Gesture-based incident reporting through smart watches

Panagiotis Kasnesis Christos Chatzigeorgiou Lazaros Toumanidis Charalampos Z. Patrikakis

Department of Electrical and Electronics Engineering,

University of West Attica, Athens, Greece

pkasnesis@yahoo.gr , {chrihatz, laztoum, bpatr}@uniwa.gr

***Abstract*— With the capabilities and autonomy of smart watches constantly increasing, there is the need for introducing applications which can exploit their full potential, establishing the use of smart watches in our daily routine. The field of personal safety and security provides an excellent basis over which applications can be developed, enabling the use of wrist worn devices as tools for easy, discreet and efficient reporting of incidents or suspicious behavior. However, current practices in report creation using smart watches rely on methods and interfaces, such as taking pictures and writing text, without taking into account gesture-based input. In this paper, we present the design a smart watch-based approach, which utilizes a Deep Learning model, to recognize specific user gestures that could result in reporting hazardous situations and could alert the authorities for assistance. We evaluated the performance of the model by training it to distinguish 5 predefined gestures from a set of random gestures performed by 9 subjects wearing a smart watch on their dominant arm. Our Deep Learning model surpasses the performance of conventional classifiers that rely on hand-crafted features and produced gesture recognition accuracies above 98% in 26,061 motion signal samples by fusing the automatically extracted features of 3-axial accelerometer signals. We conclude by discussing the related issues we have encountered by using the proposed application in real-time use and providing future directions.**

Keywords*— *deep learning, gesture recognition, smart watch, public safety

I. INTRODUCTION

The Internet of Things (IoT) is considered to power the next “Industrial Revolution”, as the number of IoT devices is expected to reach 50 billion by 2020 [1] and their applications will enhance the way people live, travel, communicate and entertain. The IoT paradigm will turn into reality Mark Weiser’s quote “*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*” [2]. However, in order to develop useful applications, the raw IoT data have to be collected and analyzed in a way that useful information can be extracted from them. In other words, the IoT applications have to become situation-aware.

Situation Awareness (SA) is defined as “*the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future*” [3]. In other words, SA refers to the possibility of knowing deeply the current situation by observing and analysing the surrounding environment and to foresee every change that can happen. Thus, SA is critical when it comes to public safety [4], and the related countermeasures in crisis situations is a key factor of a successful work of first responders and Law Enforcement Agents (LEAs). The more the situation is rich of useful information, the more the first responders and LEAs can act rapidly in properly responding and mitigating the risk.

One of the main computer science branches that serves the IoT paradigm, is wearable computing. Wrist worn devices, such as smart watches and activity trackers, are equipped with a wide range of small embedded sensors capable of monitoring situation-aware information. To this end, there are several existing commercial mobile [5][6][7] and wearable [5] applications used for creating a report that may include pictures, text and the GPS coordinates of the users, aiming to minimize the LEAs response time and improve SA.

Nevertheless, these approaches are based on non-discreet ways for crime reporting, lack of automation and, in case they are not, demand from the users to be familiarized with the application [8]. Gestures could, definitely, be considered as a solution to this issue. However, when it comes to identifying the user’s activity (e.g., via hand gestures), the signals that are collected by motion sensors, such as 3-axial accelerometers and gyroscopes, must be preprocessed, segmented, transformed (e.g., feature representation) and, finally, classified using a machine learning algorithm [9]. On top of that, the feature extraction process by human experts is time consuming and, in some cases, not effective. Deep Learning (DL) algorithms can tackle this problem, since they have the ability to automatically extract features [10]. What is more, previous implementations of DL approaches applied to sensor-based human activity recognition and hand gesture recognition have outperformed past techniques based on hand-crafted features [11][12].

Creating the ability to recognize hand gestures for incident reporting is the focus of our paper. In particular, the present paper introduces a hand gesture classifier for wrist worn smart devices, which tries to enhance the feeling of the users’ security, by giving them the ability to report incidents discreetly and instantly through the use of hand gestures. This way, citizens will be able to communicate with the LEAs, even when they are in a hazardous situation and they cannot call or shout for help. The contribution of our work can be summarized in the following:

- We implemented a novel hand gesture classifier based on a deep Convolutional Neural Network (ConvNet), and evaluated it by comparing it to machine learning techniques that are based on hand-crafted features.
- We developed a smart watch and a smartphone application that could be used to collect motion data in the wild.
- We built a hand gesture dataset containing 3-axial accelerometer, gyroscope and magnetometer signals consisted of 5 annotated gestures that can be extended by a wider set of gestures.

