Deep Convolutional Neural Networks for Human Activity Recognition with Smartphone Sensors

Charissa Ann Ronao and Sung-Bae Cho(✉)

Department of Computer Science, Yonsei University,
Seoul, 120-749, South Korea
cvronao@sclab.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

Abstract. Human activity recognition (HAR) using smartphone sensors utilize time-series, multivariate data to detect activities. Time-series data have inherent local dependency characteristics. Moreover, activities tend to be hierarchical and translation invariant in nature. Consequently, convolutional neural networks (convnet) exploit these characteristics, which make it appropriate in dealing with time-series sensor data. In this paper, we propose an architecture of convnets with sensor data gathered from smartphone sensors to recognize activities. Experiments show that increasing the number of convolutional layers increases performance, but the complexity of the derived features decreases with every additional layer. Moreover, preserving the information passed from layer to layer is more important, as opposed to blindly increasing the hyperparameters to improve performance. The convnet structure can also benefit from a wider filter size and lower pooling size setting. Lastly, we show that convnet outperforms all the other state-of-the-art techniques in HAR, especially SVM, which achieved the previous best result for the data set.

Keywords: Human activity recognition · Deep learning · Convolutional neural network · Smartphone · Sensors

1 Introduction

Human activity recognition (HAR) is a classification task that makes use of time-series data from devices such as accelerometers and gyroscopes (as seen in Fig. 1), preprocess these signals, extract relevant and discriminative features from them, and finally, recognize activities by using a classifier. Especially those gathered from sensors, time-series data have a strong 1D structure in that they are very highly correlated to temporally nearby local readings [1]. Moreover, considering that people perform activities with different poses and styles, and a complex activity can be decomposed into more basic movements, time-series sensor data have inherent translation and hierarchical characteristics [2]. Therefore, it is vital to utilize a classifier that takes into consideration of these intrinsic properties of time-series sensor signals.

Deep learning, and in particular, convolutional neural networks (convnet), has been gaining a lot of attention in recent years due to its excellent performance in fields such as image and speech. In the same way, the ability of convnets to exploit the local dependency and translation equivariance of data, together with its hierarchical feature extraction mechanism, is what makes it very suitable for use with time-series sensor signals [3]. In this paper, we exploit convnets to recognize activities using time-series data from smartphone sensors, investigate the effect of varying its architecture, and compare its performance with other state-of-the-art classifiers in the HAR domain.

This paper is organized as follows: we review the related work in Sect. 2, followed by a presentation of convnets with HAR and time-series sensor signals in Sect. 3. Section 4 presents and examines the experimental results, and we draw our conclusion in Sect. 5.

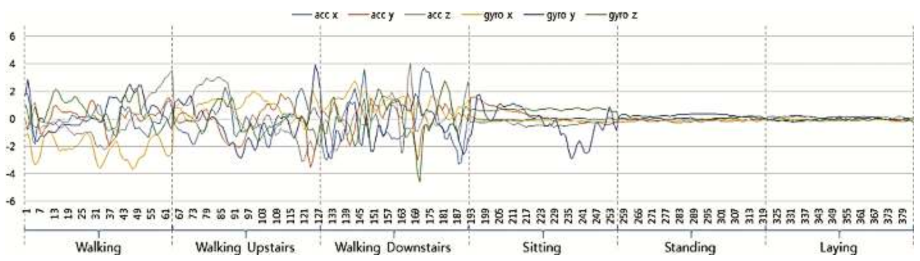

Fig. 1. One vector of accelerometer and gyroscope sensor data for every activity

2 Related Works

One of the most cited papers in HAR is by Bao and Intille, which proved that signals from accelerometers, especially those placed on the thigh, are very useful in recognizing different activities [4]. Years later, accelerometers in smartphones were utilized by Kwapisz et al. [5]. They placed the smartphone in the user's pocket and classified six different activities using classifiers such as J48 decision trees (DT), multilayer perceptrons (MLP), among others. In subsequent works, the accelerometer and gyroscope were found to be the lead sensors able to effectively recognize human activities [6]. Also, classifiers used in these latter works include naïve Bayes (NB) and k-nearest neighbors (KNN). Furthermore, Anguita et al. used a multi-class SVM to classify six different activities using 561 features extracted from both the accelerometer and gyroscope [7]. All of these classifiers, however, exhibited low performance in differentiating very similar activities such as walking upstairs and walking downstairs. We show that convnets are able to overcome this problem by exploiting the locally dependent and hierarchical characteristics of time-series sensor signals.

