Real-time Activity Recognition on Smartphones Using Deep Neural Networks

Licheng Zhang, Xihong Wu and Dingsheng Luo*
Key Lab of Machine Perception (Ministry of Education), Speech and Hearing Research Center
School of Electronics Engineering and Computer Science, Peking University
Beijing, 100871, China
Emails: {zhanglc, wxh, dsluo }@cis.pku.edu.cn
* Corresponding author

***Abstract*—Real-time activity recognition is an important research problem, which has drawn the attention of many researchers for many years. All previous works of real-time activity recognition needed to manually extract features and then fed these features into the classifiers as the inputs. However, due to the recognition time limit, they could only extract a small amount of features, which might not achieve good accuracy and could be improved. Besides, these features were manually extracted according to the experience of the researchers. Because researchers did not know which features worked better for classification, it was hard to choose suitable features, thus these manually extracted features might not be related to the specific classification task and have poor discrimination ability and could be improved. In this paper, we recommend deep neural networks for real-time activity recognition, which automatically learn suitable features and then perform classification. We collected accelerometer data of seven activities of interest on an android smartphone, including walking, running, standing, sitting, lying, walking upstairs and walking downstairs, and conducted experiments on our collected dataset to compare our method with traditional methods. Besides, we implemented a deep neural network on a smartphone and tested the recognition time of the model. The results showed that deep neural networks could achieve real-time performance and got higher accuracy than traditional methods.**

Keywords-deep neural networks; real-time activity recognition; smartphone;accelerometer; feature learning

I. INTRODUCTION

Real-time activity recognition is an important research problem, which has drawn the attention of many researchers in the past decade. During this period, a lot of works achieved the real-time performance of activity recognition. All of the works needed to manually extract features, which was a heuristic process and relied on the experience of the researchers. Then the features were fed into the classifiers as the inputs. However, due to the recognition time limit, they could only extract a small amount of features, which might not be enough to recognize different activities. Besides, these features were manually extracted, which relied on the experience of the researchers, thus might not have good discrimination ability of different activities. For instance, Kose et al. [1] collected accelerometer data on smartphones and applied low pass filter to raw data. Then they manually extracted four features: average, minimum, maximum, and standard deviation. Then they chose Clustered K-Nearest Neighbor (Clustered KNN) as the classifier. The approach has several problems. First of all, to reduce the computation time of smartphones during online application phase, the researchers only extracted four statistical features, which achieved the real-time effect, but might not achieve good accuracy and could be improved. Besides, the four features were manually extracted according to the researchers' experience. Because researchers did not know which features worked better for classification, it was hard to choose suitable features, thus these features might not have good discrimination ability and could be improved. In addition, the recognition time of KNN classifier depends on the number of samples. Therefore, in order to reduce the recognition time, the researchers created smaller training set for each activity by creating clusters, namely Clustered KNN, and performed classification based on the reduced set. But as the authors said, such processing also reduced the recognition accuracy. From literatures, other works [2-5] also had one or more aforementioned problems.

In this paper, we recommend deep neural networks (DNNs) for real-time activity recognition, which were widely used in speech recognition [6], computer vision [7] and natural language processing [8]. DNNs automatically learn suitable features without relying on the experience of the researchers, and then perform classification. We first collected accelerometer data using an Android smartphone, which was positioned in the right pant pocket. We considered seven activities of interest for recognition: walking, running, standing, sitting, lying, walking upstairs and walking downstairs. Then we trained DNNs offline and implemented the models on a Samsung Galaxy Note III mobile phone. We also tested the running time for the DNN models on a Samsung Galaxy Note III smartphone and conducted experiments on our own dataset to compare our method with traditional methods. The results of experiments on the dataset showed that DNNs could achieve real-time performance and got higher accuracy than traditional methods.

The remainder of this paper is organized as follows: Section II describes the related work. Section III describes the data collection process. Section IV details how to train the DNN model. Section V gives the details of the experiment design, and gives the recognition results of our method and traditional methods. Section VI summarizes our conclusion and gives the future work.