II. Related Work

In this section, we report significant mobile/wearable techniques proposed in the literature that are based on machine learning techniques for improving public safety and not for detecting every day activities, such as smoking [13] and eating [14]. These works make use of motion sensors in order to classify the hand gestures, the activities performed by the users, or identify the situation they are in.

Loeffler [15] presents the use of wearable sensors to recognize wrist movements that imply firearm movement in order to detect gunshots. Classification accuracy of 99.4 percent is achieved according to his report when tested against a hold-out sample of observations. In a related work, the authors of [16], expand the firearm detection task by studying the problem of identifying various categories of firearms (i.e., handgun, rifle, shotgun) and recognizing whether a firearm is autoloaded or manual.

The use of an intelligent accident detection and reporting algorithm is presented in [17]. A custom hardware module is used to transmit signals when an accident occurs, that can be used to determine the location of the accident. Car accidents detection is also described in [18], where the authors make use of a smartphone, in order to provide SA to emergency responders. Since distracted driving is one of the most significant dangers to road safety today, the authors in [19] introduce an algorithm for detecting it. In the experiment described in their paper, 16 adult participants were asked to use a driving simulator while wearing in their wrist a smart device equipped with an accelerometer and a gyroscope, achieving promising results. On the other hand, distraction from the pedestrian perspective is studied in [20], where a novel complex activity recognition framework was designed exploiting motion data from the users' mobile and wearable devices.

Another, well-investigated computer task is that of fall detection, which recognizes if a subject that carries a mobile (phone) or wearable device, has fallen. The consequences of a fall for elderly people may lead to their hospitalization and sometimes are fatal. However, fast assistance reduces the effects of a fall, so solutions in early detection of fall can have a significant impact. Some commercial wearable devices, such as GoSafe [21] produced by Philips, offer their users an emergency button in case they fall. Nevertheless, the elderly are not familiar with such technologies, while their use may not always be possible after a fall. As a result, many researchers have focused over the last few years on creating smartphone-based fall detection applications [22]. In [23] researchers used the accelerometer, magnetometer and gyroscope sensors of a smartphone to achieve up to 100% for sensitivity and 93% for specificity, while [24] focuses on reducing false alarms using the accelerometer data of a smart watch. It should be noted that fall detection applications have a social aspect because they send alert messages, to people related to the fallen user via their social media accounts, but they are human-centric. In addition to this, recently, Apple watches introduced the ability to send a distress signal after detecting a fall and subsequently detecting immobility [25].

However, though the aforementioned approaches have promising results, they are targeting specific domains (e.g., crash accidents) and do not fully exploit the capabilities of understanding the situation in which the user is engaged, which in some cases could be life threatening, and/or could be requiring the communication with the proper authorities and LEAs. Furthermore, they do not offer the ability to proactively respond to situations, such as reporting of abnormal or suspicious behavior. In particular, humans are not considered as sensors but as an observable parameter in the monitored environment. Thus, we propose the use of gestures to understand a variety of actions, which could be linked to different situations or incidents, and if needed establish a discreet communication channel between the LEAs and the citizens.

III. Dataset Collection Procedure and Application

For the data collection process, we developed a supplementary mobile application which togetherwith the smart watch application, was used to collect and annotate incoming motion data from the smart watch sensors and upload them on a server. The data collected include recordings of 3-xial accelerometer, gyroscope and magnetometer sensors, along with a timestamp and a unique ID of the device. In addition, optional personal info of each subject like age, height the used wrist (i.e., left or right) were also collected.