Among the deep learning techniques used with HAR and sensors, restricted Boltzmann machines (RBM) and sparse auto-encoders were the most common [8, 9]. However, both approaches are fully-connected neural networks which do not capture the local dependencies of time-series signals [1]. Convnets were finally applied to human activity recognition using sensor signals in [10]. However, this work only made use of a one-layered convnet architecture, which in turn did not exploit the hierarchical physiognomy of activities.

3 Human Activity Recognition with Convolutional Neural Networks

Human activities have inherent translation characteristics in that different people perform the same kind of activity in different ways, and that a fragment of an activity can manifest at different points in time [2]. Activities are also hierarchical in a sense that complex activities are composed of basic actions or movements prerequisite to the activity itself. Moreover, when using sensor signals to classify activities, it is important to take into account the temporal dependence of nearby readings [1]. Convolutional neural networks (convnet) exploit these data characteristics with its convolutional layer, which computes a mixture of nearby sensor readings, and pooling layer, which makes the representation invariant to small translations of the input [3]. A simple convnet architecture is illustrated in Fig. 2.

3.1 Convolutional Neural Networks

Convolutional neural networks perform convolution instead of matrix multiplication (as with fully-connected neural networks). Let $x_i^0 = \left[x_1, \ldots, x_N\right]$ as the accelerometer and gyroscope sensor data input vector, where N is the number of values per window. The output of the convolutional layer is:

$$c_i^{l,j} = \sigma\left(b_j + \sum\nolimits_{m=1}^{M} w_m^j x_{i+m-1}^{0,j}\right), \tag{1}$$

where l is the layer index, σ is the activation function, b_j is the bias term for the j th feature map, M is the kernel/filter size, and w_m^j is the weight for the j th feature map and m th filter index. A summary statistic of nearby outputs are derived from $c_i^{l,j}$ by the pooling layer. The pooling operation used in this paper, max-pooling, is characterized by outputting the maximum value among a set of nearby inputs, given by

$$p_i^{l,j} = \max_{r \in R}(c_{i \times T + r}^{l,j}), \tag{2}$$

where R is the pooling size, and T is the pooling stride. Several convolutional and pooling layers can be stacked on top of one another to form a deep neural network architecture. These layers act as a hierarchical feature extractor; they extricate discriminative and informative representations with respect to the data, with basic to more complex features manifesting from bottom to top.

A simple softmax classifier is utilized to recognize activities, which is placed at the topmost layer. Features from the stacked convolutional and pooling layers are aligned/flattened to form feature vectors $p^l = \left[p_1, \ldots, p_I\right]$, where I is the number of units in the last pooling layer, as input to the softmax classifier:

$$P\left(c|p\right) = \operatorname{argmax}_{c \in C} \frac{\exp(p^{L-1} w^L + b^L)}{\sum_{k=1}^{N_C} \exp(p^{L-1} w_k)}, \tag{3}$$

