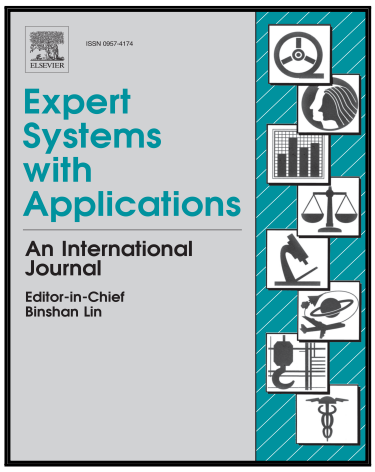


Human activity recognition with smartphone sensors using deep learning neural networks

Charissa Ann Ronao , Sung-Bae Cho

PII: S0957-4174(16)30205-6
DOI: 10.1016/j.eswa.2016.04.032
Reference: ESWA 10652

To appear in: *Expert Systems With Applications*

Received date: 4 August 2015
Revised date: 14 January 2016
Accepted date: 25 April 2016

Please cite this article as: Charissa Ann Ronao , Sung-Bae Cho , Human activity recognition with smartphone sensors using deep learning neural networks, *Expert Systems With Applications* (2016), doi: 10.1016/j.eswa.2016.04.032

Highlights

- This paper proposes a deep convolutional neural network for HAR using smartphone sensors.
- Experiments show that the proposed method derives relevant and more complex features.
- The method achieved an almost perfect classification on moving activities.
- It outperforms other state-of-the-art data mining techniques in HAR.

Human Activity Recognition with Smartphone Sensors using Deep Learning Neural Networks

Charissa Ann Ronao and Sung-Bae Cho
Department of Computer Science, Yonsei University
50 Yonsei-ro, Sudaemoon-gu, Seoul 120-749, Korea
cvronao@sclab.yonsei.ac.kr, sbcho@cs.yonsei.ac.kr

Abstract. Human activities are inherently translation invariant and hierarchical. Human activity recognition (HAR), a field that has garnered a lot of attention in recent years due to its high demand in various application domains, makes use of time-series sensor data to infer activities. In this paper, a deep convolutional neural network (convnet) is proposed to perform efficient and effective HAR using smartphone sensors by exploiting the inherent characteristics of activities and 1D time-series signals, at the same time providing a way to automatically and data-adaptively extract robust features from raw data. Experiments show that convnets indeed derive relevant and more complex features with every additional layer, although difference of feature complexity level decreases with every additional layer. A wider time span of temporal local correlation can be exploited (1x9~1x14) and a low pooling size (1x2~1x3) is shown to be beneficial. Convnets also achieved an almost perfect classification on moving activities, especially very similar ones which were previously perceived to be very difficult to classify. Lastly, convnets outperform other state-of-the-art data mining techniques in HAR for the benchmark dataset collected from 30 volunteer subjects, achieving an overall performance of 94.79% on the test set with raw sensor data, and 95.75% with additional information of temporal fast Fourier transform of the HAR data set.

Keywords. Human activity recognition; deep learning; convolutional neural network; smartphone; sensors.

1. Introduction

With the rapid technological advancement and pervasiveness of smartphones today especially in the area of microelectronics and sensors, *ubiquitous sensing*, which aims to extract knowledge from the data acquired by pervasive sensors, has become a very active area of research (Lara and Labrador, 2012). In particular, human activity recognition (HAR) using powerful sensors embedded in smartphones have been gaining a lot of attention in recent years because of the rapid growth of application demands in domains such as pervasive and mobile computing, surveillance-based security, context-aware computing, and ambient assistive living, and the ability to unobtrusively perform the recognition task (Chen et al., 2012). HAR using smartphone sensors is a classic multi-variate time-series classification problem, which makes use of 1D sensor signals and extracts discriminative features from them to be able to recognize activities by utilizing a classifier (Plotz et al., 2011). Such a tight 1D structure implies the presence of highly-correlated temporally nearby readings (LeCun and Bengio, 1998). Moreover, it is apparent that the keys to performing successful HAR are appropriately designed feature representations of sensor data and suitable classifiers (Plotz et al., 2011).

