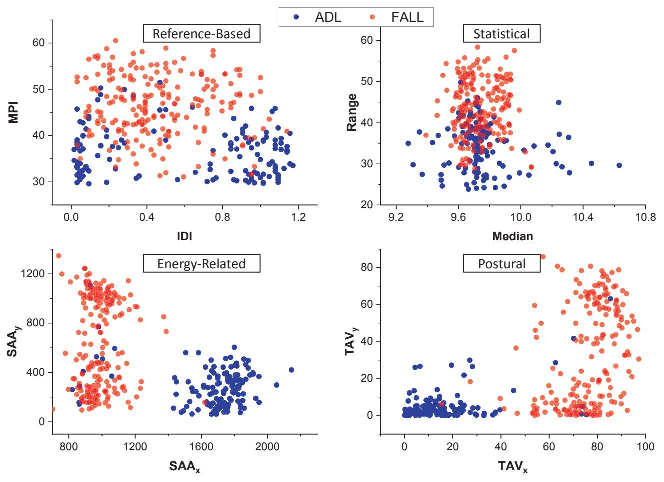

Fig. 7: Scatter plots of some of the reported features (Table I) using waist location dataset (section III-A) to compare ADL and fall events. Each plot title represents the category of two plotted features.

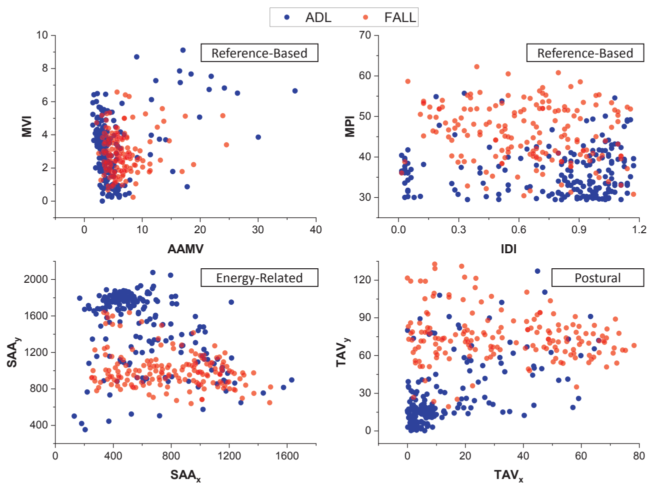

Fig. 8: Scatter plots of some of the reported features (Table I) using thigh location dataset (section III-A) to compare ADL and fall events. Each plot title represents the category of two plotted features.

Bagalá *et al.* [25] investigated and compared the performance of well-adopted fall detection algorithms based on waist or trunk accelerometer measurements. Their study showed that the number of false alarms (false positives) per day in real-life scenario ranged from 3 to 85, depending on the technique applied. Kerdegari *et al.* [17] evaluated the online performance of SFD using four volunteers monitored for a single day and reported a total of 5 false alarms per day. The results in Table VI reveal that our algorithm achieves the lowest false

1551-3203 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

TABLE VI: Online performance analysis; number of executions of each step (NS_i), total monitoring time (smartphone carried duration), and false alarm rate in free-living ADL trials.

SP Location	Subjects	Fall-like events or Inactivity Test (NS_1)	Posture Test (NS_2)	MKL-SVM (NS_3)	Recovery Detection (NS_4)	False Alarm Count (NS_5)	Total Monitoring Time (hours)	False Alarm Rate (FA's per hour)	Avg. duration between alarms (hours)
Waist	S1	4	4	1	0	**0**	36	0	–
	S2	2	2	2	0	**0**	39	0	–
	S3	0	0	0	0	**0**	37	0	–
	S4	83	58	15	4	**3**	65	0.046	21.67
Total		**89**	**64**	**18**	**4**	**3**	**177**	**0.017**	**59**
Thigh	S1	68	30	12	2	**0**	36	0	–
	S2	163	100	27	9	**6**	39	0.154	6.5
	S3	295	139	16	8	**5**	37	0.135	7.4
	S4	260	111	45	0	**0**	65	0	–
Total		**786**	**380**	**100**	**19**	**11**	**177**	**0.062**	**16.09**

alarm rate as compared to existing works [17], [25] (i.e., 1 alarm per about 59 h of usage at waist location).

