Evaluation of Deep Convolutional Neural Network Architectures for Human Activity Recognition with Smartphone Sensors

Charissa Ann Ronao\textsuperscript{a}, Sung-Bae Cho
Department of Computer Science, Yonsei University
cvronao@sclab.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

Abstract
Feature extraction is the most vital and critical stage in performing effective activity recognition. On the other hand, deep learning, most especially convolutional neural networks (convnets), have garnered a lot of attention in recent years with its success in the image and speech domains because of its powerful feature extraction mechanism. In this paper, we utilize convnets to classify activities using time-series data collected from smartphone sensors and evaluate its different architectures. Experiments show that increasing the number of convolutional layers increases performance, but the complexity of the derived features decreases with every additional layer. Moreover, as opposed to blindly increasing the number of feature maps to improve performance, preserving the information passed from layer to layer is more important.

1. Introduction
Human activity recognition (HAR) has been a central issue in a wide variety of real-world applications such as pervasive and mobile computing, surveillance-based security, and context-aware computing [1]. Because of the rapid popularization of smartphones with powerful sensors, HAR has attracted a lot of attention in recent years, resulting to a number of research breakthroughs. However, despite HAR being a classification task which comprises of three stages: data collection/preprocessing, feature extraction, and classification—existing works do not give much importance to the most critical stage in the process, feature extraction/design. In fact, it was found that the lack of systematic research on feature design has been considered as one of the major shortcomings of current HAR systems [2].

Recently, deep learning, and convolutional neural networks (convnets) in particular, have revolutionized the state-of-the-art of difficult problems such as image classification and speech recognition. This is because of the inherent characteristics of convnets to extract data-adaptive and relevant features while exploiting the local dependence and translation invariance of various data such as image, speech, and time-series signals—all this without the use of advanced data preprocessing or manual feature design [3]. In this paper, we demonstrate the use of convnets on time-series, raw sensor data from the accelerometer and the gyroscope, and investigate the implications of varying number of layers and feature maps.

This paper is organized as follows: Section 2 surveys the related work, followed by a discussion of convnets with respect to time-series, sensor data in Section 3; Section 4 presents the experimental results, and finally, Section 5 concludes the article.

2. Related Work
As of this writing, only a mere 3% of the total research efforts in deep learning are geared towards activity recognition using time-series signals. Duffner et al. exploited two-layer convnets to recognized 14 gestures using a smartphone [4]. Their work proved that convnets outperform state-of-the-art methods such as DTW, HMM, SVM, BLSTM, and hand-crafted features. In [2], Plotz et al. have applied deep belief networks to human activity recognition. However, such deep architectures does not exploit the inherent local dependency of time-series signals. Convnets were finally applied to human activity recognition using sensor signals in [5] and [6]: however, the former had assessed the problem of time-series classification in general, and the latter had only evaluated a one-layer convnet. In the speech domain, comparison of different architectures have been described in [7].
the same way, this work aims to answer the question of how feature learning and performance are affected by different convnet architectures. We illustrate the implications of increasing convnet layers and feature maps on performance, using a simple softmax classifier.

3. Convolutional Neural Networks for Human Activity Recognition

Convolutional neural networks (convnets) exploit the local dependency characteristics inherent in time-series sensor data and the translation invariant nature of activities. The convolutional layers compute a mixture of nearby sensor readings while the pooling layers make the representation invariant to small translations of the input [8]. A simple convnet architecture is illustrated in Fig. 1. Given accelerometer and gyroscope sensor data input $x^N = [x_1, ..., x_N]$, where $N$ is the number of time steps per window, the output of the convolutional layer is:

$$c^{l,i}_j = \sigma \left( b_j + \sum_{m=1}^{M} w^{l,i}_{m} x_{i+m-1}^j \right), \quad (1)$$

where $\sigma$ is the activation function, $b_j$ is the bias term for the $j$th feature map, $M$ is the kernel/filter size, $w^{l,i}_{m}$ is the weight for the $j$th map and $m$th filter index, and $l$ is the layer index.