Human activities have inherent hierarchical structures, and in the context of using sensors for HAR, are very prone to small translations at the input (Duong et al., 2009; Bengio et al., 2015). The former refers to the characteristic of activities that can be broken down to simpler actions, while the latter denotes the different forms and styles people perform the same activities. Such attributes of activities and time-series signals are very useful knowledge if properly utilized and taken advantage of by the feature extractor and classifier.

Recent breakthroughs in image and speech recognition have resulted in a new enthusiastic research field called deep learning. Convolutional neural networks (convnet), in particular, have set the latest state-of-the-art in image and speech domains (Krizhevsky et al., 2012). However, not only image and speech can benefit from such a powerful feature extraction mechanism and classifier such as convnet—HAR is also a good match especially when considering translation invariance and temporally correlated readings of time-series signals, hierarchical structure of activities, and HAR feature extraction problems.

In this paper, we propose a convnet as the automatic feature extractor and classifier for recognizing human activities using smartphone sensors. The convolution operation effectively exploits the temporally-local dependency of time-series signals and the pooling operation cancels the effect of small translations in the input (Bengio et al., 2015). Using a multi-layer convnet with alternating convolution and pooling layers, features are automatically extracted from raw time-series sensor data (illustrated in Fig. 1), with lower layers extracting more basic features and higher layers deriving more complex ones. We show how varying convnet

architectures affects the over-all performance, and how such a system that requires no advanced preprocessing or cumbersome feature hand-crafting can outperform other state-of-the-art algorithms in the field of HAR.

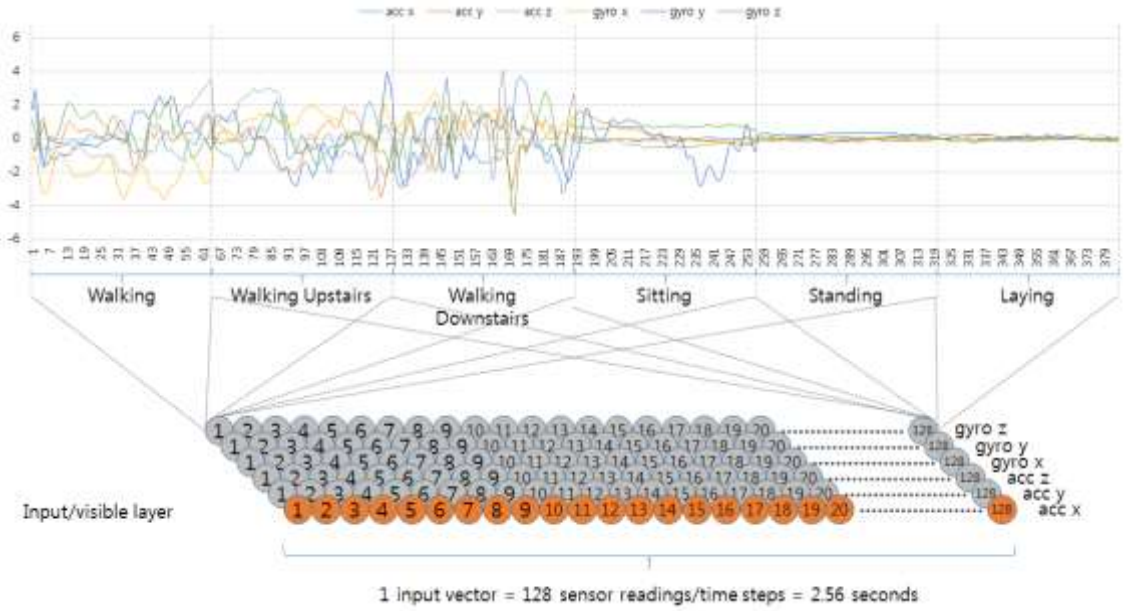


Fig. 1. 1D time-series multi-axes sensor input signal

2. Related Works

Some of the pioneering works in HAR using the accelerometer was published way back in the 90's (Foerster et al., 1999). However, the most cited work that was able to produce satisfactory results with multiple sensors placed on different parts of the body together with different data mining algorithms was by Bao and Intille (Bao and Intille, 2004). This work concluded that the sensor placed on the thigh was the most effective in recognizing different activities, which was a finding that Kwapisz and her colleagues utilized to perform HAR with only one accelerometer embedded in a smartphone (Jwapisz et al., 2010). With their own hand-crafted features from sensor data, their work showed that J48 decision trees and multilayer perceptrons achieve higher performance in terms of accuracy compared to other data mining techniques; however, both classifiers cannot efficiently differentiate between very similar activities such as walking upstairs and walking downstairs.

