Personalized Fall Detection System

Anne H. Ngu
Department of Computer Science
Texas State University
San Marcos, TX, USA
angu@txstate.edu

Vangelis Metsis
Department of Computer Science
Texas State University
San Marcos, TX, USA
vmetsis@txstate.edu

Shaun Coyne
Department of Computer Science
Texas State University
San Marcos, TX, USA
spc51@txstate.edu

Brian Chung
Department of Computer Science
The Cooper Union
New York, NY, USA
bchung2017@gmail.com

Rachel Pai
Department of Computer Science
California State University
Long Beach, CA, USA
racheljpai@gmail.com

Joshua Chang
Dell Medical School
University of Texas
Austin, TX, USA
joshua.chang@austin.utexas.edu

***Abstract*—This paper explores the personalization of smartwatch-based fall detection models trained using a combination of deep neural networks with ensemble techniques. Deep neural networks face practical challenges when used for fall detection, which in general tend to have limited training samples and imbalanced datasets. Moreover, many motions generated by a wrist-worn watch can be mistaken for a fall. Obtaining a large amount of real-world labeled fall data is impossible as fall is a rare event. However, it is easy to collect a large number of non-fall data samples from users. In this paper, we aim to mitigate the scarcity of training data in fall detection by first training a generic deep learning ensemble model, optimized for high recall, and then enhancing the precision of the model, by collecting personalized false positive samples from individual users, via feedback from the SmartFall App. We performed real-world experiments with five volunteers and concluded that a personalized fall detection model significantly outperforms generic fall detection models, especially in terms of precision. We further validated the performance of personalization by using a new metric for evaluating the accuracy of the model via normalizing false positive rates with regard to the number of spikes of acceleration over time.**

I. Introduction

Wearable devices, especially smartwatches that pair with smartphones are increasingly a platform of choice for deploying digital health applications. This is due to the fact that a smartwatch has the benefit of being unobtrusive and comfortable to wear, as it can be seen as wearing a piece of accessory. The popularity of using a smartwatch paired with a smartphone as a viable platform for deploying digital health applications is further supported by the recent release of Apple Series 4 smartwatch which has a built-in "hard fall" detection application as well as an ECG monitoring App. Recently, an Android Wear-based commercial fall detection application called RightMinder was released on Google Play. The number of digital health applications using wearable devices is going to continue to increase in the next few years.

However, falls continue to be one of the leading cause of death and injury among the elderly [1]. According to the U.S. Center of Disease Control and Prevention, one in four Americans aged 65 and older falls each year. A recent CDC report [2] also stated that around 28% of people aged over 65 lived alone.

The prevalent fall detection monitoring application is the life alert system which is expensive (more than $1000 per year) and often not effective as it requires manual input by the users, who do not wear consistently due to inconvenience. Installing camera technology and using advanced computer vision to detect if a person has fallen is intrusive and is limited to certain areas in a facility. Contrary smartwatch technology is non-intrusive, ergonomic, and easily accessible to those who wear it. Moreover, smartwatch technology enables monitoring of falls anytime and in any place.

Deep Learning (DL) has demonstrated outstanding performance in computer vision, speech recognition and natural language processing applications. The work in [3] compared traditional machine learning (SVM, Naive Bayes) techniques with the Recurrent Neural Network (RNN), for fall detection using only acceleration data captured through a wrist-worn watch, and concluded that, even in the absence of large training datasets, DL shows superior fall detection performance compared to traditional statistical learning methods. Furthermore, neural network-based models, once trained offline, require fewer computational resources during real-time classification tasks compared to traditional machine learning methods which rely on feature extraction and processing. That makes DL models more suitable for low-power devices.

Nevertheless, DL networks generally require large training datasets to achieve optimal performance. In the real-time, smartwatch-based fall detection application domain, there are no publicly available, large, annotated datasets that can be used for training, due to the nature of the problem (i.e., a fall is not a common event). Training a deep neural network with a small dataset tends to produce overfit models. Our recent work in [4], demonstrated that the scarcity of data for training a fall detection model can be mitigated to a certain extend by combining DL with ensemble techniques such as stacking. The best performing ensemble DL model (4 stacked LSTM