978-1-4673-7211-4/15 $31.00 © 2015 IEEE
DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.224

II. Related Work

Real-time activity recognition has been investigated for many years. Kose et al. [1] collected accelerometer data on smartphones and applied low pass filter to raw data. Four features were extracted: average, minimum, maximum and standard deviation. They classified four activities: walking, running, sitting and standing. They chose Clustered KNN as the classifier. The results of experiments of this work showed that increasing the *K* value improved the recognition accuracy, but meanwhile increased the recognition time. Specifically, when *K* was 10, the accuracy was 87% and the recognition time was 50 milliseconds, while when *K* was 50, the accuracy increased to 91%, but the recognition time increased to 300 milliseconds as well. They also compared the performance of Clustered KNN with that of Naïve Bayes. The results showed that Clustered KNN gave higher performance. Ermes et al. [2] collected accelerometer data from wireless motion bands and the three-dimensional raw data of acceleration were then sent to a HTC device via a Bluetooth link. Four features were extracted: average, variance, frequency of the highest peak in the power spectral density and spectral entropy. Six activities were classified: lying, sitting, standing, walking, running, and cycling using a bike for exercise. The classifier was decision tree. Lara et al. [3] collected sensor data from the smartphone GPS, smartphone accelerometer, and a sensor strap. The sensor strap collected the data of the heart of the user, respiration rate, three-dimensional acceleration, the amplitude of breath, the amplitude of electrocardiogram, and so on. They achieved a communication model that received raw sensor data via Bluetooth, sent the collected data or the recognized activity to the server via TCP/IP, and queried the database via HTTP. They also extracted some features, including mean, variance, standard deviation, and so on. They chose decision tree as the classifier. The results that they gave showed that the larger the window size and the number of features were, the longer the response time was. When the window size was 5s and the number of features was 4, the response time was 66.54 milliseconds. When the window size was 5s and the number of features was 19, the response time was 395.33 milliseconds. When the window size was 12s and the number of features was 9, the response time was 499.33 milliseconds. It should be noted that the response time was consumed by preprocessing, feature extraction, and classification on the server, thus the data communication time was not contained. Park et al. [4] used Support Vector Machine (SVM) for pose classification. The person was asked to hold a mobile device in an unknown location. They only required a triaxial accelerometer for data collection. The features that they extracted were the magnitudes of the low-frequency portion of the acceleration signal spectrum and a set of features derived from the gravity vector. Siirtola et al. [5] collected accelerometer data using smartphones. The recognition models were trained offline using the collected data and then were used for online testing on two operation systems: Symbian and Android. They extracted some features, including standard deviation, minimum, maximum, and so on. The decision tree was chosen as the classifier. They gave the recognition accuracy, but not the recognition time.

The works described above had two common problems: due to the recognition time limit, they could only extract a small amount of features, which might not achieve good accuracy and could be improved. Besides, these features were manually extracted, which relied on the experience of the researchers. Because researchers did not know which features worked better for classification, it was hard to choose suitable features, thus the features might not be related to the specific classification task and have poor discrimination ability, and could be improved.

In this paper, we recommend deep neural networks for real-time activity recognition. They can automatically learn suitable features without relying on the experience of the researchers, and have no limit of the number of extracted features. We also do not need to simplify the classifier to save recognition time as some works did [1]. In other words, deep neural networks overcome the defects of traditional methods.

III. Data Collection

To obtain a real-time human activity recognition system, we first need to collect accelerometer data on a smartphone. Then after the data being segmented into frames, we should train a DNN offline using these frames. After offline training, we will get the parameters of the DNN model. The parameters are the weights between every two subsequent layers of a DNN. Finally, we need to implement the model on the smartphone by transplanting these parameters onto the smartphone. Online testing is just simple matrix multiplication.

Previous datasets either contain a small amount of data for an individual user [9-12], or consist of extremely imbalanced data of different activities for a single subject [13, 14], thus could not represent the motion characteristics of a user. Nevertheless, as far as we know, in the field of computer vision and speech recognition, many datasets contain tens of thousands of samples. For example, the MNIST handwritten digit recognition dataset [15] is a widely used dataset in the field of computer vision, and it contains 60000 samples in the training set and 10000 samples in the testing set. In view of the above-mentioned facts, we decided to collect an activity recognition dataset that is high-volume and contains balanced data of different activities.

