Online Testing in Machine Learning Approach for Fall Detection

Lourdes Martínez-Villaseñor, Hiram Ponce, José Nuñez-Martínez, Sofía Pacheco
Universidad Panamericana. Facultad de Ingeniería.
Augusto Rodin 498, Ciudad de México, 03920, México
lmartine@up.edu.mx, hponce@up.edu.mx, 0169723@up.edu.mx, 0199375@up.edu.mx

***Abstract*—Robust fall detectors are needed to reduce the time in which a person can receive medical assistance, and mitigate negative effects when a fall occurs. Robustness in fall detection systems is difficult to achieve given that there are still many challenges regarding performance in real conditions. Fall detection systems based on smartphones present good results following a traditional methodology of collecting data, training and evaluating classification models using the same sensors and subjects, yet fail to experiment and succeed in different realistic conditions. In this paper, we propose a methodology to build a solution for fall detection, and online testing changing the sensors and subjects of evaluation in order to provide a more flexible and portable fall detector.**

***Index Terms*—Fall detection, machine learning, smartphone, edge computing, assisted living**

I. Introduction

Fall detection is important to help older people live an active independent life. Robust fall detection systems reduce the fear of falling and the time in which a person can receive medical assistance, mitigating negative effects when a fall occurs. Many diverse approaches for fall detection are reported in recent literature based mainly in wearable devices, ambient and vision sensors [1]. Even several commercial devices are available in the market for fall detection, but they are expensive and charge a monthly fee for their services [2] and/or have low performance on the field [3].

Robustness in fall detection systems is difficult to achieve given that there are still many challenges regarding performance in real conditions [4]. These challenges and limitations differ according to the selected sensors for data collection. Nevertheless, some issues are common regardless the types of sensors, for example the lack of reference framework for evaluation and the scarcity of public available datasets for validation [5]. Falls are infrequent in real life, it is difficult to collect data from real unexpected falls, and there is a great unbalance in the datasets [5]. Falls and activities often simulated by young adults, doesn't accurately represent elderly behaviors. Other design issues are computational cost and energy consumption [6]. The process of classification can be done in a server or in the data collection device. Using computational intensive classification algorithms makes some collection devices unsuitable for online training. Communication type of kinematic information of a subject and number of sensors are factors that impact energy consumption [6]. Regarding the challenges when the collection devices are wearable sensors or smartphones, placement and orientation of the sensors or devices have impact on the performance of fall detectors in addition to the above mentioned issues [7].

Fall detection systems based on smartphones reported in literature present good results following a traditional methodology: collecting data, training and evaluating classification models using the same sensors and subjects [4]. Very few of these works describe most of the steps in the complete process of fall detection: from dataset creation until the deployment of a model and real time evaluation trying to mimic realistic conditions [8] [2]. Hence, it is difficult to achieve robust fall detection systems with good performance in real life. Regarding smartphone technologies, the effort of following the traditional methodology from dataset creation, feature engineering, model training and testing in one type of smartphone does not guarantee a good performance in real life and/or other smartphones. It is desirable to build a flexible and portable solutions to deal with this issue.

In this paper, we propose a methodology to build a solution for fall detection, and online testing changing the sensors and subjects of evaluation in order to provide a more flexible and portable fall detector. Based on the creation of UP-Fall multimodal dataset presented in [9], we applied feature engineering, the analysis of sensor placement, the design of an application in a mobile device, and online testing of the system with different conditions. This online testing was performed in an iPhone XS smartphone. Four machine learning models were firstly evaluated offline, to say: random forests, support vector machines, multi-layer perceptron and k-nearest neighbors. The best model found was built using the UP-Fall Detection dataset, and then it was translated and deployed in the smartphone for online fall detection. Different subjects performed the activities and simulated falls for the online testing.

The main contribution of this work is an attempt of a novel online testing with the aim of achieving a robust, flexible and portable fall detection systems.

The rest of this paper is organized as follows. Related work is provided in Section II. In Section III, we describe our machine learning proposal for fall detection. Section IV presents the results of the online testing. Section V concludes our work.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

II. Related Work

In this section, we reviewed related fall detection approaches using wearable devices, inertial measurement unit (IMU) or smartphones. We included examples of reported methodologies that described most of the steps in the complete process of fall detection: from dataset creation until the deployment of a model and real time evaluation.