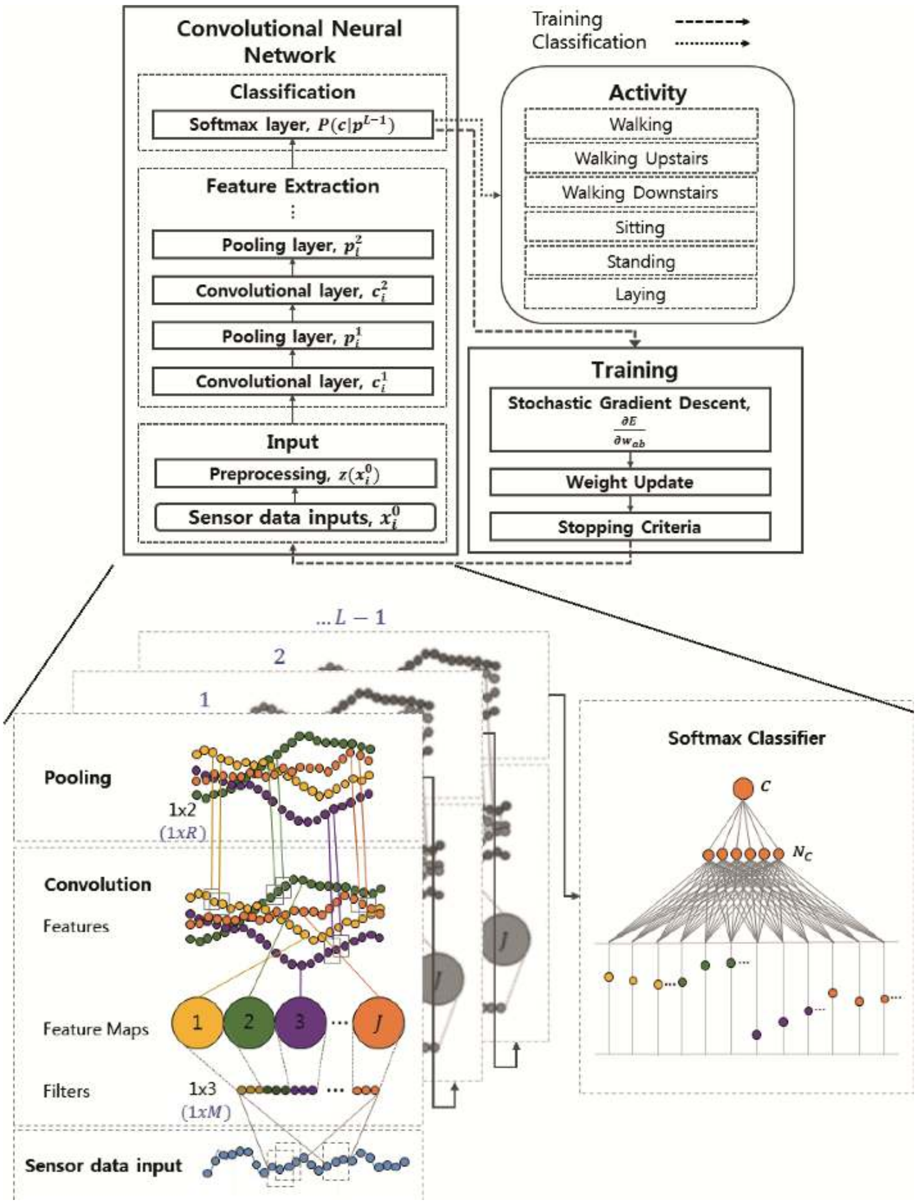

Fig. 2. Convolutional neural network architecture and training procedure

where c is the activity class, L is the last layer index, and N_C is the total number of activity classes.

Forward propagation is performed using Eqs. (1)-(3), which give us the error values of the network. Weight update and error cost minimization through training is done by stochastic gradient descent (SGD) on minibatches of sensor train data examples.

Backpropagation to adjust weights is done by computing the gradient of the convolutional weights:

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=0}^{N-M-1} \frac{\partial E}{\partial x_{ij}^{l}} y_{(i+a)}^{l-1}, \tag{4}$$

where E is the error/cost function, $y_{(i+a)}^{l-1}$ is the nonlinear mapping function equal to $\sigma\left(x_{(i+a)}^{l-1}\right) + b^{l-1}$, and deltas $\frac{\partial E}{\partial x_{ij}^{l}}$ are equal to $\frac{\partial E}{\partial y_{ij}^{l}}\sigma'\left(x_{ij}^{l}\right)$. The forward and back propagation procedure is repeated until a stopping criterion is met, e.g., if a maximum number of epochs is reached, among others.

3.2 Convolutional Neural Network Architecture and Hyperparameters

Based on the equations above, we can clearly see that there is a large number of possible combination of settings for the convnet hyperparameters, resulting in different architecture configurations. To observe and assess the effects of varying the values of these hyperparameters to the performance of the network with respect to HAR sensor data, we incorporated greedy-wise tuning starting from the number of layers L (one-layer, L_1; two-layer, L_2; and three-layer, L_3), the number of feature maps J, the size of the convolutional filter M, and the pooling size R. We varied the number of layers from 1 to 3, the number of feature maps from 10 to 200 in intervals of 10 (the same number for all layers [13]), the filter size from 1x3 to 1x15, and pooling size from 1x2 to 1x15.

4 Experiments

The publicly-available HAR smartphone dataset from the UCI repository has been utilized for all our experiments. This dataset contains accelerometer and gyroscope data from 30 subjects performing 6 different activities, namely: walking, walking upstairs, walking downstairs, sitting, standing, and laying. Data from random 21 subjects were set aside for training and the remaining data from 9 subjects, for testing. The raw accelerometer and gyroscope xyz signals were standardized to have a mean of zero (subtracted by the mean and divided by the standard deviation), resulting in a vector of 128 z-score values for every activity example. This means that we perform 6-channel (acc and gyro xyz axes), 1D convolution on the input.

For every run, padding is used to perform ‘full’ convolution with the inputs in every layer, and ReLU activation function and max-pooling operation are used. We set the learning rate to 0.01, gradually increase momentum from 0.5 to 0.99, the weight decay to 0.00005, and maximum epochs to 5000, with an early stopping criterion. The model with the best score on the validation set is saved during the run [13].

