Human Activity Recognition From Accelerometer Data Using Convolutional Neural Network

Song-Mi Lee Sang Min Yoon Heeryon Cho
Human Computer Interaction Lab
Kookmin University
Seoul, South Korea
{20123401, smyoon, heeryon}@kookmin.ac.kr

***Abstract*—We propose a one-dimensional (1D) Convolutional Neural Network (CNN)-based method for recognizing human activity using triaxial accelerometer data collected from users' smartphones. The three human activity data, walking, running, and staying still, are gathered using smartphone accelerometer sensor. The *x*, *y*, and *z* acceleration data are transformed into a vector magnitude data and used as the input for learning the 1D CNN. The ternary activity recognition performance of our 1D CNN-based method which showed 92.71% accuracy outperformed the baseline random forest approach of 89.10%.**

Keywords—human activity recognition; convolutional neural network; 3D accelerometer data; random forest

I. Introduction

The widespread usage of portable and wearable smart devices such as smartphones and smartwatches has enabled the easy gathering of human activity data using various device-embedded sensors. Prior to the popularization of smart devices, human activity recognition was performed by attaching multiple sensors to the user's body which was cumbersome [1]. However, with the proliferation of affordable smart devices, the gathering of user's activity information have become easy through the device-embedded sensors. Various methods for detecting human activity using smartphone accelerometer sensors have been proposed [2].

One of the popular methods for recognizing human activity is through the Google Activity Recognition API [3]. Google Activity API can detect user's activities such as riding a bicycle, running, walking, staying still, etc. using the sensor data collected from the user's smart device's sensors. We gathered three activity data, running, walking, and staying still, using the Google Activity API and compared the API's recognition results to the ground truth labels. We found that many of the activities returned by the API were recognized as 'unknown' or 'tilting' exhibiting low recognition performance. Such low performance is presumed to be due to the API's inability to take into account the differences in the smart device's placement (e.g., the user carries the device in her hand vs. inside her bag), the user's individual differences in gait and travel speed, and the inability to track the changes in user's activities (e.g., user begins to walk after staying still for a while).

In order to address the variability of raw human activity data in human activity recognition, we present a more robust one-dimensional (1D) Convolutional Neural Network (CNN)-based method that utilizes vector magnitude accelerometer data which reduces the possible rotational interference present in the raw data. We show the effectiveness of our method by comparing our method's performance to the baseline random forest approach. In order to carry out the evaluation experiment, we collected the three kinds of human activity data, i.e., walking, running, and staying still, in the form of triaxial (3D) accelerometer data using the smartphone accelerometer sensor.

The rest of the paper is organized as follows. We review the existing works on CNN-based human activity recognition in Section II and explain our 1D CNN-based human activity recognition method in Section III. The evaluation experiment and its result is presented in Section IV, and the conclusion is given in Section V.

II. Related Work

In recent years, several CNN-based human activity recognition methods have been proposed. Jiang and Yin proposed a method of constructing a novel activity image for Deep CNN (DCNN) using the gyroscope, total acceleration and linear acceleration signals [4]. The DCNN built from the activity image showed the recognition performance of 97.59%, 97.83%, and 99.93%, on the benchmark UCI, USC, and SHO dataset, respectively, which was higher than the baseline SVM and feature selection method. The computational cost of their DCNN was also smaller than SVM.

Alsheikh et al. tested the activity recognition performance of Deep Belief Networks (DBNs) using different parameter settings [5]. They also proposed a hybrid deep learning and hidden Markov model (DL-HMM) approach for sequential activity recognition. Instead of the raw acceleration data, they used spectrogram signal of the triaxial accelerometer data to learn the deep activity recognition models. They found that deep models with more layers outperform the shallow models, and that overcomplete representations (that is, the number of neurons at each layer is larger than the input length) are more advantageous. The accuracy of the tuned DBN was 98.23%, 91.5%, and 89.38% on the WISDM, Daphnet, and Skoda benchmark datasets, respectively.

This research was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No. R0115-15-1009, An Interaction Platform for Learning Foreign Languages with Tangible Reality Based on Experiential Awareness), and also by the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2016R1D1A1B04932889, 3D Reconstruction of High-Resolution Optical Microscopy Software Development).