For the training process, we selected a set of five *Gestures of Interest* (*GoI*):

- *Reach for Wallet* (*RW*): The user reaches and opens his/her wallet from his/her pocket (assumed to be placed in the back pocket of the trouser), using the hand with the wearable.
- *Look & Reach for Wallet* (*LRW*): The user first taps several times his/her body, as if trying to locate where (in which pocket) the wallet is,and then (s)he reaches it and makes a move as if (s)he gives it to someone.
- *Hands Up* (*HU*): The user raises his/her hands, acting like (s)he is under immediate threat (i.e., at gunpoint).
- *Distress Signal* (*DS*): The user shakes quickly his/her hand a few times, with the hand pointing down. The gesture (similar to a tremor of hand), represents the anxiety that can be occurred under stressful situations.
- *Wave* (*W*): The user acts like (s)he waves at someone.

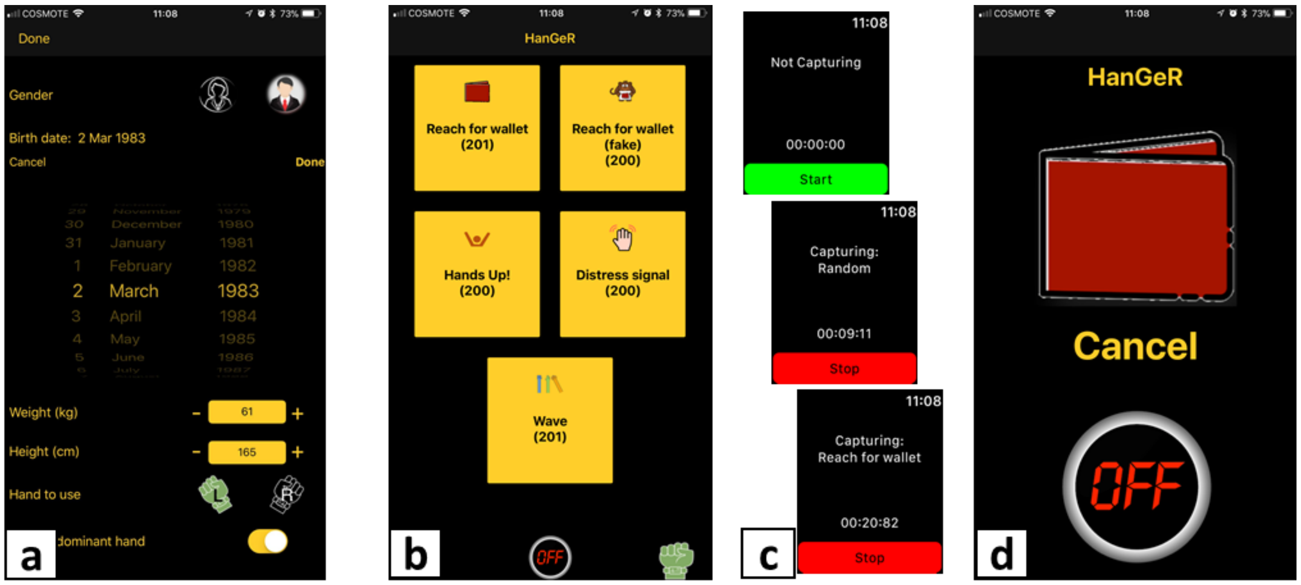

Fig. 1. a. Filling the additional info of the user that will wear the smart watch and whose data will be collected, b. the screen of the mobile application that will be used for the annotation of the data during the training phase of the machine learning model, c. The screens of the smart watch application, d. The screen in the mobile application that describes the current gesture. The captured data are recorded (and annotated) until the OFF button is pressed.

At first, the user wears the smart watch and then, opens the mobile application where a screen is presented, asking the user to enter his/her info, as seen in Fig. 1a. After registering this information, the screen in Fig. 1b is presented, displaying the available for annotation gestures. On the bottom right of the screen, the connectivity with the smart watch is indicated, with the corresponding icon changing its color accordingly: green when a smart watch with the application is present and red when a smart watch is absent, or connection to the mobile phone is lost (i.e. out of reach), or doesn't have the application installed. Tapping the gear on the bottom left allows the user to change his/her information at any time.