We used an android smartphone to collect the accelerometer data when the subject performed different activities. We did not consider collecting the data of gyroscope and magnetometer. The sampling rate was 100 Hz. The smartphone was positioned in the right pant pocket of the subject. We collected the data of seven activities of interest: walking, running, sitting, standing, lying, walking upstairs, and walking downstairs. For walking, we had the subject walk in the campus of Peking University under uncontrolled conditions for more than an hour. For running, the subject was asked to run in a playground for an hour. For

sitting and standing, the data of the subject was collected in the laboratory for an hour and more than an hour respectively. For lying, the data was collected by lying on the bed in the dormitory for an hour. For waking upstairs and walking downstairs, the data was collected when climbing the stairs and going down the stairs in a 10-floor building in Peking University. The subject repeated the up and down stairs process for several times and finally collected about an hour data for waking upstairs and walking downstairs respectively. The performing of these activities conformed to the living habits of the subject. So once the accelerometer data of the subject were collected, we could make use of the subject's accelerometer data to train a classification model and then implemented the model on smartphones, and then recognized his activity behaviors when being in the online application phase.

After data collection, we split the data into frames. The frame length was set to 1s, which had been proved to be enough to identify different activities [16]. Two consecutive frames overlapped by 50%. We took out about 19% of the data of every activity and combined them into a testing set and the rest into a training data. For example, for the an-hour data of the running activity, we put the top 81% data into the training set and the bottom 19% data into the testing set. For the other activities, we did in the same way. Finally, we got 42000 frames for the training set and 10100 frames for the testing set. TABLE I summarizes the total number of frames of different activities, the number of training samples of different activities and the number of testing samples of different samples. The first column represents different activities and the second column represents the number of all the samples of different activities, and the third column represents the number of the training samples of different activities, and the fourth column represents the number of the testing samples of different activities. From TABLE I, we can find that both in our training set and testing set, seven activities have balanced amount of frames.

TABLE I. THE NUBER OF TRAINING AND TESTING SAMPLES OF DIFFERENT ACTIVITIES IN OUR DATASET

Activities	Number of all samples	Number of traing samples	Number of testing samples
Walking	8000	6460	1540
Running	7200	5800	1400
Sitting	7200	5800	1400
Standing	8000	6460	1540
Lying	7200	5800	1400
Walking upstairs	7200	5800	1400
Walking downstairs	7300	5880	1420

IV. METHOD

Deep neural network is a kind of the neural network model. It has an input layer, two or more hidden layers, and an output layer. The graphical representation of a DNN model with two hidden layers is shown in Fig. 1. The nodes represent the variables and the links between the nodes represent the weight parameters. Arrows denote the information flow direction through the network. Due to the large amount of parameters, DNNs possess the ability of automatically learning suitable features from the raw data. The parameters of a DNN are usually randomly initialized and then the whole network is trained using back-propagation.

The DNNs that we choose for activity recognition are the deep belief networks (DBNs) [17], whose parameters are initialized by a generative pre-training and then are fine-tuned using back-propagation. In this part, we detail how to train deep belief networks.

A. Pre-training

The parameters of DBNs are initialized by a generative pre-training. Different from traditional deep neural networks, DBNs treat every two adjacent layers as a Restricted Boltzmann Machine (RBM). The input layer and the first hidden layer form a Gaussian-binary RBM, whose input units are real-valued data and liner with Gaussian noise and the hidden units are binary. Two consecutive hidden layers form a binary-binary RBM, in which all the units are binary and stochastic. A RBM is a fully connected, bipartite undirected graph, which consists of a visible layer V and a hidden layer H, as is shown in Fig. 2. There are no visible-visible or hidden-hidden connections. For a RBM, the visible units represent the observations and are connected to the hidden units using the undirected weighted connections. The hidden units act as feature detectors. Both V and H is a