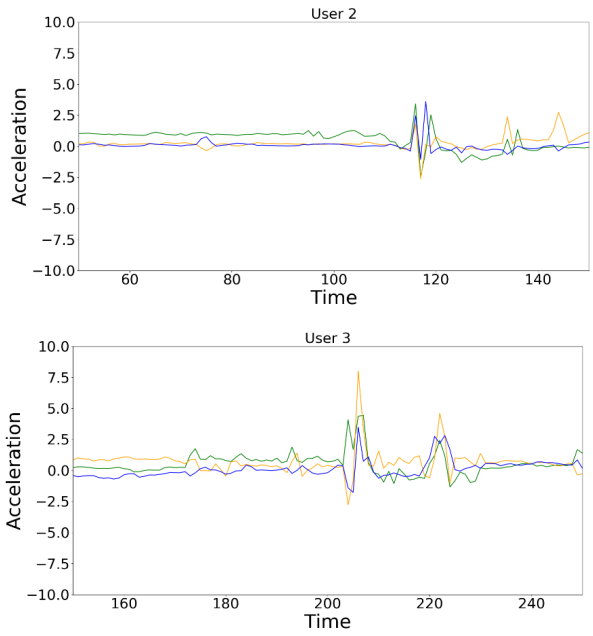

Fig. 1: Visualization of a front fall from two different subjects. The three lines in each graph are the measured acceleration values in three axes (x, y, z) over time (sec).

models) produced almost 100% recall, but exhibited 60% precision with the test data due to high false positive rates. When tested in the real-world by three subjects of different heights and weights, the model showed varying accuracy of fall detection from 95% to 70%.

Manual examination of the same type of simulated falls in Figure 1 from two different individuals with different physical, shows that one person's fall is unique from others. The high and low peaks from the two falls are considerably different. There is variation in the time dynamic within high and low peaks. The orientation of acceleration axes is different as well. This suggests that each person will have their unique fall patterns and wrist movements. We thus hypothesized that the low precision of our DL ensemble model can be mitigated via personalization.

Moreover, fall detection models which can only be trained on simulated falls and activities of daily living (ADLs) performed by healthy and young test subjects might not reflect the activity levels of elderly population. This further confirmed the need for personalization when the system is deployed.

In this paper, we aim to build a personalized fall detection method and demonstrate that it is possible to train an accurate personalized fall detection model via the collection of personal false positive samples feedback by end-users. The main contributions of the paper are:

- A methodology for personalizing fall detection model training.
- An experimental analysis on the performance of personalized model versus a generic model.
- A demonstration that a generic model trained using simulated fall data can be tailored to each person unique falling patterns via personalization.

The remainder of this paper is organized as follows. In section II, we review the existing work on fall detection and emphasize on research that specifically addresses fall detection using deep learning on smartwatches and existing personalization efforts. In section III, we introduce the SmartFall system architecture and UI that facilitates the collection of various feedbacks for personalization. In section IV, we provide a detailed description of our approach to personalization. In section V, we present various metrics including the spike of accelerometer for the evaluation of the personalized fall detection model. Then in section VI, we present our experimental results and discuss them. Finally, In section VII we present our conclusion and future work.

II. Background and Related Work

The early works in fall detection were concentrated on specially built hardware that a person could wear. Fall detection devices, in general, try to detect a change in body orientation from upright to lying that occurs immediately after a large negative acceleration to signal a fall. Those early wearable devices are not well accepted by elderly people because of their intrusiveness and limited mobility. However, modern smartphones and related devices now contain more sensors than ever before. Thus there is a dramatic increase in the research on smartphone-based fall detection and prevention in the last few years. This is highlighted in the survey paper [5]. The smartphone-based fall detection solutions, in general, collect accelerometer, gyroscope and magnetometer data for fall detection. Among the collected sensor data, the accelerometer is the most widely used. The collected sensor data were analyzed using two broad types of algorithms. The first are threshold-based algorithms which are less complex and require less computational power. The second are machine learning-based fall detection solutions. We will review both types of work below.

A threshold-based algorithm using a trunk mounted biaxial gyroscope sensor is described in [6]. The paper showed that by setting three thresholds that relate to the resultant angular velocity, angular acceleration, and change in trunk angle signals, a 100% specificity was obtained. However, there was no discussion on the practicality of attaching a trunk mounted sensor on a person for a prolonged period of time. There is also research work utilizing a personalized thresholding technique for fall detection on a waist-mounted device [7]. However, threshold-based methods do not fair well for wrist-worn devices due to the much higher variability of movements and ranges of acceleration.