By tapping the "START" button the user is able to start capturing random movements. In order to capture data for a specific move, the user has to tap one of the yellow buttons and make the relevant move with his/her hand (Fig. 1b). Meanwhile, the smart watch shows whether it is capturing the user's gestures or not, and what gesture is currently capturing. There is also the option to start or stop capturing by tapping the appropriate button (Fig. 1c). When the user selects a gesture to capture, the screen changes and the capturing movement is shown along with two buttons (Fig. 1d). Through the use of the "Cancel" button, the user has the ability to discard the current gesture in case (s)he makes a wrong move. When the gesture is completed, the user taps the "OFF" button in order to annotate properly the data. If gesture capturing is enabled, the smart watch application gathers nine signals in the background (i.e., from the 3-axial accelerometer, 3-axial gyroscope and 3-axial magnetometer) and uploads them at periodic intervals to a server, through the mobile application.

Finally, it should be noted that random movements (e.g., do computer work, walking, scratching head etc.) were also collected, constituting the sixth class in our classification scheme. The created Hand Gesture Recognition (HanGeR) dataset was segmented using a 5-second window and no overlap, leading to the creation of 26,061 samples[1] (i.e., discrete gestures). Nine subjects wore a Sony SmartWatch 3 SWR50 [26] on the wrist of their preference and performed in total 1,543 RW, 1,518 LRW, 1,498 HU, 1,532 DS, 1,520 W and 18,450 random movements (R). The average sampling rate for the 3-axial accelerometer is approximately 200 Hz, while for the 3-axial gyroscope is approximately 186 Hz and the 3-axial magnetometer is approximately 103 Hz.

IV. Deep ConvNet Architecture

This section describes in detail the architecture of the proposed deep ConvNet [27] that is used to identify the *GoI*, produced by a user/citizen wearing a wrist worn device. ConvNets have proven to be capable of automatically extracting the temporal local dependencies of time-series signals and researchers have proven that motion signals are correlated with each other [28]. Moreover, we take advantage of our previous work [29], where we proved that by processing separately motion signals to extract features from them and applying afterwards sensor fusion to the low-level extracted features, using a 2D convolution operation over them, more general activity/gestures patterns are discovered. Thus, we developed a deep ConvNet, which takes as input vertically stacked motion signals in order to exploit the semantics and the grid-like topology of the input data, in contrast with Fully Connected Neural Networks (FCNNs) [30].

The product of a convolution operation is the summation of the elementwise multiplication between a part of the input tensor (receptive field) and a filter w. In particular, the i^{th} product element of a discrete 1D convolution between the input array x and a 1D filter w equals:

[1] https://consert.eee.uniwa.gr/datasets/

$$c_i^{l,q} = b^{l,q} + \sum_{d=1}^{D} w_d^{l,q} x_{i+d-1}^{l-1,q} \quad (1)$$

where l is the layer index, q is the activation map index, D is the total width of the filter w, and b is the bias term. While, the i^{th}, j^{th} product element of a discrete 2D convolution between input matrix x and 2D filter w equals:

$$c_{i,j}^{l,q} = b^{l,q} + \sum_{h=1}^{H} \sum_{d=1}^{D} w_{d,h}^{l,q} x_{i+d-1,i+h-1}^{l-1,q} \quad (2)$$

where h is the total height of the filter w.

The network hyperparameters were optimized by minimizing the cross-entropy loss function using the Adam optimizer [31]. The optimizer's hyperparameter values were as follows: the learning rate was equal to $5x10^{-4}$, the beta1 equal to 0.9, beta2 equal to 0.999, and epsilon equal to 1E-08. Moreover, we set the batch size equal to 64 and the minimum number of epochs to 500, but the training procedure was automatically terminated if the best training accuracy had not improved after 100 epochs.

In order to obtain subject dependent results, we used a 9-fold cross-validation technique and split our dataset into training set containing 7 subjects, and validation set and test set containing 1 subject. The model that achieved the lowest error rate on the validation set was saved, and its filters were used to obtain the accuracy of the model on the test set. Finally, we normalized each sensor signal's values by subtracting the mean value and dividing by the standard deviation.

The final architecture of the developed ConvNet is as follows:

- Layer 1: 16 1D convolutional filters with a size of 1x15, followed by a ReLU [32] activation function, a strided 1D max-pooling operation [27] with size 1x8 and a dropout [33] probability equal to 0.5.
- Layer 2: 24 1D convolutional filters with a size of 1x15, followed by a ReLU activation function, a strided 1D max-pooling operation with size 1x8 and a dropout probability equal to 0.5.
- Layer 3: 32 2D convolutional filters with a size of 3x15. This is followed by a ReLU activation function, a global max pooling operation [34] and a dropout probability equal to 0.5.
- Layer 4: Dense layer with number of classes as output units, i.e., W^4 has the shape 32x6, followed by a *softmax* activation function.