Figure 3 shows the results of every run with increasing L and J. On validation data, it can be seen that there is a gradual increase in the performance gap after adding an additional layer. On the other hand, on test data, it can be observed that there is a noticeably bigger jump in performance with the addition of the second layer than with the addition of the third layer. This means that indeed, second layer features are much more complex than first layer ones, but difference in complexity between the second layer and the third layer features are not that great. Yet, we cannot deny that adding a third layer to the network still improves performance. We have also tried to add a fourth layer, but it didn't prove to be beneficial as it just lowered the performance.

The configuration that achieved the best accuracy on the test set is carried over for tuning the filter and pooling sizes, as seen in Fig. 4. All through the run, the best performance convnet would be of an L_3 architecture. With $J = 200$, $M = 11$, and $R = 2$, convnet achieved a performance of 92.60 % on test data. This shows that blindly increasing the hyperparameters as large as we can does not necessarily translate to better performance. Instead, it is more important to preserve the information passed from input to the convolutional layers; the product of the number of features and the number of values in the input should be roughly constant, or ensured to be maintained, with the addition of each layer. This is seen with the best J configurations, with $J(L_1) = 120$ and $J(L_2) = 130$ very close to the number of points in an activity example ($N = 128$), and the jump to $J(L_3) = 200$ covering for the addition of three convolutional layers from the input. Also, high performance were seen with filter sizes 1x9 to 1x14, a span of 0.18 to 0.28 s, indicating that there tends to be a wider correlation between nearby time readings that should be exploited (as compared to 1x3 (0.06 s), where only immediate neighboring readings are considered).

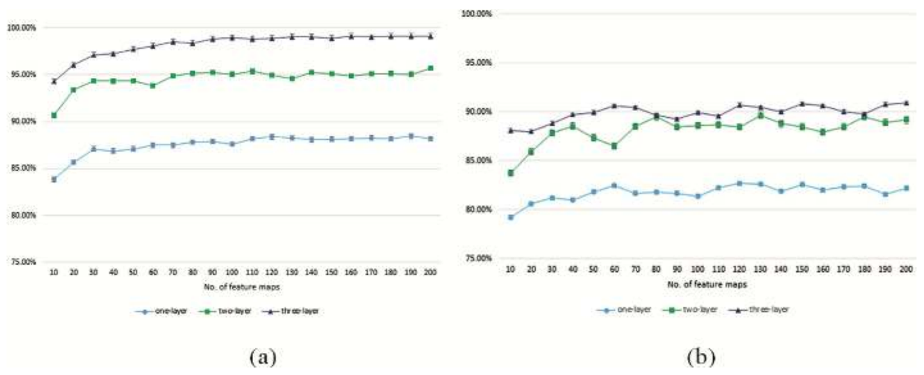

Fig. 3. Performance of different convnet architectures with increasing number of layers and feature maps on (a) validation data and (b) test data

Incorporating an MLP with 1000 units (with dropout) instead of only a softmax layer increases the performance by 1.2 %, and adopting an inverted pyramid architecture (96-192-192-1000-6, $M = 9$, and $R = 3$) with a tuned learning rate (0.02) gives us the best convnet performance of 94.79 %, as seen in Table 1. Comparing the confusion matrix of convnet with SVM's, it is apparent that convnet is better at discriminating moving activities than the latter, especially with very similar ones such as walking downstairs and walking upstairs, which achieved 100 % accuracy. However, stationary activities are a little hard for convnet to recognize, with Laying obtaining the worst classification rate.

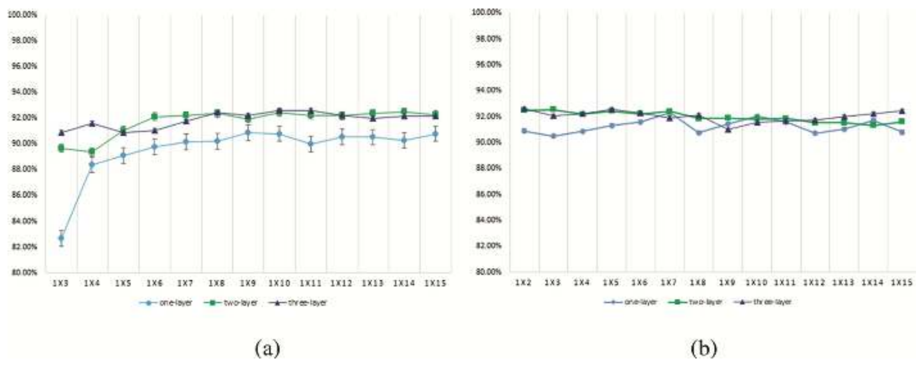

Fig. 4. Performance of convnet with $J(L_1) = 120$, $J(L_2) = 130$, $J(L_3) = 200$, and increasing (a) filter size, and with $M = 9$, $M = 14$, $M = 11$, and increasing (b) pooling size