Application of deep learning to fall detection, in particular the use of Recurrent Neural Networks (RNN's), to detect falls has been attempted by researchers; however, to our knowledge, no existing work uses solely accelerometer data collected by a smartwatch to detect falls. In [8], the authors describe an RNN-based architecture trained and evaluated on the URFD dataset [9], which contains accelerometer data taken from a sensor placed on the pelvis, and produces a

95.71% accuracy. The authors also describe a method to obtain additional training data by performing random rotations on the acceleration signal; training a model with this data gives an accuracy of 98.57%.

Another system based on RNN for fall detection using accelerometer data is proposed in [10]. The authors train and test their model with the SisFall dataset [11], which contains accelerometer data sampled at 200 Hz collected from a sensor attached to the belt buckle. In order to deal with a large imbalance in training data, of which ADL's form the vast majority, the authors define a weighted-cross entropy loss function, based on the frequency of each class in the dataset, that they use to train their model. In the end, their model attains a 97.16% accuracy on falls and a 94.14% accuracy on ADL's.

Detecting falls with accelerometer data from a smart watch was reported recently in [12]. The authors used a combination of Ensemble Stacked Auto-Encoders (ESAE) and unsupervised feature extraction, OCCCH (One-Class Classification based on the Convex Hull) for pattern recognition. They achieved an accuracy of 95.25% in sensitivity and 96.25% in specificity. Their approach of reducing the fall detection problem to a one-class classification problem since it is easy to collect lots of training data for ADLs is unique. However, there is no real-world validation of the experimental result. Testing the fall detection model on a specified set of falls and ADLs data is not a good indicator of how the model will perform in the real-world, as we show in the experimental section of this work.

Deep Learning for Human Activity Recognition (HAR) has demonstrated superior classification results on raw data, and eliminates the need for human crafted features [13]. However, DL networks face practical performance limitations when employed in fall detection: imbalanced dataset, small samples and data quality can significantly degrade the performance of the classifier. One way to overcome this real life limitations is to combine sets of diverse LSTM learners into an ensemble of classifiers as shown in [14]. In our earlier work [4], we drew our inspiration from them and experimented with Bagging, AdaBoosting, and Stacking ensemble DL models for fall detection. We demonstrated that Stacking-based ensemble DL model has the best recall and precision. However the model still suffered from false positives (FP) in controlled experiments. Our subsequent real-world experiments on the stacking ensemble DL model concluded that the variation of ADLs across different subjects cannot be exhaustively enumerated during simulated ADL data collection experiments, thus resulting in high false positive rates during real world testing. A personalized fall detection model that can adapt to each person's daily routines and activity levels is a better approach to resolve the high false positives problem.

In summary, our work is similar to many of the existing works in the sense that we utilized machine a learning model for finding fall patterns in time series data. We differ from them in the sense that our fall detection model obtains accelerometer data from an off-the-shelf smartwatch rather

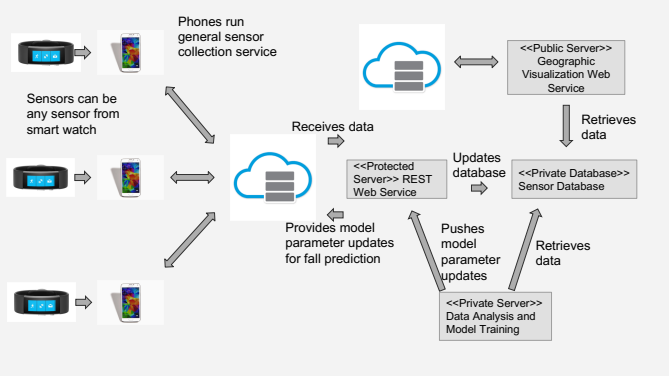

Fig. 2: Architecture of SmartFall system.

than specialized equipment placed on different parts of the body, or a smartphone. This presents several challenges not addressed in previous work. Because of its placement on the wrist, a smartwatch will naturally show more fluctuation in its measurements than a sensor placed on the pelvis or belt buckle. Moreover, the accelerometer data obtained by a specialized equipment is sampled at a much higher (approx. 200 Hz) frequency; this is a significantly higher than the frequency used by commodity off-the-shelf smartwatch, which samples at 30-50 Hz.