The pooling layer follows the convolutional layer to replace the output of the latter into a kind of summary statistic of nearby outputs. The max-pooling operation produces the maximum value among a set of nearby inputs, given by

$$p^{l,i}_j = \max_{r \in R} (c^{l,i}_{j(kt+r)}), \quad (2)$$

where $R$ is the pooling size, $T$ is the pooling stride, and $p^{l,i}_j$ is the pooling layer output.

Several convolutional and pooling layers can be stacked to form a deep neural network architecture. These layers act as a hierarchical feature extractor, which extricates discriminative and informative representations with respect to the data. At the topmost layer, a very simple softmax classifier is utilized to recognize activities. Given flattened activity feature values $p^{l} = [p_1, ..., p_I]$, where $I$ is the number of units in the last pooling layer, the output of the softmax classifier is the activity class $c$:

$$P(c|p) = \arg \max_{c \in C} \frac{\exp(p^{l-1}w^k + b^k)}{\sum_{k=1}^{N_C} \exp(p^{l-1}w^k)} \quad (3)$$

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

The first round of forward propagation is performed using eqns. (1)–(3), giving us the error values of the network. Training is done by stochastic gradient descent (SGD) on minibatches of sensor training examples. Backpropagation to update weights is done by computing the gradient of the convolutional weights:

$$\frac{\delta E}{\delta w^k_{ab}} = \sum_{i=0}^{N-1} \frac{\delta E}{\delta y^l_{i+a}} \sigma^l \left( \sum_{j=1}^{M} w_{m}^{l,i} x_{i+m-1}^j \right), \quad (4)$$

where $E$ is the error/cost function, $y^l_{i+a}$ is the nonlinear mapping function equal to $\sigma^l \left( \sum_{j=1}^{M} w_{m}^{l,i} x_{i+m-1}^j \right)$, and deltas $\frac{\delta E}{\delta y^l_{i+a}} = \sigma^l \left( \sum_{j=1}^{M} w_{m}^{l,i} x_{i+m-1}^j \right)$.

The pooling layer does not really do any learning in itself: they are just mechanisms that introduce sparseness as well as translation invariance. 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.

4. Experimental Results

The publicly-available HAR smartphone dataset from the UCI repository has been utilized for all our experiments. It 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, 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 to 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 sensor data input.

For every run, we set filter size to 1x3 and pooling size to 1x2, with strides all equal to 1x1. Padding is used to perform ‘full’ convolution with the inputs in every layer, and ReLU activation function is used. We set the learning rate to 0.01, gradually increase momentum from 0.5 to 0.99, set 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. We increase the number of feature maps, $J$, from 10–200 (by intervals of 10), and $J$ is the same for all convolutional layers in one architecture [9].

As seen in Fig. 2, the $J$ configurations that achieved the best error rate on test data in the one-layer, two-layer, and three-layer categories are 120, 130, and 200, respectively. We have continued to increase $J$ for the three-layer architecture but performance did not improve further. This shows that increasing the number of feature maps as large as we can does not necessarily translate to better performance. Instead, it is actually more important to preserve the information passed from the 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 $J_1 = 120$ and $J_2 = 130$ very close to the number of points in an activity example (N = 128), and the jump to $J_3 = 200$ covering for the addition of three convolutional layers since the input. It can also be observed in the figure that there is a noticeably bigger jump in performance after adding the 2nd layer than after adding the 3rd layer. This means that indeed, derived 2nd layer features are much more complex than 1st layer basic features, but difference in complexity between 2nd layer and 3rd layer features are not that great. Yet, we cannot deny that adding a third layer still improves performance.

5. Conclusion

We have examined different convnet architectures with 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.

Future studies will include increasing the filter and pooling sizes, experimenting with increasing number of features from bottom to top layer, and the inclusion of frequency convolution.

References