Sharma et al. used neural networks (ANN) (Sharma et al., 2008), while Khan used decision trees and the Wii Remote to classify basic activities (Khan, 2013). Another work declared k-nearest neighbors (kNN) as the best classifier, but still failed to effectively classify very similar activities (Wu et al., 2012). Nevertheless, the latter and another work by Shaoib et al. both

testified to the usefulness of the gyroscope in conjunction with the accelerometer when classifying activities (Shoaib et al., 2014). Anguita et al. used 561 hand-designed features to classify six different activities using a multiclass support vector machine (SVM) (Anguita et al., 2012). All of these works have derived their own set of hand-designed features, which makes them very hard to compare with each other due to different experimental grounds, and encountered difficulty in discriminating between very similar activities. In this work, using both the accelerometer and gyroscope of a smartphone, we show that convnets are able to overcome these problems of current HAR systems.

Research about HAR using deep learning techniques and their automatic feature extraction mechanism is very few. Among the first works that ventured in it are (Plotz et al., 2011), which made use of restricted Boltzmann machines (RBM), and (Vollmer et al., 2013; Bhattacharya et al., 2014; Li et al., 2014), which both made use of slightly different sparse-coding techniques. The above mentioned deep learning methods indeed automatically extract features from time-series sensor data, but all are fully-connected methods that do not capture the local dependency characteristics of sensor readings (LeCun and Bengio, 1998). Convolutional neural networks (convnets) were finally used together with accelerometer and gyroscope data in the gesture recognition work by Duffner et al. (Duffner et al., 2014), which have concluded that convnet outperforms other state-of-the-art methods in gesture recognition including DTW and HMM.

(Zheng et al., 2014a; Zeng et al., 2014b) both applied convnets to HAR using sensor signals, but the former assessed the problem of time-series in general and the latter only made use of a one-layered convnet, which disregards the possible high advantage of hierarchically extracting features. However, Yang, et al. applied convnets to HAR with hierarchical model to confirm the superiority in several benchmark problems (Yang, et al., 2015). It is obvious for the deep learning to become the dominant technique for the HAR sooner or later, and in this paper, we aim to give the whole picture of utilizing the convnet to work out the problem of HAR from H/W and S/W to hyperparameter tuning, and evaluate the performance with the larger benchmark data collected from 30 volunteer subjects.

3. The Proposed Method

Human activities are also hierarchical in a sense that complex activities are composed of basic actions or movements prerequisite to the activity itself (Duong et al., 2009). Moreover, they are translation-invariant in nature 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 (Bengio et al., 2015). In addition, considering that recognizing activities using sensors involve time-series signals with a strong 1D, highly-temporally correlated structure (LeCun and Bengio,

1998), and extracted sensor features having a very high impact on over-all performance (Plotz et al., 2011), the use of a technique that addresses all these is very vital. Convolutional neural networks (convnets) exploit these signal and activity characteristics through its convolution and pooling operations, together with its hierarchical derivation of features.

We start by describing our hardware and software setup, and move on to the concepts of convnets, its hyperparameters, and regularization techniques used.

3.1. Hardware and Software Setup

Fig. 2 shows the hardware and combinations of software constructed in our research. When it comes to convolutional neural networks, the network's size is limited mainly by the amount of memory available on the GPUs being used and by the amount of training time that the user is willing to tolerate (Krizhevsky et al., 2012). For example, training LeNet5 requires a minimum GPU RAM of 1GB. Logically, a larger GPU RAM is needed to train much bigger convolutional networks. To address this, our hardware is composed of two Intel Xeon E5 CPUs that drive two NVIDIA Quadro K5200 GPUs. The former has six cores and twelve threads each, powerful enough to drive an NVIDIA GPU. The two NVIDIA Quadro K5200 GPUs have 8 gigabytes of RAM, 2,304 CUDA cores, and a bandwidth of 192GB/s. The bandwidth of the Quadro K5200 is considered to be in the higher end of the hierarchy of NVIDIA GPUs; its RAM is greater than that of Tesla K20, and its bandwidth is comparative to the latter as well.