In the realm of personalization of fall detection using machine learning, the study by Tsinganos and Skodras [15], is closest to our work. Their acceleration data was sampled at 50 Hz compared to our at 31.2 Hz. Their data was collected using a smartphone while ours is via a smartwatch. They used a traditional K-Nearest Neighbor (KNN) machine learning algorithm while we used deep learning. To incorporate personalization, the authors added the misclassified ADLs (i.e. false positives) back into the training dataset one sample at a time and concluded that seven samples could reduce false positives by 10%. Similarly, we also collected the false positives and added them back to each training dataset. However, we collected and added the false positives data in batches. The size of the batch is dependent on how long the watch is being worn continuously and how many false positives are generated during that period of time.

Finally, existing commercial products, such as Apple Watch Series 4, have not published any information on the method or performance of their fall detection application. Apple's product description states that the smartwatch can detect "hard falls", however, they do not specify what constitutes a hard fall. If fact, during our simulated fall experiments, where healthy subjects fell on an air mat placed on the floor, Apple Smartwatch never any detected falls.

III. SmartFall Architecture and App

Figure 2 shows an overview of the software architecture of our SmartFall application that we deployed for our personalization experimentation. It is a three-layered architecture with the smartwatch on the edge and the smartphone in the middle layer which runs the application. Our system is structured such that the sensed data from the smartwatch can be stored

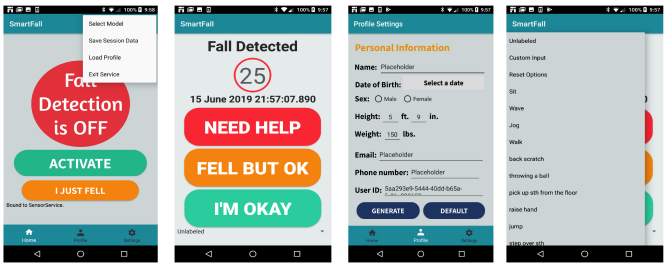

Fig. 3: SmartFall App User Interface.

locally on the smartphone to preserve privacy, and is in close proximity to the program that processes and analyzes the data in real-time. However, due to the limited storage capacity of the smartphone, we also provide the option of periodically removing the sensed data or transferring the sensor data (with consent from user) to a server securely for continuous refinement of the fall detection model and for the long term archival. The server, which is situated in the inner most layer also serves as the heavy-duty computational platform which consists of multiple services including a web server to host applications that can visualize aggregated sensor data for public health education, a sensor database for archiving and visualizing sensed data from the smartwatch of the user who has given the consent and machine learning services for analysis of the archived data for continuous refinement of the fall detection model.

Initially, the Microsoft Band 2 was chosen as the wrist worn device over other options in this prototyping phase due to the variety of sensors it supports and the low cost of acquiring the watch. Unfortunately, at the time of this writing, Microsoft has discontinued the support for Microsoft Band II in May 2019. Therefore, we had to port the system to run on a Huawei watch which is Android Wear compatible and will make our application available on more types of smartwatches in the future. Huawei watch has similar sampling rate of 32 ms as Microsoft Band II. But, the orientation of X, Y, Z accelerometer data sensed is different from Microsoft Band II. However, this can be easily fixed in the data collection codes. The Android-based Nexus 5X smartphone was chosen to run our SmartFall application and receive sensor data from the smartwatch via a low-power bluetooth communication protocol. This Nexus smartphone has a 1.8GHz hexa-core processor and 2 gigabytes of RAM memory. This proved sufficient for real time computation of the features, and for making the predictions, using models which were pre-trained offline.

A. Archiving Service and UI Design

We have implemented an archiving service which can be configured with a protocol where a participating user (with consent) can transmit sensed data in three-minute chunks to a designated server via a WiFi connection periodically. The UI design for SmartFall App also includes buttons specifically for labelling false positive, true positive and false negative data samples in real time from users. Those labelled archived data samples serve as additional data for re-training our Ensemble RNN model.

Figure 3 shows four views of the user interface (UI) of the SmartFall application. The screen on the left shows the home screen UI for the application and the second screen shows the UI when a fall is detected. We followed the best practices advocated in [16] for the design of the UI for the elderly. The three main principles we adopted were strict color scheme with high contrasts, legible and big fonts, simple description of the system to engage them to use it.