V. Experimental Results

The experiments were executed on two computer workstations in order to accelerate the training process. The first one is equipped with a NVIDIA GTX Titan X GPU, featuring 12 gigabytes RAM, 3072 CUDA cores, and a bandwidth of 336.5 GB/s, while the second with a NVIDIA GTX 1080 Ti GPU featuring 11 gigabytes RAM, 3584 CUDA cores and a bandwidth of 484 GB/s. We used Python as programming language, and specifically the Numpy library for matrix multiplications, data preprocessing and segmentation, scikit-learn for training the conventional machine learning algorithms and the Keras high-level neural networks library using as backend the TensorFlow library. In order to accelerate the tensor multiplications, we used CUDA Toolkit in support with the cuDNN, which is the NVIDIA GPU-accelerated library for deep neural networks. Both workstations had 16.04 Ubuntu Linux operating system.

The proposed DL algorithm was evaluated by comparing its performance to conventional classifiers: Support Vector Machines (SVM), Logistic Regression (LR), k-nearest neighbors (k-NN), Decision Tree (DT), Random Forest (RF) and FCNN. Due to the fact that too many features in some cases decrease the accuracy [19][35] and too few features may increase the model's bias, we selected to extract a limited set of time-dependent features from the accelerometer signals, which have been proven to be more robust and efficient than frequency-dependent features in context recognition [9], to feed the algorithms. Table I presents them along with a short description.

The evaluation strategy we adopted was twofold: 1) evaluate the performance of our model in an almost perfectly balanced dataset that contained only the 5 *GoI* and 2) evaluate the performance of our model in a more realistic scenario where random movements were included. The results of the former are illustrated in Fig. 2. The proposed deep ConvNet achieved the highest performance in terms of accuracy

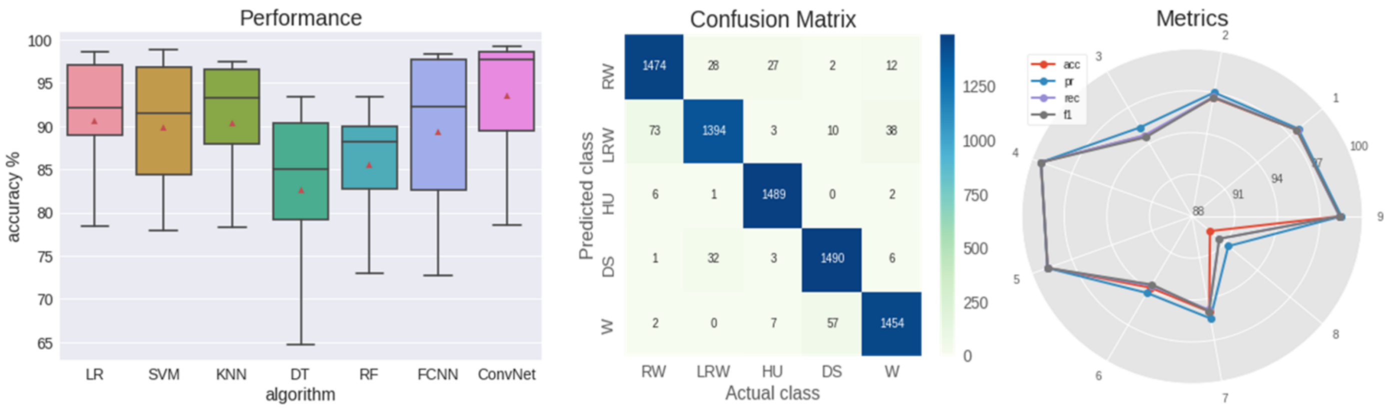

Fig. 2. Plots diplaying: a) the performance, in terms of accuracy, of the selected machine learning algorithms, b) the confusion matrix of the ConvNet and c) the accuracy, precision, recall and F1_score metrics per subject of the ConvNet model for the 5 *GoI*