The related work of Frank et al. [8] presented the development of a system describing the complete process from the dataset creation to a naturalistic real-time evaluation. Twenty persons wearing an IMU in the belt performed sequences of activities of daily living (including falling) under semi-naturalistic conditions. Next, they performed feature design and selection followed by a Naïve Bayes activity classification. An evaluation with new participants was done in which some synchronization and classification difficulties were reported. Falls in particular presented misclassifications (59% precision, 44% recall). This approach is very complete given that they described their process starting with the dataset creation and performed a new set of real-time experiments, nevertheless, they only used one IMU position.

Ajerla et al. [2] proposed a framework using edge computing and wearable devices for real-time fall detection. The learning models were developed based on "MobiAct" dataset [10] and a long short-term memory (LSTM) network trained model was deployed in the edge computing framework. They performed different experiments to determine the waist as the best position for sensors. Code for execution of the pipeline is available in GitHub. It is important to note that the authors executed experiments in order to select the best sensor position, deployed their model in a framework and even shared some resources, hence they used a publicly available dataset. On the other hand,the creators of "MoviAct" dataset (using sensors embedded in a smartphone) [10] presented only a study to estimate the optimal feature set and an analysis of the pipeline for human activity recognition and fall detection. Their analysis included experimentation with random device orientation, but the same device placement in a pocket.

Hassan et al. [11] introduced a framework for fall detection in which real-time data were retrieved from an accelerometer sensor embedded on a smartphone. Data are processed and analyzed by an online fall detection system running on the smartphone itself. They used MobiAct public dataset to train the model offline.

Chen et al. [12] proposed FedHealth, a federated transfer learning framework for wearable healthcare. They addressed the aggregation of data from separate organizations and personalization problem. First they built a cloud model and next utilized transfer learning methods to achieve personalization. They suggest the deployment of FedHealth for elderly care and fall detection among others.

Another approach using smartphones for fall detection is described in [13]. The authors collected the dataset from ten participants simulating eight type of fall and activities of daily living recorded in real life. They focused their work in adaptability of novel detectors to different conditions. In a later work [14], Medrano et al. proposed a study to personalize smartphone-based fall detectors and designed experiments under two different training conditions for each subject. They did not report if the model was deployed for real-time evaluation.

Albert et al. [15] presented a fall detection and classification system based on smartphone. They described the data acquisition from 15 subjects that simulated falls wearing an accelerometer and a smartphone attached on belt on the back of the subjects. Nine subjects also wore the devices for ten days collecting everyday behavior. Hence, no falls were recorded in the second dataset and the collecting sensors were the same.

In summary, we observe that good results are reported in training and testing phases for fall detection, but very few works present realistic evaluation.

III. Machine Learning Approach

This section describes our methodology to build a solution for fall detection and to evaluate the performance using online testing on a mobile device. In such that case, our methodology consists of five steps: (1) the creation of a multimodal dataset, (2) the application of feature engineering, (3) the analysis of sensor placement, (4) the design of the application in a mobile device, and (5) the online testing of the system. Figure 1 shows our methodology. Details of each step are presented following.

A. Dataset Creation

The first step considers to create a dataset specifically for fall detection. Even though there exist several data sets in the literature, it is important that the information stored corresponds to the type of system that will be created. In that sense, sensor-based, vision-based or multimodal-based approaches should be taken into account.

For this purpose, we selected to use a multimodal dataset, namely UP-Fall Detection, from our previous work [9]. In a nutshell, this is a public large dataset comprising a set of 44 raw sensor signals and video recordings from two cameras with different points of view. The dataset consists of data information of non-overlapping simple human daily activities and falls. Seventeen healthy young subjects without any impairments (1.66 ± 0.05 m height and 66.8 ± 12.9 kg weight), 9 males and 8 females, ranging from 18 to 24 years old, performed 11 different actions (activities and falls), three trials each. Table I summarizes those actions, including a numeric label. Particularly, all falls were simulated by self-generation of the subjects, and all of them were collected in the same direction (right-to-left).