We will briefly highlight some of the key features of SmartFall app. The home screen (leftmost screen in Figure 3) launches the SmartFall App when the user presses the "ACTIVATE" button. The user must set up a profile and load the profile before the App can be activated. The "I JUST FELL" button on this screen is designed to collect and label false negative data samples, i.e. falls that were not detected by the App.

When a fall is detected, the second screen of Figure 3 pops up on the smartphone, an audible sound is generated, and a timer of 30 seconds is initiated. The user is shown three buttons for interaction. The "NEED HELP" button will send a text message to the caregiver and also save and label the sensed data samples as true positives. The "FELL BUT OK" button will save the sensed data during that prediction interval as true positives without notifying the caregiver. The "I'M OKAY" button will save these data as false positives. If a fall is detected and the user does not interact with any of these three buttons, after the timer expires, the system assumes that the user might be hurt or unconscious and an alert message is generated. The system can be configured to send the message to the caregiver automatically. The third UI screen is for the one time initialization of the user profile before the application can be launched. This UI includes setting up the contact details of the caregiver. Note that minimal personal data is collected and all those data are stored locally in the phone. The automatically generated user-id is used by the system to differentiate different user's data on the server. During data archiving, only this user-id and the selected sensed data such as the accelerometer data are sent to the server.

When saving the data samples as false positives (pressing the "I'M OKAY" button), the user also has an option to label the activity they were doing at the time the prompt appeared. This allows us to keep track of what activities are difficult for the model to recognize as ADLs. The rightmost UI screen is being launched to enable the user to tag that activity. If the activity is not already in the tagged list, user can select the "Custom input" menu item and enter a new activity which will be added to the list.

IV. Methodology

A. Experimental setup and design

The smartwatch dataset[1] used to create our generic model was collected using a Microsoft Band II smartwatch from 14

[1]This dataset is available from http://www.cs.txstate.edu/~hn12/data/SmartFallDataSet under the SmartFall folder.

volunteers of good health aged 21-60, height of 5ft to 6.5ft and weight from 100 lbs to 230 lbs. This dataset has a total of 528 falls and 6573 ADLs. A detailed description on how the data was collected is available in our earlier publication [3]. During the personalization experiments, we used Huawei watch running Android WearOS version 2.0. Note that Huawei watch has similar sampling rate of 32 ms as Microsoft Band II.

Our previous research on smartwatch-based fall detection system [4], showed that a stacked ensemble of deep LSTM models can perform better than a single LSTM model. This model has almost perfect recall, but has only around 60% precision. A 60% precision means that for every fall that was correctly detected in the test set, about 0.66 false positives were generated. Our SmartFall data contained a total of 528 falls, which means about 348 false positives were generated overall in our cross validation experiments. This number is not very meaningful as an evaluation criterion of how many false positives someone would get in real world use. That is because the ratio of falls that exist in dataset compared to the total size of the dataset, including the ADLs, is much higher than what one would get in real-world use.

The goal of the personalization experiments is to validate whether we can mitigate the false positive problem via user's feedbacks. The experiments were designed to minimize inconvenience to the user who participated in the experiments while still capturing as much information as possible about the user's activities that can be used to improve the model. We recruited five users for these experiments. Each user was told to pair the Huawei watch with the smartphone running our SmartFall application, wear the watch on their left wrist and perform two sessions of prescribed activities as well as wearing the watch for a few hours, while following their usual daily routines.

The prescribed activities matched those existing in the general training dataset and were used as a performance control baseline. The first set of the prescribed activities is a true positive data sample collection session. The user is to preform at least 20 falls on a mattress (five of each: back, front, left and right). This allows us to track the recall of the model as it becomes personalized. The second session is the false positive collection. The user is asked to perform a number of prescribed ADLs (e.g. sitting down, walking, brushing teeth, and hand waving). Table I shows the list of prescribed ADLs. These are common activities that an individual might perform in a day. A spreadsheet was given to each user to record the correctly detected (true positives) and missed falls (false negatives), as well as possible false alerts (false positives).