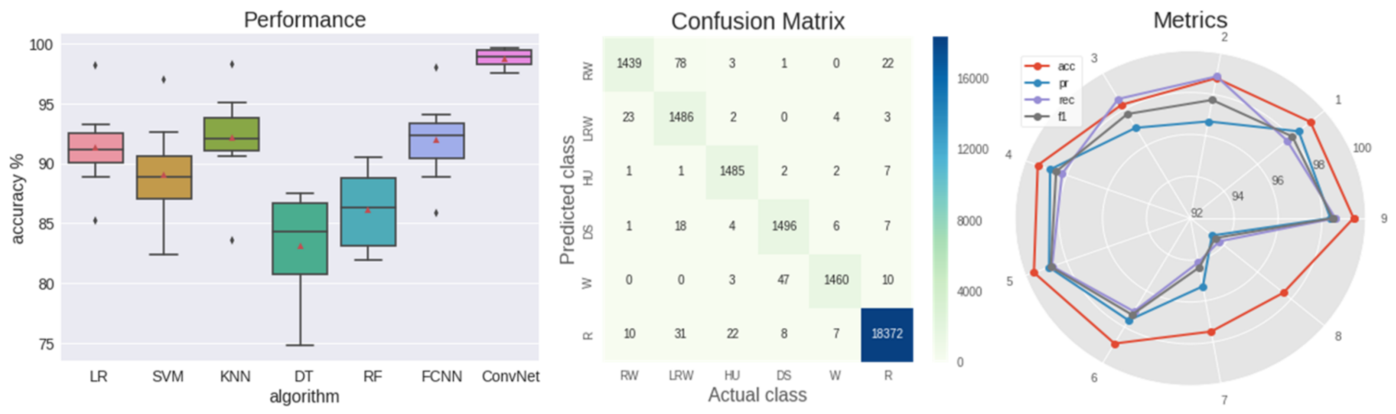

Fig. 3. Plots diplaying: a) the performance, in terms of accuracy, of the selected machine learning algorithms, b) the confusion matrix of the ConvNet and c) the accuracy, precision, recall and F1_score metrics per subject of the ConvNet model for the 5 *GoI* and the random movements.

(~93.56%). Plotting the confusion matrix, it is obvious that a lot of *LRW* gestures were misclassified as *RW*. Consequently, the *RW* gesture had the lowest precision and the *LRW* the lowest recall. The radar (spider) chart displays in two-dimensions, the per subject performance of the ConvNet using four metrics, the accuracy (red color), precision (blue color), recall (purple color) and F1_score (gray color); this chart illustrates if our model generalizes well (i.e., achieves similar performances for all the subjects), which is accomplished with the exception of subject 8 that has the lowest results (F1-score: 90.49%), in contrast to subject 4 that has the highest (F1-score: 99.31%).

TABLE I. SELECTED FEATURES

Feature	***Description***
Mean	Average value
Min	Minimum value
Max	Maximum value
Median	Median value
Standard deviation	Measure of dispersion
Skewness	The degree of asymmetry of the signal distribution
Kurtosis	The degree of "peakedness" of the signal distribution

It should be noted that the machine learning algorithms had as input only the 3-axial accelerometer signals. The motion sensors (gyroscope and accelerometer) that Sony SWR50 is equipped with, are on separate modules, and are not synchronized, while they have different sampling rate and suffer from sampling rate instability (regularity of the timespan between successive measurements). As a result, fusing all the signals together resulted in acquiring worse results (about 2% decrease in terms of accuracy). In addition to this, the network consists of 32,566 and is considered to be very light-weight if we take into consideration the fact that in our previous work PerceptionNet [29] contains ~ 485,136 parameters and SqueezeNet [36] uses 421,098 parameters for inference.

Regarding the second evaluation strategy, the proposed deep ConvNet achieved the highest accuracy (~98.73%), as it is displayed in Fig. 3. Interestingly, the accuracy of our model increased over 5% by adding the random movements (i.e., increasing the dataset size using unlabeled gestures), while the rest of the models improved only by 1%. Furthermore, due to class imbalance (i.e., random samples are more than the other gestures) we can see in the radar chart that the precision, recall and F1-score metrics differ a lot from that of accuracy, however, their values are, also, promising (precision: 97.15 %, recall: 97.31% and F1-score: 97.16%).