The dataset was collected with the following measurement devices: five wearable IMUs each with three-axis accelerometer, three-axis gyroscope and one ambient light sensors; one brainwave sensor; six pairs of infrared proximity sensors, in grid formation, to detect presence or absence of a person in the environment; and two cameras, one in lateral view and the other in front view related to the motion of the subjects. After synchronization, cleaning and pre-processing (following the

Fig. 1. Methodology for the development of a fall detection system using machine learning on a mobile device with online testing.

TABLE I
TYPES OF ACTIVITIES AND FALLS IN THE DATASET.

Type of action	Description	Class label
Fall	Forward using hands	1
	Forward using knees	2
	Backward	3
	Sideward	4
	Attempting to sit in an empty chair	5
Daily activity	Walking	6
	Standing	7
	Sitting	8
	Picking up an object	9
	Jumping	10
	Laying	11

TABLE II
MEASURE DEVICES AND PLACEMENT FOR DATASET CREATION.

Measure device	Placement
Wearable IMU 1	Left ankle (as bracelet)
Wearable IMU 2	Right pocket (of pants)
Wearable IMU 3	Waist (on the belt)
Wearable IMU 4	Neck (as collar)
Wearable IMU 5	Left wrist (as watch)
Helmet	Forehead
Infrared sensors 1–4	Along the length of environment
Infrared sensors 5–6	Along the width of environment
Camera 1	Lateral view (length side of environment)
Camera 2	Front view (width side of environment)

methodology described in [16]), the dataset comprised 296,364 samples of raw sensor and video frames of cameras, collected at 18 Hz. Table II details the placement of each measure device used in the dataset.

For a complete description of the UP-Fall Detection dataset, see [9]. It is publicly available at: `http://sites.google.com/up.edu.mx/har-up/`.

B. Feature Engineering

The second step is feature engineering. It is based on treating the raw signals from sensors and/or the video recordings from cameras to obtain meaningful information from the data. Several sub-steps were done: (i) windowing and feature extraction, (ii) binary labeling and (iii) feature selection.

1) Windowing and Feature Extraction: For convenience, we focused only in the wearable IMU sensor signals throughout this work. First, we segmented the raw data in small chunks or windows. Then, we calculated different features for each window.

We first experimented with three different windowing sizes: 1-second, 2-second and 3-second time lengths. Windowing was done with 50% of overlapping. Then, we constructed three different datasets, each one for each window size. For each window, in these different lengths, we extracted twelve features in time: mean, standard deviation, root mean square, maximal amplitude, minimal amplitude, median, number of zero-crossing, skewness, kurtosis, first quartile, third quartile and median auto-correlation. Also, we extracted six frequency features over the fast Fourier transform (FFT): mean, median, entropy, energy, principal frequency and spectral centroid. Since the UP-Fall Detection dataset also contains these feature sets, we used them for this work.

TABLE III
NUMBER OF SAMPLES IN EACH FEATURE DATASET CREATED.

Class label	1-sec dataset	2-sec dataset	3-sec dataset
1	172	86	51
2	175	85	50
3	209	109	66
4	169	85	52
5	230	116	79
Total *fall*	**955**	**481**	**298**
6	6,032	2,965	1,940
7	7,003	3,339	2,151
8	6,051	2,973	1,947
9	227	115	65
10	2,957	1,428	918
11	8,892	4,210	2,750
Unknown	177	81	55
Total *no-fall*	**31,339**	**15,111**	**9,826**

2) Binary Labeling: Then, a binary labeling was done to consider one two classes for fall detection: *fall* and *no-fall*. In this regard, we tagged actions from 1 to 5 as *fall*, and the remaining actions from 6 to 11 (daily activities) as *no-fall*. Unknown actions were automatically tagged as *no-fall*. Notice that falls actually have three actions in the entire activity: first subjects are standing up (class-7 or *no-fall*), then they start falling until reaching the ground (class 1–5 or *fall*), lastly they remain laying down (class-11 or *no-fall*) for a moment.

After that, we implemented a voting strategy [9] to determine the most probable action (*fall* or *no-fall*) within each window. Table III summarizes the number of samples per label in each of the feature sets created.

3) Feature Selection: Since we are interested on save computational resources, we implemented a feature selection procedure to determine the most useful features. To do so, we used a proposed method by Witten and Frank [17] in which the selection of features are done by combining subsets of attributes and evaluating them in a classifier, and then ranking the most powerful attributes found in each subset. In this regard, the evaluation of subsets were implemented with a scheme-independent technique, while the three ranking methods evaluated attribute correlation, relief and classification, methodology reported in [9]. It is remarkable to say that feature selection was implemented together with the sensor

placement analysis, as discussed below.