With regard to the thigh position however, the performance was not as good as waist location. The SP is loosely attached to the user body inside the trouser pocket thus, generating added noise due to its own movement. Additionally, the number of fall-like events generated per day are much higher than the waist location and consequently, yielding a higher false alarm rate. The SP self/independent/free movements inside the pocket, frequent intense movements of thigh, and the difficulty in isolating sitting from lying by employing TA or TAV features are the main factors causing higher fall-like event counts. The false alarms rate of 1 alarm per 16 h of usage (at thigh location) reported herein is possibly of annoyance to users and their emergency contacts, but this rate decreases if the user cancels the alert proactively.

In contrast to many earlier efforts that were evaluated in laboratory environments using scripted simulated ADL movements, FallDroid was tested with free-living trials as well. This real life testing protocol is better than its scripted counterpart as the users acts naturally thus, covering their complete everyday actions including several unexpected events. Nonetheless, the application was assessed using young volunters. But since elderly people show reduced activities and have a more sedentary lifestyle, it is expected that the system would perform even better when used by elderly. The performance can further be enhanced through personalization of the App [26].

For any SP-based monitoring system, power consumption due to keeping the CPU awake (Partial-Wake-Lock) and sensors provides the baseline as represented in Scenario I. The added battery drain in Scenario II is due to the active fall detection algorithm. As visible in Fig. 9, the battery is majorly consumed for keeping the CPU awake and sensor usage. Since FallDroid invokes the computationally lower TBM unit most of the time, 86% of the fall-like events are rejected without involvement of the MKL-SVM classifier. The overhead battery consumption of our algorithm is very less with a reduction in system ON-time by 6 h.

The estimated system ON-time is calculated as the usage time on battery plus the estimated remaining time to use.

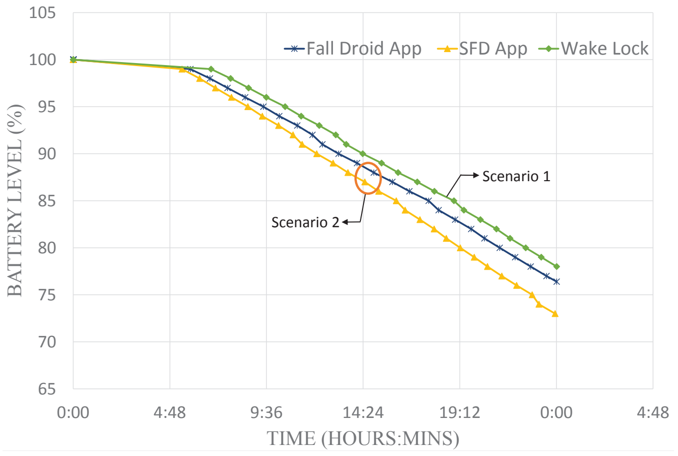

Fig. 9: Battery consumption comparison of FallDroid App (proposed) and SFD App with respect to reference consumption (Wake-Lock and sensor usage) as described in Scenarios II and I respectively of Section III-C.

With the CPU kept awake and samples collected by the accelerometer, the device ON-time was obtained to be 92 h. This however, reduced to 86 h (or 75 h) if we executed FallDroid (or SFD). These values were recorded only when the user did not use the SP for any other purpose (scenario II). In real-life, these values depend hugely on the SP usage frequency of the user. The FallDroid App drains 26.52% battery over a period of 25 hours 17 minutes during real-life usage of smartphone (Scenario III).