Hammerla, Halloran, and Plotz explored the three types of deep learning models for activity recognition, namely, Deep feed-forward networks (DNN), convolutional networks (CNN), and recurrent networks (RNN) on the three benchmark datasets, the Opportunity, PAMAP2, and Daphnet Gait dataset [6]. These datasets collected human activity data using various wearable sensors. They proposed a novel regularization approach for recurrent methods. They found that the regular DNNs require a significant parameter tuning and that for prolonged activities such as walking and running, CNN is recommended.

Ronao and Cho used the accelerometer and gyroscope triaxial sensor data to perform 6-axes, 1D convolution for constructing a deep CNN [7]. Their network achieved 94.79% activity recognition accuracy on raw sensor data and 95.57% accuracy with additional Fast Fourier Transform (FFT) information on the sensor data.

Yang, Nguyen, San, Li, and Krishnaswamy utilized multi-channel time series data to recognize the user's activity and hand gestures [8]. They introduced temporal convolution and pooling approach during the CNN learning process to improve the recognition performance. Their CNN showed better results than the SVM with radial basis function kernel and deep belief network on Opportunity and Hand Gesture datasets.

In comparison to the existing approaches which employ multiple, varied sensor data (accelerometer, gyroscope, etc.) [4,6,7,8], our method uses only the triaxial accelerometer data (*x*, *y*, and *z* component signals) collected from a single accelerometer sensor. Our approach is also different to [5] in that we calculate the vector magnitude of the signal to remove possible rotational interference present in the *x*, *y*, and *z* acceleration signals.

III. Proposed Method

As shown in Fig. 1, we gathered *x*, *y*, and *z* acceleration signals and transformed them into vector magnitude data, and used the vector magnitude data to construct a 1D CNN for ternary activity classification.

A. Data Preprocessing

Data Gathering: Five graduate students were instructed to record the three activities, i.e., walking, running, and staying still, using the smartphones provided by our research team. Five Nexus 6P smartphones made by Huawei with our accelerometer data gathering application installed were used for data gathering. The students carried the smartphones in various positions (e.g., held in hand, placed in a pocket, carried in a bag/knapsack, etc.) when recording the activity data. The accelerometer data were collected via the accelerometer sensor embedded in the smartphones through our data gathering application. The acceleration values generated from the sensor's *x*, *y*, and *z*-axis were recorded at 1Hz frequency (i.e., one sample per second) along with the activity labels per given timestamp (hand labeled by the students). A total of 2,377 seconds of running, 3,588 seconds of walking, and 3,934 seconds of staying still time series activity data were collected, which added up to 9,899 seconds of human activity data.

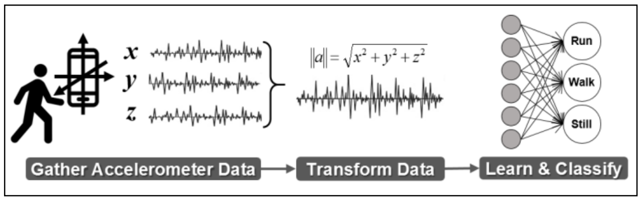

Fig. 1. Overall flow of our 1D CNN-based acitivity recognition

Data Transformation: Because the accelerometer sensor measures the magnitude of the *x*, *y*, and *z* components' acceleration, the changes in the *x*, *y*, and *z* values can be tied to the user's movement if the user is carrying the smartphone. However, since the triaxial acceleration data include a rotation component, such rotation component may interfere with the precise recognition of the user's movement. In order to keep the rotational interference to a minimum, we transform the raw *x*, *y*, and *z* acceleration data into vector magnitude data. The vector magnitude data can be obtained by calculating the Euclidean norm of *x*, *y*, and *z* values as follows:

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

Equation (1) transforms the *x*, *y*, and *z* component values to a single representative value while reducing the error possibly generated by the rotation component. The vector magnitude time series data are then divided into three activity data based on the ground truth labels. For each activity data, two-size feature vectors, ten and twenty second vectors, are created by slicing the time series data using the ten and twenty second window while sliding the window repeatedly by one second.