C. Sensor Placement Analysis

In this work, we investigated the best wearable sensor placement for the fall detection system using the UP-Fall Detection dataset. For this analysis, we considered only the five IMU sensors (see Table II). In that sense, we created five feature data sets, one for each IMU sensor, and then we built four supervised machine learning classifiers for each measure device. Those methods were selected based on the literature [5], [9], [13], [18], and they are the following:

- *Random Forest (RF)*. This is one of the most used methods in fall detection and human activity recognition [18]. It implies an ensemble of decision trees aiming to process the inputs into them, and computing the output class as the most frequent solution of the given trees.
- *Support Vector Machines (SVM)*. It is also a popular machine learning classifier for fall detection systems [5]. The idea of SVM is to map the inputs to a different space in which a hyper-plane separates the output classes. These hyper-planes are built over kernels that are trained to fulfill the classification task.
- *Multi-Layer Perceptron (MLP)*. This is a classical artificial neural network using perceptrons as activation functions. It is typically employed for general nonlinear classification [5].
- *k-Neareast Neighbors (KNN)*. This is an instance-based method that seeks the k-nearest neighbors of training points and compares them with an input data point. The output response is based on the most frequent class observed in the latter neighbors. This is a well-known method used in many applications because of its responsiveness and easiness of implementation [13].

Table IV summarizes the hyper-parameters of the classifiers implemented throughout this work.

Since the data sets are unbalanced (i.e. more *no-fall* than *fall* tags), we balanced the data sets doing an oversampling in the minority class (*fall*) by doubling the samples. In addition, we sub-sampled the majority class (*no-fall*) to one third. We split each feature data set in 70% for training and 30% for testing.

To evaluate the classifiers, we calculated the accuracy metric as in (1), where TP, TN, FP and FN represent the true positive, true negative, false positive and false negative values, respectively.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

For each classifier model, we conducted 10 repetitions. We reported the average performance of the classifier as shown in Table V. Notice that this approach was done for each feature data set at different window lengths (1-sec, 2-sec and 3-sec).

From Table V, we ranked the performance of the classifiers based on the sensor placement. Table VI summarized these results using the accuracy metric. It shows the IMU sensors sorted in descending order. As observed in Table VI, the best performance (ranked in first and second places) is obtained using the following sensor placements: neck, right pocket and waist. Also, left wrist performed the worst in almost all the classifiers. From that, the window length preference of classifiers are: 3-sec in RF, 3-sec in SVM, 2-sec in MLP and 2-sec in KNN for right pocket, and 3-sec in RF, 3-sec in SVM, 1-sec in MLP and 3-sec in KNN for neck.

From the above, we decided to use the right pocket as the best sensor placement, also considering easy adoption. Then, we decided to use 3-sec size in windowing (50% overlapping) and RF as the binary classifier.

TABLE IV
HYPER-PARAMETERS SELECTED FOR THE CLASSIFIERS.

Classifier	Hyper-parameters
RF	estimators $= 10$ min. samples split $= 2$ min. samples leaf $= 1$ bootstrap $= true$
SVM	$c = 1.0$ kernel $= radial\_basis\_function$ kernel coefficient $= 1/features$ shrinking $= true$ tolerance $= 0.001$
MLP	hidden layer size $= 100$ activation function $= ReLU$ solver $= stochastic\_gradient$ penalty parameter $= 0.0001$ batch size $= \min(200, samples)$ initial learning rate $= 0.001$ shuffle $= true$ tolerance $= 0.0001$ exponential decay (first moment) $= 0.9$ exponential decay (second moment) $= 0.999$ regularization coefficient $= 1e^{-8}$ max. epochs $= 10$
KNN	neighbors $= 5$ leaf size $= 30$ metric $= Euclidean$

D. Application Design

We developed a mobile application (App) for iOS version 13.0 for fall detection. The App was programmed for online testing purposes. It enables data collection from sensors embedded in an iPhone XS, namely three-axis accelerometer and three-axis gyroscope. Feature extraction and selection as well as the prediction of *fall* or *no-fall* are executed in real-time. The following describes the functions enabled in the App and its workflow (see Figure 2).