The software is installed with the latest Ubuntu Linux operating system (14.04), in conjunction with Python and the CUDA Toolkit. Python has very efficient libraries for matrix multiplication, which is vital when working with deep neural networks, while the CUDA Toolkit provides a comprehensive development environment for NVIDIA GPU-accelerated computing. Built on top of these is Theano, a C/CUDA compiler that enables floating point operations on the GPU, with a tight integration with the Python Numpy library and allows for speeds rivaling hand-crafted C implementations for large amounts of data (Bergstra et al., 2010).

Next in line is Pylearn2—a rapidly developing machine learning library in Python (Goodfellow et al., 2013). The library focuses on flexibility and extensibility of implementing deep learning algorithms, which makes sure that nearly any research idea is feasible to implement in the library. Implementation on Pylearn2 has three main code blocks: the data provider, the deep learning algorithm, and the YAML configuration file. The data provider accesses the database of sensor data and converts it to a form that the Pylearn2 algorithm implementation can understand (properly-shaped numpy arrays). The algorithm implementation includes all the parts of the particular deep learning model, complete with neural network classes, a cost function, and a training algorithm. The YAML configuration file contains the

training procedure, hyperparameter settings, algorithm function calls, and even data preprocessing directions, all in one place, which enables researchers to easily reproduce their research and save experiment parameters for future reference.

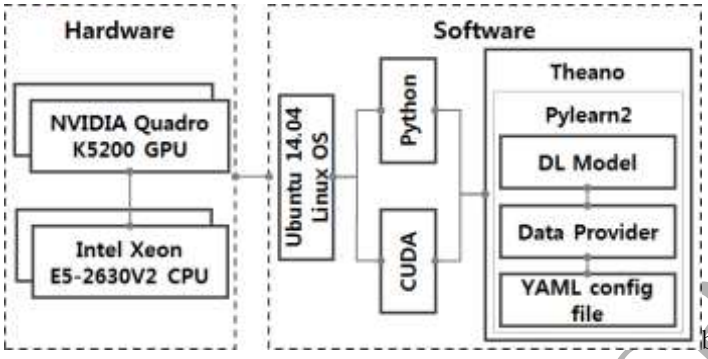


Fig. 2. Hardware and software setup

3.2. Deep Convolutional Neural Networks for HAR

Convolutional neural networks perform convolution operations instead of matrix multiplication. Fig. 3 shows the whole process of the convolutional neural networks for training and classification processes for HAR and the hyperparameters that should be determined. In this figure, if $x_i^0 = [x_1, \dots, x_N]$ is the accelerometer and gyroscope sensor data input vector and N is the number of values per window, the output of the first convolutional layer is:

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

where l is the layer index, σ is the activation function, b_j is the bias term for the jth feature map, M is the kernel/filter size, and w_m^j is the weight for the feature map j and filter index m. Similarly, the output of the lth convolutional layer can be calculated as follows:

$$c_i^{l,j} = \sigma\left(b_j^l + \sum_{m=1}^{M} w_m^{l,j} x_{i+m-1}^{l-1,j}\right). \tag{2}$$

A summary statistic of nearby outputs is 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

$$p_i^{l,j} = \max_{r \in R}(c_{i \times T + r}^{l,j}), \tag{3}$$

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 extract discriminative and informative

representations with respect to the data, with basic to more complex features manifesting from bottom to top.