B. Network Architecture

The network architecture of our 1D CNN consists of one convolutional layer, one max-pooling layer, one fully-connected layer, and one softmax layer that outputs the probability of each of the three activities. Figure 2 shows the 1D CNN architecture of our proposed method.

a. Input: A fixed time-length accelerometer data in the form of vector magnitude data explained above was used as input. Specifically, we constructed two kinds of input vectors, ten and twenty-second vectors. The input vector $\boldsymbol{v}$ has N dimensions ($\boldsymbol{v} \in R^N$), and in our experiment N=10, 20. Figure 1 *a. input* shows the ten-second input vector of a given training data size. Each row of the input indicates a ten-second vector.

b. Convolution: The convolution operations were performed using the three window sizes, 3, 4, and 5, with stride size of one for all window sizes yielding a feature map (or in our case, vector) size of 8, 7, and 6. Note that since our input data are a vector instead of a matrix, the sliding window was moved only to the right for each input data. A total of 128 filters were used on each input vector to create the feature vectors for each window size. Figure 1 *b. convolution* exhibits the resulting three sets of feature vectors generated from the convolution operations.

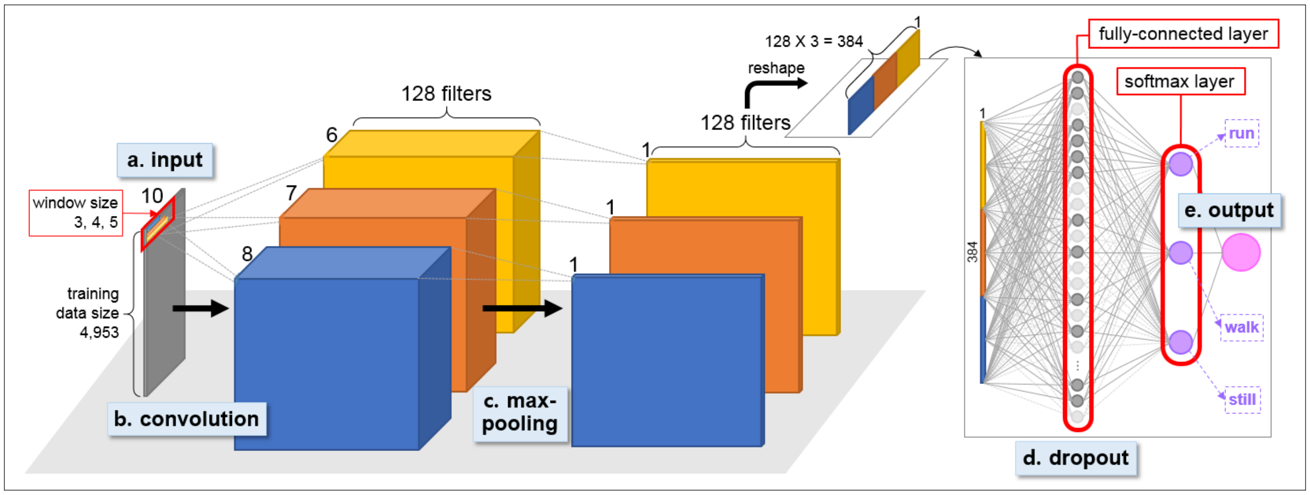

Fig. 2. The network architcture of our 1D CNN-based activity recognition method[1]

c. **Max-pooling:** For each feature vector of a given window size and filter type, 1-max-pooling was performed to select the largest feature value. Figure 1 *c. max-pooling* shows the max-pooled result of the three sets of feature vectors. The max-pooled result was then concatenated as shown in Fig. 1 *reshape* to create a long 384-length feature vector for each input data. Each long feature vector contains the most salient feature extracted using the 128 filter × 3 window size combinations.

d. **Dropout:** The convolved and max-pooled feature vectors were then placed as the input to the fully-connected neural network with dropout applied. The dropout was applied to prevent the neural network from overfitting. Figure 1 *d. dropout* shows the fully-connected neural network with dropout[1]. We set the percentage of the dropout to 0.5 in our evaluation experiment to be explained later.

e. **Output:** The softmax layer was placed as an output layer of the fully-connected layer as shown in Fig. 1 *softmax layer*. Each unit (or node) in the softmax layer calculates the probability of each activity (i.e., run, walk, and still) given the long feature vector. The activity with the highest probability is then determined as the predicted (or recognized) activity and the activity label is outputted to the final node (in pink) as shown in Fig. 1 *e. output*.