First, the subject ID, activity and trial must be captured before starting each trial of each activity performed by each subject. Once the trial is done the collection must be stopped (see the frontal view of the app in Figure 3). Raw data from sensors are acquired and stored in CSV files when the trial is executed. The structure of raw data collected with the smartphone is shown in Table VII. A timestamp with date and time is also saved.

Temporal features are extracted and selected from sensors raw data of the smartphone. We tested our fall detection system using a 3-second windowing with overlapping of 50% . The

TABLE V

ACCURACY PERFORMANCE OF THE CLASSIFIERS DEPENDING ON THE WINDOW LENGTH PER IMU SENSOR. IT INCLUDES THE MEAN (STANDARD DEVIATION) VALUES. NUMBERS IN BOLD REPRESENT THE BEST METRIC WITHIN THE WINDOW LENGTH PER SENSOR PLACEMENT.

Classifier	Window length	Left ankle	Right pocket	Waist	Neck	Left wrist
RF	1-sec	97.86 (0.21)	97.89 (0.28)	97.96 (0.23)	97.76 (0.38)	97.17 (0.33)
	2-sec	**98.42 (0.32)**	98.47 (0.43)	**98.50 (0.37)**	98.28 (0.48)	**98.32 (0.52)**
	3-sec	98.19 (0.44)	**98.57 (0.33)**	98.41 (0.43)	**98.70 (0.31)**	97.88 (0.34)
SVM	1-sec	95.17 (0.48)	95.52 (0.50)	**95.65 (0.29)**	95.43 (0.38)	**95.40 (0.33)**
	2-sec	95.29 (0.69)	95.13 (0.41)	95.38 (0.46)	95.32 (0.40)	95.37 (0.42)
	3-sec	**95.69 (0.41)**	**95.76 (0.50)**	95.65 (0.69)	**95.73 (0.54)**	95.24 (0.34)
MLP	1-sec	72.90 (21.60)	85.78 (3.57)	**79.62 (4.84)**	**83.70 (4.94)**	**77.20 (11.96)**
	2-sec	82.13 (2.95)	**86.16 (2.66)**	73.49 (13.06)	83.13 (7.64)	76.69 (18.29)
	3-sec	**83.12 (1.74)**	83.55 (2.82)	78.31 (4.14)	72.81 (23.41)	76.35 (19.19)
KNN	1-sec	87.31 (0.52)	91.76 (0.31)	89.98 (0.36)	90.43 (0.37)	87.32 (0.35)
	2-sec	**87.33 (0.83)**	**91.96 (0.61)**	91.20 (0.53)	90.80 (0.49)	87.19 (0.58)
	3-sec	88.56 (0.71)	91.32 (0.69)	**91.84 (0.59)**	**91.31 (0.60)**	**87.47 (0.94)**

TABLE VI

RANKING (TOP-BOTTOM) OF THE BEST SENSOR PER CLASSIFIER, BASED ON THE ACCURACY (IN PARENTHESIS).

Rank	RF	SVM	MLP	KNN
1	(98.70) Neck	(95.76) Pocket	(86.16) Pocket	(91.96) Pocket
2	(98.57) Pocket	(95.73) Neck	(83.70) Neck	(91.84) Waist
3	(98.50) Waist	(95.69) Ankle	(83.12) Ankle	(91.31) Neck
4	(98.42) Ankle	(95.65) Waist	(79.62) Waist	(87.47) Wrist
5	(98.32) Wrist	(95.40) Wrist	(77.20) Wrist	(87.33) Ankle

TABLE VII

DATA COLLECTED WITH THE SMARTPHONE

Sensor	Raw data
Accelerometer	x-axis y-axis z-axis Date Time
Gyroscope	x-axis y-axis z-axis Date Time

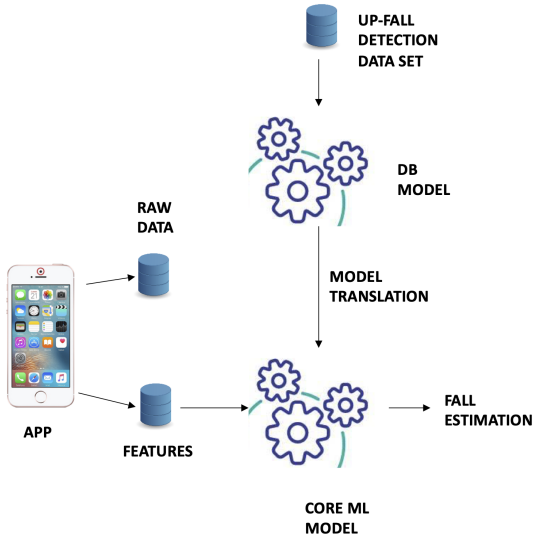


Fig. 2. Diagram of the workflow in the App design.

selected features are shown in Table VIII. A timestamp with date and time is added to features file.