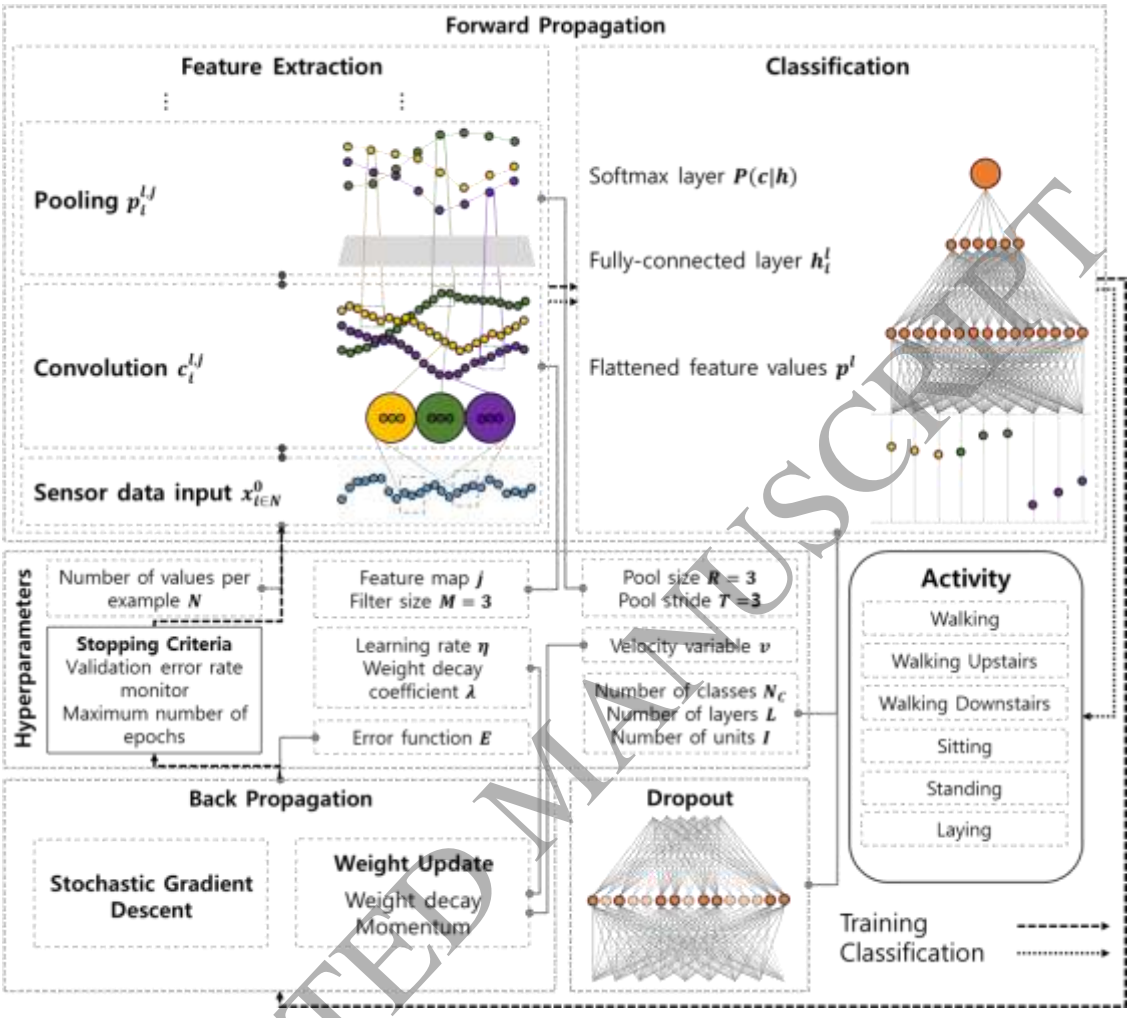


Fig. 3. The overview of the convolutional neural network used for this paper

A combination of fully-connected layer and softmax classifier (or just a simple softmax layer) can be utilized to recognize activities, which acts as the topmost layer. Features from the stacked convolutional and pooling layers are flattened to form feature vectors $p^l = [p_1, \dots, p_I]$, where I is the number of units in the last pooling layer, as input to the fully-connected layer:

$$h_i^l = \sum_j w_{ji}^{l-1}(\sigma(p_i^{l-1}) + b_i^{l-1}), \tag{4}$$

where σ is the same activation function used in the previous layers, w_{ji}^{l-1} is the weight connecting the i^{th} node on layer l-1 and the j^{th} node on layer l, and b_i^{l-1} is the bias term. The output of the last layer, the softmax layer, is the inferred activity class:

$$P(c|p) = \underset{c\in C}{\operatorname{argmax}} \frac{\exp(p^{L-1}w^L + b^L)}{\sum_{k=1}^{N_C}\exp(p^{L-1}w_k)}, \tag{5}$$