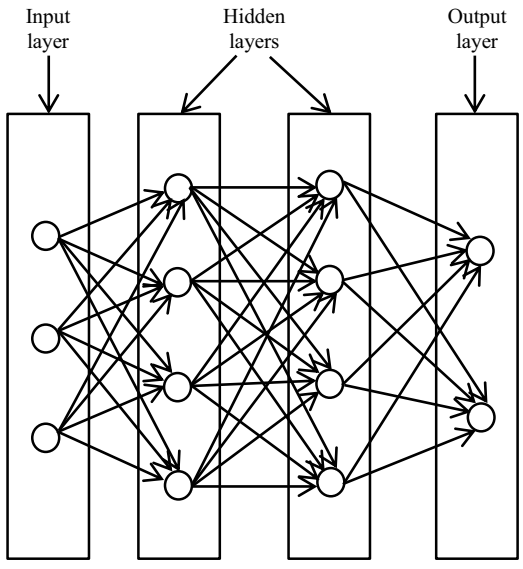

Figure 1. The graphical representation of a deep neural network

vector, representing the visible units and hidden units respectively. Again, the links between the nodes represent the weight parameters, which, however, are undirected. Suppose the variables (V, H) take values (v, h). Then the RBM is trained as follows: sampling a new state h for the hidden units based on $p(h|v)$, then sampling a new state v for the visible units based on $p(v|h)$, and then repeating the above process [18]. Noticing that a RBM is an undirected graph and the weighted connections are undirected, alternate updating v and h is feasible. Once the RBM is trained, we treat the hidden units of this RBM as the visible units of the next RBM and the next layer as the hidden units, and then train the next RBM. Once all the RBMs of a deep belief network are trained, the generative pre-training is finished.

B. Fine-tuning

Pre-training learns a generative model, which does not make use of the information of the labels, and the learned features (the hidden units) may be irrelevant for classifying different categories. So after pre-training, a "softmaxed" layer of label units is added. The number of the label units is the number of categories. Then the whole network is treated as a feed-forward, deterministic neural network. And then a discriminative fine-tuning is performed on the whole network using the back-propagation algorithm, making use of the precious information in the labels, to slightly adjust the features in each layer, which were discovered by the unsupervised pre-training before, and get the category boundaries right for better classification.

V. EXPERIMENTS

In this part, we did experiments to compare the performance of the DNN model with those of traditional methods. Besides, we implemented the DNN model on a smartphone to show that our method could achieve the real-time effect.

A. Experiment Design

To prove the high accuracy of DNNs, we compared the DNN model with a large number of traditional techniques, which took manually extracted features as inputs. We selected eight classification methods: Bayesian nets, Naïve Bayes, SVM, Logistic regression, KNN, Decision tree, Random forest, and three-layer feed-forward neural network (3-layer NN). These classifiers were more often used and could be found in [1-5, 12, 19-23]. 3-layer NN has an input layer, a hidden layer and an output layer. Due to its small amount of parameters, it still needed to extract features manually and took these features as inputs. The collected accelerometer data included three dimensions: x, y and z. We calculated the magnitude of acceleration denoted as mag:

$$mag = \sqrt{x^2 + y^2 + z^2}. \quad (1)$$

Then we extracted features for x, y, z and mag for the eight methods. We manually extracted four features: average, minimum, maximum, and standard deviation, which were used in [1]. Therefore, the dimensionality of manually extracted features is 16. Every dimension is normalized to be between 0 and 1. The eight classifiers were performed using the WEKA machine learning toolkit (Waikato Environment for Knowledge Analysis) [24, 25]. For the DNN model, the inputs were the raw accelerometer data without any preprocessing. We used the deep belief network code from the homepage of Geoffrey Hinton [26]. The training set was divided into mini-batches with each mini-batch having 100 frames. Each mini-batch contained balanced samples from all classes. This strategy improved model generation since it removed needless biases [16]. The DNN model was pre-trained using stochastic gradient decent with a mini-batch at a time. An epoch refers to passing over all the mini-batches, i.e., the entire training set. We ran 100 epochs for the Gaussian-binary RBM at a learning rate of 0.001 and ran 50 epochs for the binary-binary RBMs at a learning rate of 0.1. During pre-training, we used a weight-cost of 0.0002 and a momentum of 0.9 [27]. After pre-training, a global supervised fine-tuning on the whole network using back-propagation was performed to make the learned features by pre-training have more discriminative capacity [28]. Seven "softmaxed" output units were then added to the top layer. The units of the top layer of a DNN model represent different classes and all of them have a probability of being classified into the corresponding activity, and the one with the maximum probability is identified as the real class. For fine-tuning, we used the conjugate gradients algorithm on larger mini-batches at a time, which contained 1000 frames. And in the conjugate gradient fine-tuning, we used Carl Rasmussen's "minimize" code [29], and performed three line