The best 3-sec size in windowing (50% overlapping) Random Forest binary classifier was deployed for testing in the smartphone. We used Core ML framework [19] to integrate our machine learning model into the app. The classifier was applied in real-time to deliver the fall estimation which was added and saved in the feature file.

TABLE VIII

FEATURES EXTRACTED AND SELECTED FROM SENSOR DATA

Sensor-axis	Feature
Accelerometer: z-axis (g)	Standard deviation
Accelerometer: y-axis (g)	Third quartile
Gyroscope x-axis (deg/s)	Skewness
Accelerometer: y-axis (g)	Kurtosis
Gyroscope z-axis (deg/s)	Median
Accelerometer: z-axis (g)	Maximal amplitude
Gyroscope z-axis (deg/s)	Autocorrelation
Gyroscope z-axis (deg/s)	Kurtosis
Accelerometer: y-axis (g)	Maximal amplitude
Accelerometer: x-axis (g)	Autocorrelation

E. Online Testing

Lastly, we designed an online testing to measure the performance of our fall detection system on the smartphone. For this purpose, we created in this work a new dataset with different conditions than those implemented in the UP-Fall Detection dataset. With the new dataset, the deployed binary classifier is tested on the smartphone in real-time. This new dataset is intended to measure the flexibility and portability of our fall detection system. The details of the new dataset and the description of the experiments are provided following.

1) Dataset for Online Testing: This dataset was created with different subjects and different sensors, in contrast to the UP-Fall Detection dataset. At first, we recruited fourteen young subjects without any impairments (1.70 ± 0.09 m height and 68.8 ± 17.2 kg weight), 6 males and 8 females, ranging from 19 to 37 years old. They performed the same 11 actions

TABLE IX
PERFORMANCE OF THE ONLINE TESTING FOR EACH ORIENTATION OF THE SMARTPHONE IN THE RIGHT POCKET. IT REPORTS THE ACCURACY AND RECALL METRICS FOR EACH SUBJECT, AND THE AVERAGE.

Subject	Orientation 1		Orientation 2		Orientation 3	
	accuracy (%)	recall (%)	accuracy (%)	recall (%)	accuracy (%)	recall (%)
1	70.94	85.71	92.94	80.0	92.42	83.33
2	80.20	75.0	93.51	100.0	95.09	100.0
3	72.25	66.67	87.14	50.0	85.71	37.50
4	75.59	71.43	91.63	75.0	95.26	33.33
5	83.17	28.57	98.56	100.0	98.09	80.0
Average	75.79	57.78	91.79	69.23	92.55	57.50

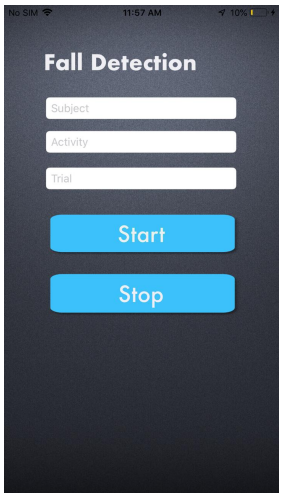


Fig. 3. Screenshot of the App for online testing the fall detection system.

(activities and falls) as summarized in Table I, three trial each. The same protocol for action performance and data pre-processing was adopted from [9], [16].

As measurement device, we used one iPhone XS smartphone with its three-axis accelerometer and three-axis gyroscope, placed in the right pocket and collected at 100 Hz. For manual tagging purposes, we placed a camera in lateral view; but images grabbed from it were not used in the fall detection system.