IV. Experiments

In our evaluation experiment, two 1D CNN models were constructed using the two types of input vectors of varying length, namely, ten-second and twenty-second vectors. We describe the experimental setup and the result of the evaluation experiment and compare our method to the baseline random forest activity recognition in this section.

[1] Note that the fully-connected neural network depicted in Fig. 1 *d. dropout* shows an abstracted version of the neural network. The dimension of the input-hidden-output layer is 384×384×3.

A. Dataset

The two types of input vectors, ten-second and twenty-second vectors, were prepared as train and test data for the experiment. We indicate the former as Feature$_{10}$ and the latter as Feature$_{20}$. Table 1 summarizes the size of each train and test data. The dataset was preprocessed to generate vector magnitude data as described in Section II Data Transformation. The three activity labels, run, walk, and still, each constituted exactly one third of the train and test data for both Feature$_{10}$ and Feature$_{20}$ data.

B. Baseline Method and Evaluation Metrics

Our 1D CNN method was compared to the baseline random forest method for evaluation. Random forest is known to be a good classification method for recognizing human activities [9]. We used MATLAB implementation of the random forest. For the evaluation metrics, we calculated precision and recall for each activity. We also calculated the overall accuracy of human activity recognition for 1D CNN and random forest methods.

C. Network Parameters and Training Methodology

We used the open source software TensorFlow [10] to implement our 1D CNN. The following parameters were used for training our neural network for both Feature$_{10}$ and Feature$_{20}$ data: *convolution window size* = 3, 4, and 5; *window stride size* = 1; *number of convolution filters* = 128; *percentage of dropout* = 50%; *training batch size* = 64; and *training epoch size* = 200. For the loss function, we computed the sigmoid cross entropy given logits. For optimization, we used the Adaptive Moment Estimation (ADAM) optimization method [11].

TABLE I. Train & Test Data

Data	Train	Test	Total
Feature$_{10}$	4,953	2,073	7,026
Feature$_{20}$	4,923	1,743	6,666

D. Results

Table II and III compares the human activity recognition performance between the baseline random forest method and our 1D CNN method using Feature$_{10}$ and Feature$_{20}$ data respectively. The diagonal entries in bold show the number of correctly classified test data. The bottom right corner of each confusion matrix displays the overall accuracy of each method for each feature data. We see that for both Feature$_{10}$ and Feature$_{20}$, our 1D CNN (91.32% and 92.71%) outperforms the random forest approach (85.72% and 89.10%). The highest accuracy is achieved when 1D CNN is constructed using Feature$_{20}$ (92.71%). Note that the random forest also performs better when Feature$_{20}$ data are used. We presume that this is because Feature$_{20}$ contains more activity data. Overall, we see that the activity recognition performance is improved if the input vector dimension is increased, and because of this, one might hastily think that the recognition performance may improve if longer input vectors (e.g., thirty-second vector) were used in activity recognition.

However, if we compare closely the Tables II and III, we see that both the precision and recall of the 'walk' activity is the lowest for both approaches in Table II whereas in Table III, the 'walk' activity shows the lowest precision but the highest recall for both approaches. We surmise that the activity signal generated for 'walk' tends to contain ambiguous signals that can either be interpreted as 'run' and 'still', and this may be causing the equivocal judgement leading to the low recognition performance in Table II. On the other hand, in the case of Table III, since the input vector length is doubled (Feature$_{20}$), all three signals, run, walk, and still, are more likely to contain similar-looking signals of each other as a result, causing the classifier to learn the 'mixed-looking' signals to be the predominant signal. Consequently, 'walk' is predicted more often than other activities leading to the high recall rate for 'walk'.