Table 1. Confusion matrix of best convnet

		Predicted Class						
		W	WU	WD	Si	St	L	Recall
Actual Class	Walking	491	3	2				98.99%
	W. Upstairs		471					100.00%
	W. Downstairs			420				100.00%
	Sitting				436	34	21	88.80%
	Standing		1		24	496	11	93.23%
	Laying				43	23	471	87.71%
	Precision	100.00%	99.16%	99.53%	86.68%	89.69%	93.64%	**94.79%**

We further compare the best convnet with other algorithms, as seen in Table 2. Results show that convnet outperforms other state-of-the-art techniques, which are all using hand-crafted features. It is slightly better than SVM, which previously achieved the best performance in this data set. Using hand-crafted features with convnet also further improves performance, showing that convnet derives another layer of more discriminative features above the hand-crafted ones.

Table 2. Performance of best convnet compared to other algorithms

Algorithm	Accuracy
Naïve Bayes	76.63 %
J48 decision trees	83.63 %
Artificial neural network	91.08 %
Support vector machine	94.61 %
Convnet (L_3)	94.79 %
HCF + Convnet ($J\left(L_1\right) = 200$)	95.75 %

5 Conclusion

We have evaluated convolutional neural networks in recognizing activities using time-series, accelerometer and gyroscope sensor data. It is found that the complexity of derived features indeed increases with increasing convolutional layers, but the difference in complexity between adjacent layers decrease with each additional layer. Preserving the information passed from input to convolutional layers is also important, and a wider correlation between nearby readings should be exploited. Lastly, it was shown that convnet outperformed all other state-of-the-art algorithms in HAR, particularly SVM (which has previously achieved the best performance on the HAR dataset), exhibiting noticeably better performance in classifying moving activities than the latter.

Future studies will include examining and comparing the confusion matrices of other HAR algorithms with convnet, and the inclusion of frequency convolution.

Acknowledgements. This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015-R0992-15-1011) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

1. LeCun, Y., Bengio, Y.: Convolutional Networks for Images, Speech, and Time-Series, The Handbook of Brain Theory and Neural Networks, pp. 255–258 (1998)
2. Dobrucalı, O., Barshan, B.: Sensor-activity relevance in human activity recognition with wearable motion sensors and mutual information criterion. Inf. Sci. Syst. **264**, 285–294 (2013)
3. Bengio, Y., Goodfellow, I.J., Courville, A.: Deep Learning, Book in preparation for MIT Press (2015). http://www.iro.umontreal.ca/~bengioy/dlbook
4. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Ferscha, A., Mattern, F. (eds.) PERVASIVE 2004. LNCS, vol. 3001, pp. 1–17. Springer, Heidelberg (2004)
5. Kwapisz, J., Weiss, G., Moore, S.: Activity recognition using cell phone accelerometers. SIGKDD Explor. **12**(2), 74–82 (2010)
6. Wu, W., Dasgupta, S., Ramirez, E.E., Peterson, C., Norman, G.J.: Classification accuracies of physical activities using smartphone motion sensors. J. Med. Internet Res. **14**(5), 105–130 (2012)
7. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. Eur. Symp. Artif. Neural Netw. (ESANN) **19**, 437–442 (2013)
8. Plotz, T., Hammerla, N.Y., Olivier, P.: Feature learning for activity recognition in ubiquitous computing. Int. Joint Conf. Artif. Intell. (IJCAI) **2**, 1729–1734 (2011)
9. Vollmer, C., Gross, H.-M., Eggert, J.P.: Learning features for activity recognition with shift-invariant sparse coding. In: Mladenov, V., Koprinkova-Hristova, P., Palm, G., Villa, A.E., Appollini, B., Kasabov, N. (eds.) ICANN 2013. LNCS, vol. 8131, pp. 367–374. Springer, Heidelberg (2013)
10. Zeng, M., Nguyen, L.T., Yu, B., Mengshoel, O.J., Zhu, J., Wu, P., Zhang, J.: Convolutional neural networks for human activity recognition using mobile sensors. In: International Conference on Mobile Computing, Applications and Services (MobiCASE) (2014)
11. Bengio, Y.: Practical Recommendations for Gradient-Based Training of Deep Architectures. arXiv:1206.5533v2 (2012)