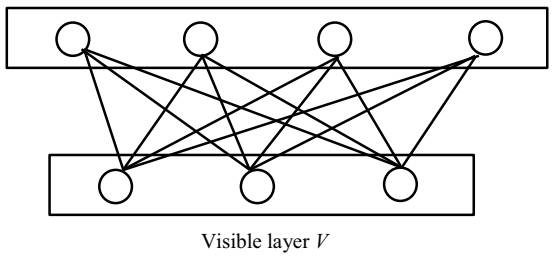

Figure 2. The undirected graph of a RBM

TABLE II. THE NOTATIONS OF DIFFERENT CLASSIFICATION METHODS

Classification Methods	Notations
Bayesian nets	BN
Naïve Bayes	NB
Support vector machine	LibSVM
Logistic regression	LR
K-nearest neighbor	KNN
Decision tree	J48
Random forest	RF
three-layer feed-forward neural network	3-layer NN
Deep neural network	DNN

searches for each mini-batch in each epoch. We performed 600 epochs of fine-tuning on the entire training set. We also adjusted the structure of the DNN model. When we adjusted the number of units of one layer, we kept the number of units of the other layers fixed. Finally, we got a better DNN model. The DNN model that had a better recognition result had a 400-500-500- 2000-7 structure. That is to say, it had 400 units in the input layer, 500 units in the first and second hidden layer, 2000 units in the third hidden layer, and 7 units in the output layer, representing 7 categories. 400 input units consist of 100 samples (1s frame length) of each dimension, including x, y, z and mag.

After offline training, we implemented the DNN model on a Samsung Galaxy Note III mobile phone for online recognition. Samsung Galaxy Note III contains a 2.3 GHz processor. The mobile phone first collected a frame of data, whose length was 1 s, containing 100 data points. Then the data were fed into the DNN model, which had been realized on the Samsung Galaxy Note III mobile phone, as the inputs, and then activity recognition were performed. Online recognition is just simple matrix multiplication. Finally, the phone outputted the recognized activity. The running time of the DNN model was taken as the recognition time. The overall framework of real-time activity recognition using the DNN model, including offline training and online application, is shown in Fig. 3.

B. Accuracy

The notations of different methods are listed in TABLE II. The classification results of different methods are presented in Fig. 4. The horizontal axis represents the error rate and the vertical axis represents different classification methods. Because both in our training set and testing set, seven activities have balanced amount of frames, taking accuracy as the evaluation criterion is advisable for comparing the performance of different methods. For the KNN classifier, we set K from 1 to 15 and the better accuracy was gotten when K was 1. Higher values of K had no significant impact on the accuracy. The results showed that compared with the eight methods that extracted features manually, the DNN model achieved the best accuracy with 1 percent higher than the second, i.e., KNN, with K to be 1. The reason that the DNN method has excellent recognition performance than traditional methods lies in that DNNs automatically learn features suitable for classification without relying on human experience or domain-specific expert knowledge.

C. Recognition Time

The result of the recognition time of the DNN model on a Samsung Galaxy Note III mobile phone is presented in TABLE III. The first column represents the structure of the DNN model and second column represents the running time. Given a frame of data, the recognition time refers to the computation time of the DNN model before it gives the recognized activity. The recognition time of the DNN model is 132.7656 milliseconds, which is very small compared with the frame length of 1s. The result of the recognition time demonstrates that deep neural networks can absolutely achieve the real-time performance when being in the online application phase. The reason that deep neural networks have small recognition time lies in that online activity recognition on smartphones for the DNN model is just simple matrix multiplication.

TABLE III. RECOGNITION TIME OF THE DNN MODEL

Structure of DNN model	Recognition time (ms)
400-500-500-2000-7	132.7656

According to the above results, we can draw the conclusion that the DNN model not only completely achieves the real-time goal, but also has excellent recognition performance.