We changed the orientation of the smartphone in each trial. In the first trial (*orientation 1*), the smartphone was oriented upside down and with the screen flipped to the body of the subject. In the second trial (*orientation 2*), the smartphone was oriented upright and with the screen flipped to the body of the subject. In the third trial (*orientation 3*), the smartphone was oriented upright and with the screen flipped out the body of the subject.

We encountered battery and communication issues when dealing with the smartphone. In the first case, sensor information lagged so it did not match with the performance of the subject. In the second case, portions of data were not consistently transferred to the server. For those reasons, in this paper, we only report information from five subjects in which these issues were not found.

The work was approved by the Research Committee of the School of Engineering in Universidad Panamericana (Mexico). For this study, all the subjects that participated previously filled out an agreement, considering the regulations and data policies applicable. The decision to participate in these experiments was voluntary.

2) Settings of the Fall Detection System: For experimentation, we setup the app in the smartphone to retrieve data from the built-in sensors (accelerometer and gyroscope). From the sensor placement analysis, we defined a windowing approach of 3 seconds with 50% overlapping. At the end of each window, the raw sensor data was processed and the same features implemented in the machine learning model were extracted locally in the smartphone. These features were input to the deployed model and the estimated values were collected in real-time. For further analysis, all the raw sensor data, the features extracted and the estimations done in the smartphone were sent to the server.

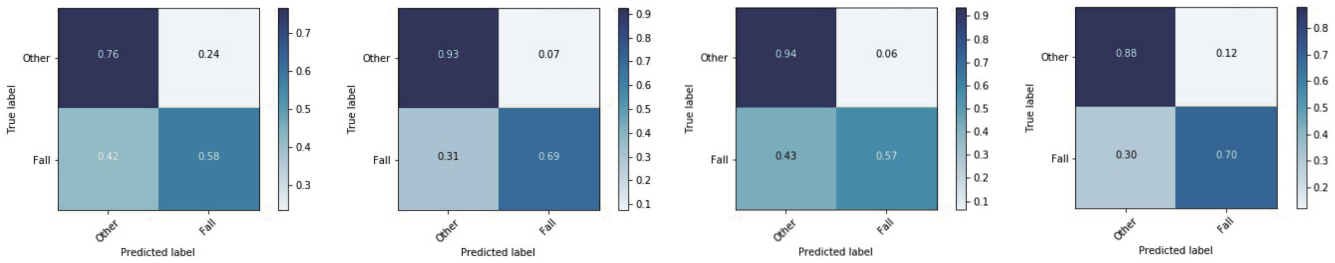
3) Metrics of Evaluation: For this test, we calculated the accuracy metric as in (1) and the recall metric as in (2).

$$recall = \frac{TP}{TP + FN} \tag{2}$$

IV. RESULTS OF ONLINE TESTING

This section reports the results of the online testing of the deployed binary classifier for fall detection, as described above. The results reported the performance of the five subjects with the smartphone wore in the right pocket. The estimated falls were computed on the mobile device in real-time considering a windowing approach of 3 seconds with 50% overlapping. Table IX summarizes the results of the online testing performance in the different orientations (trials) of the smartphone (trials). The overall performance obtained was 87.56% of accuracy and 69.79% of recall.

In Table IX, it can be observed that the performance of the deployed fall detection system depends on the orientation of the smartphone. In this regard, both *orientation 2* and *orientation 3*, where the smartphone was oriented upright, performed better than *orientation 1* based on the accuracy metric. It is remarkable to say that the IMUs in the UP-Fall Detection dataset were randomly oriented [9], thus orientation in the online testing gives insights on the preferable orientation of the smartphone (i.e., upright or upside-down). In terms of the recall metric, all the orientations performed similar results, with some preference on *orientation 2* (the screen of the smartphone is flipped to the body of the subject).



(a) (b) (c) (d)

Fig. 4. Normalized confusion matrices of the average performance in the online testing: (a) *orientation 1*, (b) *orientation 2*, (c) *orientation 3*, and (d) overall performance (87.56% of accuracy and 69.79% of recall).

Moreover, Figure 4 shows the average normalized confusion matrices of the three orientations and the overall performance summarizing all the orientations. The confusion matrices of the orientations (Figure 4(a)–(c)) show that *fall* actions are estimated less accurate than *no-falls*. This behavior is associated to the imbalanced dataset, thus better techniques for training machine learning models with imbalanced data are required.