VI. Conclusions and Further Works

In this paper, we developed FallDroid, an Android-based fall detection and emergency alert application which supports multiple carrying locations (waist and thigh) for SPs and uses a simple user-friendly GUI for better accessibility. The proposed two-step fall detection algorithm achieves excellent classification results on fall-like events. The first stage adopts threshold-based method (TBM) to effectively discard most of the ADL data, whereas MKL-SVM is used in the second step

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

for classifying difficult fall-like events. Such structure helps reducing false alarms while maintaining low computation cost on average, resulting in lesser power consumption. The proposed algorithm was shown to outperform existing applications [9], [17] in terms of both, offline and online analysis.

As future work, the application can be tested in normal routine usage involving the elderly population. Such recorded data on the cloud storage will serve very useful for further refinement of the algorithm. Additionally, by using individual user data to train the MKL-SVM classifier, a personalized user-dependent fall detection system can be anticipated, enabling us to compare the reduction in false alarms. Lastly, in order to enhance the overall system performance, another potential future work could be on the feature engineering / extraction part to find even better features. Currently, the AAMV, TAV and SAA metrics seem quite good at fall detection, but perhaps higher order derivatives or other non-linear features comprised of complementary features could be considered especially in TBM part (first stage of algorithm) to increase specificity.

Supplementary Material

– Figure: The GUI of FallDroid Application.
– Figure: Scatter plots representing each feature distribution for waist location dataset.
– Figure: Scatter plots representing each feature distribution for thigh location dataset.

References

[1] C. Todd, D. Skelton, and W. H. Organization, *What are the main risk factors for falls amongst older people and what are the most effective interventions to prevent these falls?* World Health Organization, 2004.

[2] S. N. Robinovitch, F. Feldman, Y. Yang, R. Schonnop, P. M. Leung, T. Sarraf, J. Sims-Gould, and M. Loughin, "Video capture of the circumstances of falls in elderly people residing in long-term care: an observational study," *The Lancet*, vol. 381, no. 9860, pp. 47–54, 2013.

[3] S. R. Lord, C. Sherrington, H. B. Menz, and J. C. Close, *Falls in older people: risk factors and strategies for prevention*. Cambridge University Press, 2007.

[4] Y. S. Delahoz and M. A. Labrador, "Survey on fall detection and fall prevention using wearable and external sensors," *Sensors*, vol. 14, no. 10, pp. 19 806–19 842, 2014.

[5] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomedical engineering online*, vol. 12, no. 1, p. 66, 2013.

[6] "Lifecall fallalert system. available online: http://lifecall.com/ (accessed on 15 september 2016)."

[7] C. Wang, W. Lu, M. R. Narayanan, D. C. W. Chang, S. R. Lord, S. J. Redmond, and N. H. Lovell, "Low-power fall detector using triaxial accelerometry and barometric pressure sensing," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2302–2311, 2016.

[8] A. Shahzad, S. Ko, S. Lee, J.-A. Lee, and K. Kim, "Quantitative assessment of balance impairment for fall-risk estimation using wearable triaxial accelerometer," *IEEE Sensors Journal*, vol. 17, no. 20, pp. 6743–6751, 2017.

[9] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini, and A. Vecchio, "A smartphone-based fall detection system," *Pervasive and Mobile Computing*, vol. 8, no. 6, pp. 883–899, 2012.

[10] J. Zhou, P.-L. P. Rau, and G. Salvendy, "Older adults use of smart phones: an investigation of the factors influencing the acceptance of new functions," *Behaviour & Information Technology*, vol. 33, no. 6, pp. 552–560, 2014.

[11] M. A. Habib, M. S. Mohktar, S. B. Kamaruzzaman, K. S. Lim, T. M. Pin, and F. Ibrahim, "Smartphone-based solutions for fall detection and prevention: challenges and open issues," *Sensors*, vol. 14, no. 4, pp. 7181–7208, 2014.

[12] R. Luque, E. Casilari, M.-J. Morn, and G. Redondo, "Comparison and characterization of android-based fall detection systems," *Sensors*, vol. 14, no. 10, pp. 18 543–18 574, 2014.