B. Personalization Strategy

The personalization strategy we adopted was a Training from Scratch (TFS) strategy. TFS aims to improve the model by re-training from scratch with additional false positive data samples collected from a specific user. We retrain a new model using the new dataset (feedbacks from the user) in addition to all of the original generic dataset. The decision to adopt the TFS approach versus a Transfer Learning (TL) approach,

TABLE I: SIMULATED ACTIVITIES OF DAILY LIVING (ADLS).

Activities Preformed	
Walking for at least 60 seconds	Sit down on a couch 3 times
Drink water	Change the channel on a TV
Eat food	Put on a jacket
Pick up 3 unique items from floor (wallet, backpack, piece of paper)	Brush Teeth
Walk down 5 flights of stairs	Get out of bed
Walk up 1 flight of stairs	Open many doors (usually 7)
Walk up hill	Tie shoes
Touch face	Put down a phone

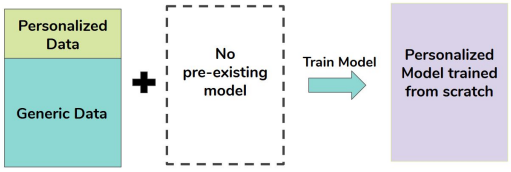

Fig. 4: Training from Scratch Strategy.

which utilizes an existing trained model and continues training with new data, was made because it exhibited higher accuracy in our experiments. Figure 4 is an illustration on TFS training strategy.

We achieve the personalization via two re-training sessions. In our first iteration, we used the data collected from the false positive collection session and all the data from the generic model and re-trained the model. We named this TFS round I model. We then deploy the personalized model (TFS round I) on the SmartFall App and asked the user to repeat the false positive session. During this second iteration, we also asked the user to perform frequently flagged false positive activities 10 more times. Using all the data used to train the TFS round I model, plus the data collected on frequently flagged false positives, we created a final personalized model using TFS. We named this TFS round II model.

Finally, to check that personalization is effective, we swap personalized models among the users and ask them to complete the false positive and true positive collection sessions a final time. In Section VI, we discuss and explain the experimental results from personalization.

V. EVALUATION METRICS

We aim to train a personalized fall detection model with a high recall or sensitivity. A missed fall is represented in our evaluation experiments as a False Negative (FN). We also do not want to have too many "false alarms", which in our evaluation is represented as False Positives (FP), and thus, we want to achieve a high precision and specificity. Table II shows the various metrics we used for evaluation and how they are computed.

These traditional evaluation metrics are used in the initial evaluation of our personalization model. However, the traditional method of cross validation by reserving a portion of training data for testing is not feasible when creating

TABLE II: EVALUATION METRICS. T=TRUE, F=FALSE, P=POSITIVE, N=NEGATIVE.

Measure	Calculation formula
Recall/Sensitivity	$TP/(TP+FN)$
Precision	$TP/(TP+FP)$
Specificity	$TN/(TN+FP)$
Accuracy	$(TP+TN)/(TP+TN+FP+FN)$

TABLE III: FALSE POSITIVES PER MODEL ON EACH USER.

Model name	**False Positives per User**					
	U1	U2	U3	U4	U5	Avg.
Generic	12	10	5	10	11	9.6
TFS round I	4	4	6	2	2	3.6
TFS round II	2	2	1	1	0	1.2

personalized models. One would either have to use a very small personalized testing set, or one would need to test using part of the generic set of data which will not be representative of improvements made due to personalization. Thus, it is necessary to use real world experiments to evaluate our models. We do this in two ways.

The first is the most obvious, we want to maintain a high recall while reducing false positives. For each model we ask the users to complete a set of prescribed activities. We then can measure recall by the number of falls detected as well as observe a change in the number of false positives generated. Furthermore, we also include a weighted F1-score (F_β), using the equation below, in order to combine precision and recall performance into one number.

$$F_\beta = (1+\beta^2) * \frac{recall * precision}{(recall + \beta^2 * precision)} \quad (1)$$

We set β to 3 for our calculations as this accurately reflects the higher emphasis we put on recall.

However, it is expected that activities that produce high acceleration values with the arm will generate more false positives than sedentary activities. To evaluate the number of false positives that occur for each user while controlling for different activity levels and life styles (e.g. younger, more active subjects vs. less active elderly), we used a new custom metric for measuring false positives. The new metric takes into consideration the number of acceleration "spikes" that occurred during a time period and uses those as normalization factor for the false positive rate. We consider a "spike" to occur when the magnitude of acceleration for a user exceeds the double of their average acceleration. The average acceleration is computed by processing all the acceleration data archived during the testing period. The total number of fall False Positives (FP) a particular model generates will be compared against the total number of spikes a user emitted to give our *Normalized Precision (NP)* value:

$$NP = (\#spikes - FP)/\#spikes \quad (2)$$

Spikes are a good normalization criterion to use in our case because there is a direct correlation between the user's activity level and the number of false positives.