By plotting the confusion matrix we see that the model, again, struggles to distinguish the *RW* gesture from the *LRW* gesture. Moreover, it misclassified a lot of *W* samples as *DS*; this misclassification could have been solved if the gyroscope signals were also used, since *DS* gesture contains a lot of rotational movement. Furthermore, it should be noted that 49 out of 7,611 *GoI* were classified as random, which can be translated into the fact that once per 155 dangerous situations an incident report is not sent to the platform. Finally, and we obtained 78 false positive incident reports (total number of misclassified random gestures). This means that a smart watch application that would run in real-time and use a 5 second time window with 80% overlap it will falsely upload a report to the platform every 4 minutes and 15 seconds. Nonetheless, this can be addressed by including the GPS coordinates in the report or providing the ability of cancelling the report before it is sent (i.e., a prior warning through a notification on the smartwatch).

VI. CONCLUSION

In the current paper, we proposed and evaluated a gesture-based incident reporting application that could be deployed on smartwatches using a DL model. This technique allows users to use gestures as interface in order to alert the LEAs for hazardous events through the automatic preparation and sending of notifications triggered by different gestures. The results we obtained using the hand gesture recognition dataset that we built, were very encouraging, since the DL model managed to produce accuracies above 98% and F1-score over 97% by using a late sensor fusion technique to automatically extract features of 3-axial accelerometer signals. It is of course important to highlight that our proposal comes not to replace the existing methods for emergency reports but rather to provide a method for sending more elaborate reports. We argue that gesture-based applications will be broadly deployed to wrist worn devices in the near future to enable discreet and remote communication.

Future steps target at testing the proposed technique in a real-world scenario while improving the model's applicability and the deployment of the DL model in commercial smart watches, even if they do not rely on the same OS and their sensors have different sampling rate. Moreover, in order to increase the performance, the use of other motion signals (e.g., Magnetometer) and that of a trigger gesture (such as in speech recognition task) before executing the *GoI* should be investigated. Finally, one-shot learning techniques could be examined by exploiting the high-level features that are extracted after the global max pooling layer as embeddings, in order to give the users, the capability to add their own gestures.

Acknowledgment

This work was funded by the European Commission under project TRILLION, grant number H2020-FCT-2014, REA grant agreement n° [653256]. Moreover, we gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan-X GPU used for this research.

References

[1] "Bringing Things to Life with IoT". Available at: https://blog.intuz.com/bringing-things-to-life-with-iot/. Last accessed at 09/10/2018.

[2] M. Weiser, "The computer for the twenty-first century," in Scientific American, vol. 165, pp. 94, 1991.

[3] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," in Human Factors, vol. 37, no. 1, pp. 32-64, 1995.

[4] E. Moradi-Pari., A. Tahmasbi-Sarvestani & Y. P. Fallah, "A hybrid systems approach to modeling real-time situation awareness component of networked crash avoidance systems," in IEEE Systems Journal, vol. 10, no. 1, pp. 169-178, 2016.

[5] "CommandWear". Available at: https://commandwear.com/features. Last accessed at 21/09/2018.

[6] "CrimeReports". Available at: https://www.crimereports.com. Last accessed at 21/09/2018.

[7] "CitizenCop". Available at: http://www.citizencop.org. Last accessed at 21/09/2018.

[8] J. Gong, Z. Xu, Q. Guo, T. Seyed, X. Chen, X. Bi & X.-D. Yang, "WrisText: One-handed Text Entry on Smartwatch using Wrist Gestures," in proceedings of CHI 2018, April 2018.

[9] D. Figo, P. C. Diniz, D. R. Ferreira and J. M. Cardoso, "Preprocessing techniques for context recognition from accelerometer data" in Personal and Ubiquitous Computing, vol. 14, no. 7, pp. 645–662, 2010.

[10] P. Kasnesis, C. Z. Patrikakis & I. S. Venieris, "Changing Mobile Data Analysis through Deep Learning," in IEEE IT Professional Mobile Data Analytics, vol. 19, pp. 17–23, 2017.

[11] C. A. Ronao, S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," in Expert Systems with Applications, vol. 59, pp. 235-244, October 2016.

[12] F. J. Ordóñez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition," in Sensors, vol. 16, no. 1, pp. 1-25, January 2016.