TABLE II. RANDOM FOREST VS. OUR 1D CNN USING FEATURE$_{10}$

Random Forest		**Predicted Class**			**Recall**
	Activity	***Run***	***Walk***	***Still***	
Actual Class	***Run***	**587**	100	4	84.95%
	Walk	65	**561**	65	81.19%
	Still	0	62	**629**	91.03%
Precision		90.03%	77.59%	90.11%	**85.72%**
1D CNN		**Predicted Class**			**Recall**
	Activity	***Run***	***Walk***	***Still***	
Actual Class	***Run***	**605**	84	2	87.55%
	Walk	68	**597**	26	86.40%
	Still	0	0	**691**	100%
Precision		89.90%	87.67%	96.11%	**91.32%**

TABLE III. RANDOM FOREST VS. OUR 1D CNN USING FEATURE$_{20}$

Random Forest		**Predicted Class**			**Recall**
	Activity	***Run***	***Walk***	***Still***	
Actual Class	***Run***	**522**	57	2	89.85%
	Walk	33	**530**	18	91.22%
	Still	0	80	**501**	86.23%
Precision		94.05%	79.46%	96.16%	**89.10%**
1D CNN		**Predicted Class**			**Recall**
	Activity	***Run***	***Walk***	***Still***	
Actual Class	***Run***	**527**	54	0	90.71%
	Walk	18	**560**	3	96.39%
	Still	0	52	**529**	91.05%
Precision		96.70%	84.08%	99.44%	**92.71%**

V. CONCLUSION

We presented a 1D CNN-based human activity recognition method that uses triaxial accelerometer data gathered from the user's smartphone. Our method outperformed the baseline random forest method in ternary human activity classification, and exhibited the best classification accuracy when longer-length (i.e., Feature$_{20}$) accelerometer data was used for learning the neural network. We found that the dimension of the input vector affects the activity recognition performance, and that figuring out a way to disambiguate the 'walk' signal in particular will likely lead to the improvement in the activity recognition performance.

REFERENCES

[1] A.M. Khan, Y.-K. Lee, S.Y. Lee, and T.-S. Kim, "A triaxial accelerometer-based physical-activity recognition via augmented-signal reatures and a hierarchical recognizer", IEEE Trans. on Information Technology in Biomedicine, vol. 14, pp. 1166-1172, 2010.

[2] S. Dernbach, B. Das, N.C. Krishnan, B.L. Thomas, and D.J. Cook, "Simple and complex activity recognition through smart phones," in Proc. of the Int'l Conf. on Intelligent Environments, pp. 214-221, 2012.

[3] Google APIs for Android: ActivityRecognitionAPI, https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi (retrieved: Dec. 14, 2016)

[4] W. Jiang and Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," in Proc. of the 23rd ACM Int'l Conf. on Multimedia (MM '15). ACM, New York, NY, USA, pp. 1307-1310, 2015.

[5] M.A. Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, H.-P. Tan, "Deep activity recognition models with triaxial accelerometers," in Proc. of the AAAI Workshops, Phoenix, Az, USA, February 2016.

[6] N.Y. Hammerla, S. Halloran, T. Plotz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," in Proc. of the 25th Int'l Joint Conf. on Artificial Intelligence (IJCAI '16), 2016.

[7] C.A. Ronao and S.-B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," Expert Syst. Appl. vol. 59, C, pp. 235-244, October 2016.

[8] J.B. Yang, M.N. Nguyen, P.P. San, X.L. Li, S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in Proc. of the 24th Int'l Joint Conf. on Artificial Intelligence (IJCAI '15), 2015.

[9] P. Casale, O. Pujol, and P. Radeva, "Human activity recognition from accelerometer data using a wearable device," in Proc. of the 5th Iberian Conf. on Pattern Recognition and Image Analysis (IbPRIA '11), Springer-Verlag, Berlin, Heidelberg, pp. 289-296, 2011.

[10] Google TensorFlow, https://www.tensorflow.org/ (retrieved: Dec. 13, 2016)

[11] D.P. Kingma and J.A. Ba, "Adam: a method for stochastic optimization," arXiv:1412.6980, December 2014.