VI. EXPERIMENTAL RESULTS

We now present our experimental results of personalization. We asked five users (all young and healthy adults) to test the TFS method of personalization. We asked the users to participate in the various types of false positive and true positive collection sessions described in Section IV. The "Generic" model uses the original model (stacked ensemble of four LSTMs). "TFS round I" is a re-trained model with specific user feedbacks from the false positive session using a prescribed set of ADLs as specified in Table I. "TFS round II" is a second iteration of re-training with additional false positive feedback on those activities (ADLs) that the model has difficulty in identifying. The "Swapped" is a model that was originally a "TFS round II" model meant for someone else. The "Personal" is a model that is targeted for a particular user.

Table III shows the number of false positives generated during the false positive session using a prescribed set of ADLs (Table I). Each user preforms the same activities the same number of times. This controlled data collection session takes 20 minutes to complete and provides some insight into the improvements made. The less false positives generated, the more precise is the model. TFS round I and II both see significant reductions in false positives generated. In addition to these scores, users were also asked to repeat the true positive session at various points to ensure recall was maintained. Table IV shows recall, precision and F1 scores of the various models. Note that users 4 and 5 were recruited at a later stage of the experimental evaluation, and thus we did not collect data for the for "Generic" and "TFS round I" models for users 4 and 5. We use "n/a" to indicate that in the table.

Overall, these results show that recall is being maintained while precision has significantly increased in TFS round II model. In Table V, we swapped the models for users, i.e. a model that was trained on the personalized data of one user, was evaluated on another user. The results show that "Swapped" model has lower precision of 0.71 as compared with the 0.87 for the "Personal" model .

To further validate the practically of personalized models for real world performance, we asked the users to wear the watch for several hours while they go about their daily lives. We asked that they remained moderately active during this time (e.g. not sleeping or sitting down the whole time). This allows us to get a spike score or normalized precision across users of different activity levels. On average, we collected around 8 hours of data from each user. We compared the spike-normalized result of each user using Generic with additional ADLs data from all users versus TFS round II model. Table VI shows the Normalized Precision of 0.98 with our TFS round II model versus the generic model with all users feedback of 0.94.

TABLE IV: RECALL AND PRECISION FOR EACH USER. MISSING DATA FOR USERS 4 AND 5 ARE MARKED AS "n/a".

Model		Scores per User					
		U1	U2	U3	U4	U5	Avg.
Generic	Recall	.85	.95	.90	n/a	n/a	.90
	Precision	.53	.63	.72	n/a	n/a	.63
	F1	.65	.76	.80	n/a	n/a	.74
TFS round I	Recall	.95	.95	.85	n/a	n/a	.92
	Precision	.79	.79	.65	n/a	n/a	.75
	F1	.86	.86	.74	n/a	n/a	.82
TFS round II	Recall	.90	.90	.95	.90	.95	.92
	Precision	.82	.82	.90	.86	.95	.87
	F1	.86	.86	0.93	.88	.95	.89

TABLE V: COMPARISON WITH SWAPPED MODELS.

TFS round II		Scores per User					
		U1	U2	U3	U4	U5	Avg.
Swapped	Recall	.80	.65	.85	.90	.95	.83
	Precision	.62	.52	.71	.82	.86	.71
	F1	.70	.58	.77	.86	.90	.76
Personal	Recall	.90	.90	.95	.90	.95	.92
	Precision	.82	.82	.90	.86	.95	.87
	F1	.86	.86	.93	.88	.95	.89

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have demonstrated that personalization can reduce the negative effects of small dataset size in fall detection applications. In particular, personalized feedback helps reduce false positives while maintaining recall in Recurrent Neural Network (RNN) models. A personalized fall detection model can be achieved by asking the user to wear the watch for a few hours and provide feedbacks when prompted. Most importantly, our personalization strategy does not require the users to perform any falls themselves. Users can carry on with their normal daily activities with minimal inconvenience while the fall detection model is personalized within a day.