VI. CONCLUSION AND FUTURE WORK

Previous works of real-time activity recognition needed to extract features manually and only extracted a small amount of features, making these features have poor discrimination ability. In this paper, we recommend deep neural networks for real-time activity recognition, which can automatically learn suitable features without relying on the experience of the researchers, and have no limit of the number of extracted features. We also do not need to simplify the classifier to save the recognition time. We first collected an activity recognition dataset. Our dataset has two characteristics comparing to previous available datasets. One is that it contains a large amount of data for an individual user. The other is that the data of different activities for a single subject is balanced. Then the collected dataset was divided into two parts, i.e., training set and testing set. The DNN model was offline trained using the training set, including a process of pre-training and a process of fine-tuning. We did experiments on our own dataset to compare

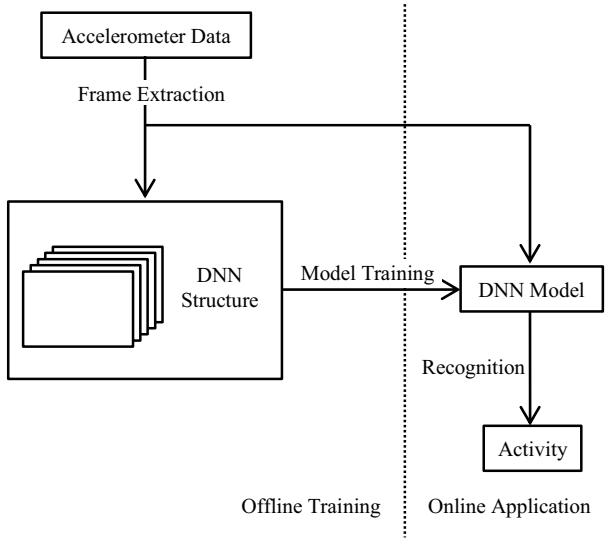

Figure 3. The overall framework of real-time activity recognition using the DNN model

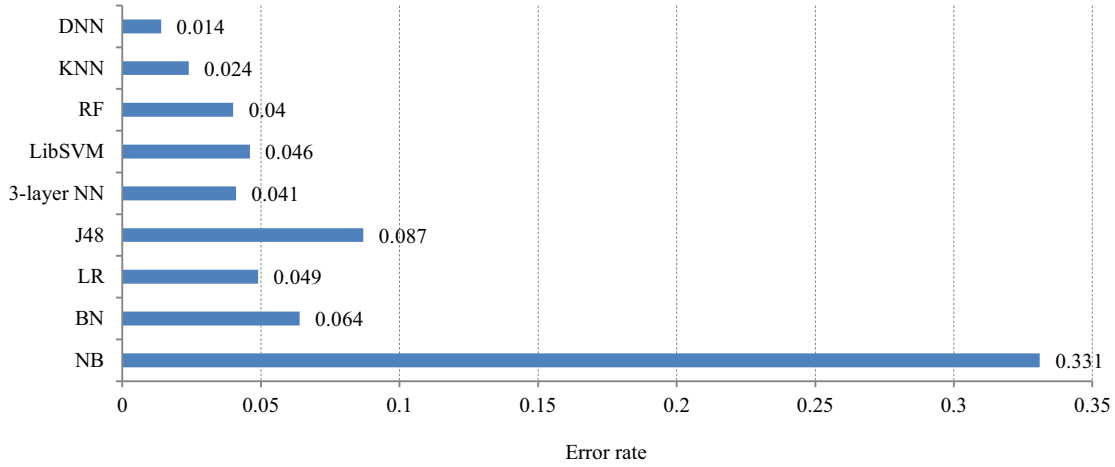

Figure 4. The classification results of different methods on our dataset

the performance of our method with those of the selected eight traditional methods. The classification results showed that the DNN method outperformed traditional methods. We also implemented the DNN model on a Samsung Galaxy Note III mobile phone and tested its online recognition time. The result of recognition time showed that the DNN method absolutely achieved real-time performance.

For future work, we will continue to collect the accelerometer data for other activities, such as riding a bike. Besides, we will collect the motion data for other subjects and then make our dataset public for researchers to compare different methods. In addition, we will use other methods, which also automatically learn features, such as recurrent neural network and convolutional neural network, to try to improve activity recognition performance.