[13] A. Parate, M.-Ch. Chiu, Ch. Chadowitz, D. Ganesan, E. Kalogerakis, "RisQ: recognizing smoking gestures with inertial sensors on a wristband," in Proceedings of the 12th MobiSys, pp. 149-161, 2014.

[14] E. Thomaz, I. Essa, Gr. D. Abowd, "A practical approach for recognizing eating moments with wrist-mounted inertial sensing," in Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp), pp. 1029-1040, 2015.

[15] C. E. Loeffler, "Detecting Gunshots Using Wearable Accelerometers," in PLoS ONE, vol. 9, no. 9, 2014.

[16] Md Abdullah, Al H. Khan, D. Welshy and Nirmalya Roy, "Firearm Detection Using Wrist Worn Tri-Axis Accelerometer Signals," WristSense 2018 (PerCom Workshops), pp. 221-226, 2018.

[17] R. K. Megalingam, R. Nammily Nair and S. Manoj Prakhya, "Wireless vehicular Accident Detection and Reporting System," in ICMET, pp. 636-640, 2010.

[18] C. Thompson, J. White, B. Dougherty, A. Albright, C. S. Douglas, "Using Smartphones to Detect Car Accidents and Provide Situational Awareness to Emergency Responders" Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, vol. 48, pp. 29-42, 2010.

[19] B. Goel, A. K. Dey, P. Bharti, K. B. Ahmed and S. Chellappan, "Detecting Distracted Driving Using a Wrist-Worn Wearable," in WristSense 2018 (PerCom Workshops), pp. 233-238, 2018.

[20] N. Vinayaga-Sureshkanth, A. Maiti, M. Jadliwala, K. Crager, J. He and H. Rathore, "Towards a Practical Pedestrian Distraction Detection Framework using Wearables," in WristSense 2018 (PerCom Workshops), pp. 239 - 245, 2018.

[21] "GoSafe". Available at: https://www.lifeline.philips.com/safety-solutions/gosafe.html. Last accessed at 21/09/2018.

[22] "SafeBeep", Available at: https://www.safebeep.com/. Last accessed at 21/09/2018.

[23] I. N. Figueiredo, C. Leal, L. Pinto, J. Bolito and A. Lemos, "Exploring smartphone sensors for fall detection," mUX: The Journal of Mobile User Experience, 2016.

[24] I. Maglogiannis, C. Ioannou, G. Spyroglou, P. Tsanakas, "Fall Detection Using Commodity Smart Watch and Smart Phone," Artificial Intelligence Applications and Innovations, pp. 70-78, 2014.

[25] "Apple Watch Series 4". Available at: https://support.apple.com/en-us/HT208944. Last accessed at 31/10/2018.

[26] "Sony SmartWatch 3 SWR50". Available at: https://www.sonymobile.com/global-en/products/smart-products/smartwatch-3-swr50/#gref. Last accessed at 21/09/2018.

[27] I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning" MIT Press, ch. 10, pp. 330-372, 2016.

[28] W. Jiang and Z. Yin, "Human Activity Recognition using Wearable Sensors by Deep Convolutional Neural Networks", in Proceedings of the 23rd ACM conference on Multimedia, pp. 1307-1310, 2015.

[29] P. Kasnesis, C. Z. Patrikakis and I. S. Venieris, "PerceptionNet: PerceptionNet: A Deep Convolutional Neural Network for Late Sensor Fusion," in Intelligent Systems Conference, pp. 101-119, London 2018.

[30] C. Vollmer, H.-M. Gross, and J. P. Eggert, "Learning features for activity recognition with shift-invariant sparse coding", In Artificial Neural Networks and Machine Learning–ICANN, pp. 367-374, 2013.

[31] D. P Kingma and J. Ba, "A method for stochastic optimization," in International Conference on Learning Representation, 2015.

[32] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," In JMLR, vol. 15, pp. 1929-1958, 2014.

[34] M. Lin, Q. Chen, S. Yan, "Network In Network," In International Conference on Learning Representations, 2014.

[35] H. Almuallim and T. G. Dietterich, "Learning With Many Irrelevant Features," in Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 547–552, 1991.

[36] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5mb model size," arXiv:1602.07360, 2016.