The most significant drawback of the personalization strategy, when scaling up for multiple users, is the long training time involved in creating a new model, although such training can take place offline, on a server.

There are other methods of personalization yet to be explored, such as creating a group model that targets users with specific height, weight, gender, and associated physical or cognitive impairments. For example, for an elderly person who has Parkinson disease with postural instability symptoms is more likely to fall backward, while those with freezing gait symptoms may be more likely to fall forward. We expect that group-based personalization would require collecting a significantly larger dataset, and our current SmartFall App is a tool that can be utilized for crowd sourcing of such data with the right incentive.

TABLE VI: AVERAGE NORMALIZED PRECISION (NP) RESULTS, CALCULATED USING THE FORMULA PRESENTED IN EQUATION 2.

	Generic	Generic with additional ADLs	TFS round II
Hours worn	33.07	14.89	41.68
Number of F.P.'s	560	36	36
Number of spikes	1067	599	1677
Normalized Precision	.48	.94	0.98

ACKNOWLEDGEMENT

We thank the National Science Foundation for funding the research under the Research Experiences for Undergraduates site programs (CCF-1659807 and CNS-1757893) at Texas State University to perform this piece of work and the infrastructure provided by a NSF-CRI 1305302 award. We also thank various undergraduate students who volunteered to perform simulated falls and ADLs.

REFERENCES

[1] "Falls are the leading cause of death in older americans," https://www.cdc.gov/media/releases/2016/p0922-older-adult-falls.html, accessed: 2019-6-17.

[2] "2017 profile of older americans," https://acl.gov/sites/default/files/AgingandDisabilityinAmerica/2017OlderAmericansProfile.pdf, accessed: 2019-9-7.

[3] T. R. Mauldin, M. E. Canby, V. Metsis, A. H. H. Ngu, and C. C. Rivera, "Smartfall: A smartwatch-based fall detection system using deep learning," *Sensors*, vol. 18, no. 10, 2018.

[4] T. Mauldin, A. H. Ngu, V. Metsis, M. E. Canby, and J. Tesic, "Experimentation and analysis of ensemble deep learning in iot applications," *Open Journal of Internet Of Things (OJIOT)*, vol. 5, no. 1, pp. 133–149, 2019.

[5] 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.

[6] A. K. Bourke and G. M. Lyons, "A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor," *Medical Engineering and Physics*, vol. 30, no. 1, pp. 84–90, 2008.

[7] L. Ren, W. Shi, Zhifeng Yu, and Jie Cao, "Alarm: A novel fall detection algorithm based on personalized threshold," in *2015 17th International Conference on E-health Networking, Application Services (HealthCom)*, Oct 2015, pp. 410–415.

[8] T. Theodoridis, V. Solachidis, N. Vretos, and P. Daras, "Human fall detection from acceleration measurements using a recurrent neural network," in *Precision Medicine Powered by pHealth and Connected Health*. Springer, 2018, pp. 145–149.

[9] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," *Computer methods and programs in biomedicine*, vol. 117, no. 3, pp. 489–501, 2014.

[10] M. Musci, D. De Martini, N. Blago, T. Facchinetti, and M. Piastra, "Online fall detection using recurrent neural networks," *arXiv preprint arXiv:1804.04976*, 2018.

[11] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, "Sisfall: A fall and movement dataset," *Sensors*, vol. 17, no. 1, p. 198, 2017.

[12] L. Chen, R. Li, H. Zhang, L. Tian, and N. Chen, "Intelligent fall detection method based on accelerometer data from a wrist-worn smart watch," *Measurement*, vol. 140, pp. 215 – 226, 2019.

[13] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3 – 11, 2019, deep Learning for Pattern Recognition.

[14] Y. Guan and T. Plötz, "Ensembles of deep lstm learners for activity recognition using wearables," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 2, pp. 11:1–11:28, Jun. 2017.

[15] P. Tsinganos and A. Skodras, "A smartphone-based fall detection system for the elderly," in *Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis*, Sep. 2017, pp. 53–58.

[16] H. M. Salman, W. F. W. Ahmad, and S. Sulaiman, "Usability evaluation of the smartphone user interface in supporting elderly users from experts' perspective," *IEEE Access*, vol. 6, pp. 22 578–22 591, 2018.