Furthermore, the confusion matrix of the overall performance reveals that the deployed binary classifier model estimates the *fall* class 70% of the times while the *no-fall* class is detected 88% (Figure 4). In contrast to the literature, our proposed deployed fall detection system outperforms the work reported in [8] with 59% of precision and 44% of recall.

From the results above, it is evident that there is a decreasing in the predictive power of the deployed fall detection system in online testing (overall accuracy of 87.56%) when comparing with the binary classifier model trained directly from the UP-Fall Detection dataset (overall accuracy of 98.57%, see Table V). Differences in sensors, e.g. sampling rate, resolution, orientation, limited the performance of the deployed model. We consider that advanced techniques, e.g. transfer learning, should be applied for improving the performance of the deployed fall detection system.

V. Conclusions

In this paper, we presented a solution for fall detection, and designed an online testing to measure the performance of our fall detection system on a smartphone. For this purpose, we designed new experiments and created a new dataset with different conditions than those implemented in the previously gathered UP-Fall Detection dataset.

Our methodology, including online testing in deployed fall detection systems, is one of the few existing works related to evaluate the performance of deployed machine learning models online. Even if many of the research in fall detection systems report high performance of the classifier models, there is still a lack of understanding on how to translate and deploy those models on real, portable, constrained –and possibly different– platforms (e.g. different smartphone models).

For future work, it is recommended to test other techniques like transfer learning to improve the performance of the classification. In order to enable our App to work in real world, it is necessary to implement the ability to run in the background, and to test it in a larger pool of subjects.

References

[1] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.

[2] D. Ajerla, S. Mahfuz, and F. Zulkernine, "A real-time patient monitoring framework for fall detection," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[3] N. Noury, P. Rumeau, A. Bourke, G. ÓLaighin, and J. Lundy, "A proposal for the classification and evaluation of fall detectors," *Irbm*, vol. 29, no. 6, pp. 340–349, 2008.

[4] 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.

[5] ——, "A comparison of public datasets for acceleration-based fall detection," *Medical engineering & physics*, vol. 37, no. 9, pp. 870–878, 2015.

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

[7] A. Özdemir, "An analysis on sensor locations of the human body for wearable fall detection devices: Principles and practice," *Sensors*, vol. 16, no. 8, p. 1161, 2016.

[8] K. Frank, M. J. Vera Nadales, P. Robertson, and T. Pfeifer, "Bayesian recognition of motion related activities with inertial sensors," in *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing-Adjunct*. ACM, 2010, pp. 445–446.

[9] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez-Martínez, and C. Peñafort-Asturiano, "UP-fall detection dataset: A multimodal approach," *Sensors*, vol. 19, no. 9, p. 1988, 2019.

[10] G. Vavoulas, C. Chatzaki, T. Malliotakis, M. Pediaditis, and M. Tsiknakis, "The mobiact dataset: Recognition of activities of daily living using smartphones." in *ICT4AgeingWell*, 2016, pp. 143–151.

[11] M. M. Hassan, A. Gumaei, G. Aloi, G. Fortino, and M. Zhou, "A smartphone-enabled fall detection framework for elderly people in connected home healthcare," *IEEE Network*, vol. 33, no. 6, pp. 58–63, 2019.

[12] Y. Chen, J. Wang, C. Yu, W. Gao, and X. Qin, "Fedhealth: A federated transfer learning framework for wearable healthcare," *arXiv preprint arXiv:1907.09173*, 2019.

[13] 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.

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

[15] 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.

[16] C. J. Penafort-Asturiano, N. Santiago, J. P. Nunez-Martinez, H. Ponce, and L. Martinez-Villasenor, "Challenges in data acquisition systems: Lessons learned from fall detection to nanosensors," in *2018 Nanotech-*

nology for Instrumentation and Measurement (NANOfIM). IEEE, 2018, pp. 1–8.

[17] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[18] S. Kozina, H. Gjoreski, and M. G. Lustrek, "Efficient activity recognition and fall detection using accelerometers," in *International Competition on Evaluating AAL Systems through Competitive Benchmarking*. Springer, 2013, pp. 13–23.

[19] A. Inc, "Framework core ml:integrate machine learning models into your app," 2020 (accessed April 3, 2020). [Online]. Available: https://developer.apple.com/documentation/coreml