ACKNOWLEDGMENT

The work is supported in part by National Basic Research Program (973 Program) of China (No. 2013CB329304), the National Natural Science Foundation of China (No. 90920302, No. 91120001, No.61121002), a "Twelfth Five-Year" National Science & Technology Support Program of China (No. 2012BAI12B01) and the Key Program of National Social Science Foundation of China (No. 12&ZD119).

REFERENCES

[1] M. Kose, O. D. Incel, and C. Ersoy, "Online human activity recognition on smart phones," in *Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*, pp. 11-15, 2012.

[2] M. Ermes, J. Parkka, and L. Cluitmans, "Advancing from offline to online activity recognition with wearable sensors," in *30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 4451-4454, 2008.

[3] O. D. Lara and M. A. Labrador, "A mobile platform for real-time human activity recognition," in *IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 667-671, 2012.

[4] J. G. Park, A. Patel, D. Curtis, S. Teller, and J. Ledlie, "Online pose classification and walking speed estimation using handheld devices," in *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 113-122, 2012.

[5] P. Siirtola and J. Roning, "Ready-to-use activity recognition for smartphones," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pp. 59-64, 2013.

[6] G. E. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," In *IEEE Signal Processing Magazine*, vol. 29, pp. 82-97, 2012.

[7] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504-507, 2006.

[8] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, pp. 160-167, 2008.

[9] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *Ambient assisted living and home care*, pp. 216-223, 2012.

[10] M. Zhang and A. A. Sawchuk, "Usc-had: a daily activity dataset for ubiquitous activity recognition using wearable sensors," In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 1036-1043, 2012.

[11] C. McCall, K. K. Reddy, and M. Shah, "Macro-class Selection for Hierarchical k-NN Classification of Inertial Sensor Data," in *PECCS*, pp. 106-114, 2012.

[12] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, and P. J. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," *Sensors*, vol. 14, pp. 10146-10176, 2014.

[13] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometer," *ACM SIGKDD Explorations Newsletter*, vol. 12, pp. 74-82, 2011.

[14] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. D. R. Millán, and D. Roggen, "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, pp. 2033-2042, 2013.

[15] The MNIST data set is available at http://yann.lecun.com/exdb/mnist/index.html.

[16] T. Plötz, N. Y. Hammerla, and P. Olivier, "Feature learning for activity recognition in ubiquitous computing," *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, vol. 22, pp. 1729-1734, 2011.

[17] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, pp. 1527-1554, 2006.

[18] A. Fischer and C. Igel, "An introduction to restricted Boltzmann machines," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 14-36, 2012.

[19] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive Computing, IEEE*, pp. 1–17, 2004.

[20] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity Recognition from Accelerometer Data," *Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence*, vol. 5, pp. 1541-1546, 2005.

[21] J. Parkka, M. Ermes, P. Korpipaa, J. Mantyjarvi, J. Peltola, and I. Korhonen, "Activity Classification Using Realistic Data From Wearable Sensors," *IEEE Transactions on Information Technology in Biomedicine,* vol. 10, pp. 119-128, 2006.

[22] W. Wu, S. Dasgupta, E. E. Ramirez, C. Peterson, and G. J. Norman, "Classification accuracies of physical activities using smartphone motion sensors," *Journal of medical Internet research*, vol. 14, no. 5, e130, 2012.

[23] O. D. Incel, M. Kose, and C. Ersoy, "A review and taxonomy of activity recognition on mobile phones," *BioNanoScience*, vol. 3, pp. 145-171, 2013.

[24] S. R. Garner, "Weka: The waikato environment for knowledge analysis," in *Proceedings of the New Zealand computer science research students conference*, pp. 57-64, 1995.

[25] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, pp. 10-18, 2009.

[26] The MATLAB code is available at http://www.cs.Toronto.edu/~hinton/MatlabForSciencePaper.html.

[27] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," *Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science*, vol. 7700, pp. 599-619, 2012.

[28] "UCL Tutorial on: Deep Belief Nets," www.cs.toronto.edu/~hinton/ucltutorial.pdf.

[29] Carl Rasmussen's "minimize" code is available at http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/