[13] E. Casilari, R. Luque, and M.-J. Morn, "Analysis of android device-based solutions for fall detection," *Sensors*, vol. 15, no. 8, pp. 17 827–17 894, 2015.

[14] F. Sposaro and G. Tyson, "ifall: an android application for fall monitoring and response," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*. IEEE, pp. 6119–6122.

[15] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan, "Perfalld: A pervasive fall detection system using mobile phones," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, pp. 292–297.

[16] M. V. Albert, K. Kording, M. Herrmann, and A. Jayaraman, "Fall classification by machine learning using mobile phones," *PloS one*, vol. 7, no. 5, p. e36556, 2012.

[17] H. Kerdegari, S. Mokaram, K. Samsudin, and A. R. Ramli, "A pervasive neural network based fall detection system on smart phone," *Journal of Ambient Intelligence and Smart Environments*, vol. 7, no. 2, pp. 221–230, 2015.

[18] C. Medrano, R. Igual, I. Plaza, and M. Castro, "Detecting falls as novelties in acceleration patterns acquired with smartphones," *PloS one*, vol. 9, no. 4, p. e94811, 2014.

[19] A. T. zdemir and B. Barshan, "Detecting falls with wearable sensors using machine learning techniques," *Sensors*, vol. 14, no. 6, pp. 10 691–10 708, 2014, rEVIEW Paper, Comparing Machine Learning techniques(supervised) performance.

[20] "Idc worldwide mobile phone tracker. available online: http://www.idc.com/."

[21] M. Clark, J. Lim, G. Tewolde, and J. Kwon, "Affordable remote health monitoring system for the elderly using smart mobile device," *Sensors & Transducers*, vol. 184, no. 1, p. 77, 2015.

[22] "Google play store. available online: https://play.google.com/store (accessed on 15 june 2016)."

[23] A. Bourke, J. Obrien, and G. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," *Gait & posture*, vol. 26, no. 2, pp. 194–199, 2007.

[24] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "Simplemkl," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2491–2521, 2008.

[25] F. Bagal, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, "Evaluation of accelerometer-based fall detection algorithms on real-world falls," *PloS one*, vol. 7, no. 5, p. e37062, 2012.

[26] C. Medrano, I. Plaza, R. Igual, . Snchez, and M. Castro, "The effect of personalization on smartphone-based fall detectors," *Sensors*, vol. 16, no. 1, p. 117, 2016.

Ahsan Shahzad (S'14) received the B.Eng. degree in electrical engineering with honors from the University of Engineering and Technology (UET), Lahore, Pakistan, in 2009, and M.Eng. degree in electronics engineering from Gwangju Institute of Science and Technology (GIST), Gwangju, Korea, in 2014. He is currently a postgraduate student in the School of Electrical Engineering and Computer Science, GIST, Gwangju, South Korea.

His research interests include mhealth systems, biomedical signal processing and application design, gait analysis, and pattern recognition.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2839749, IEEE Transactions on Industrial Informatics

Kiseon Kim (M'84-SM'98) received the B.Eng. and M.Eng. degrees in electronics engineering from Seoul National University, Seoul, Korea, in 1978 and 1980, respectively, and the Ph.D. degree in electrical engineering systems from the University of Southern California, Los Angeles, CA, USA, in 1987. From 1988 to 1991, he was with Schlumberger, Houston, TX, USA. From 1991 to 1994, he was with the Superconducting Super Collider Lab, TX. In 1994, he joined Gwangju Institute of Science and Technology, Gwangju, Korea, where he is currently a Professor. He is the member of The National Academy of Engineering of Korea, the Fellow IET and the Senior Editor of IEEE Sensors Journal. His current interests include biomedical applications design, wideband digital communications system design, sensor network design, and analysis and implementation both at the physical layer and at the resource management layer.

1551-3203 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.