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 equations (1)-(4), 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 training data examples. Backpropagation through the fully-connected layer is computed by

$$\frac{\partial E}{\partial w_{ij}^l} = y_i^l \frac{\partial E}{\partial x_j^{l+1}}, \tag{6}$$

where E is the error/cost function, $y_i^l = \sigma(x_i^l) + b_i^l$, w_{ij}^l is a weight from a unit u_i^l in layer l to a unit u_j^{l+1} in layer $l+1$, and x_j^{l+1} is the total input to unit u_j^{l+1}. Backpropagation to adjust weights in the convolutional layers is done by computing the gradient of the weights:

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=0}^{N-M-1} \frac{\partial E}{\partial x_{ij}^l} y_{(i+a)}^{l-1}, \tag{7}$$

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

3.2.1. Regularization

Very large weights can cause the weight vector to get stuck in a local minimum easily since gradient descent only makes small changes to the direction of optimization. This will eventually make it hard to explore the weight space. Weight decay or L2 regularization is a regularization method that adds an extra term into the cost function that penalizes large weights. For each set of weights, the penalizing term $\lambda \sum_w w^2$ is added to the cost function:

$$E = E_0 + \lambda \sum_w w^2, \tag{8}$$

where E_0 is the unregularized cost function, and λ is the weight decay coefficient. With this new cost function, the learning rule becomes:

$$w_i = (1 - \eta\lambda) w_i - \eta \frac{\partial E_0}{\partial w_i}, \tag{9}$$

where $1 - \eta\lambda$ is the weight decay factor.

Momentum-based gradient descent introduces the notion of velocity for the parameters being optimized, in such a way that the gradient changes the velocity rather than the position in weight

space directly. Let $v = [v_1, \ldots, v_K]$ as velocity variables, one for each weight variable. The gradient descent update rule becomes:

$$v \rightarrow v' = \mu v - \eta \nabla E, \tag{10}$$

$$w \rightarrow w' = w + v', \tag{11}$$

where μ is the momentum coefficient.

Dropout modifies the network itself to avoid overfitting instead of modifying the cost function. It works by randomly and temporarily deleting a node in the network, while leaving input and output neurons intact, which makes it equivalent to training a lot of different neural networks. The networks with different architectures will overfit in different ways, but their average results will effectively reduce overfitting (Hinton et al., 2012). It also forces neurons not to rely on the presence of other particular neurons, enabling learning of more robust features (Krizhevsky et al., 2012). Dropout is accompanied by an include probability, and is done independently for each node and for each training example. In our proposed convnet architecture, dropout is applied only to the fully-connected layer.

3.2.2. Hyperparameters

It is clear that there is a large number of possible combinations of setting for the convnet hyperparameters. To assess the effects of varying the values of these hyperparameters on the performance of the network when using HAR sensor data, we incorporated greedy-wise tuning starting from the number of layers L (one-layer, L_1; two-layer, L_2; three-layer, L_3; and four-layer, L_4), 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 4, the number of feature maps from 10 to 200 in intervals of 10 (the same number for all layers (Wu et al., 2012)), the filter size from 1x3 to 1x15, and pooling size from 1x2 to 1x15. Up until the adjustment of pooling size, we use only a simple softmax classifier. On the other hand, we switch to a multilayer perceptron on the succeeding runs to be able to show performance improvements that result in incorporating the final changes to the architecture.

4. Experiments

4.1. Data Set and Experimental Setup

Accelerometer and gyroscope tri-axial sensor data were collected from 30 volunteer subjects who performed six different activities while the smartphone was in their pockets. These sensor data were sampled at a rate of 50Hz, and were separated into windows of 128 values, with 50%

overlap; the 128-real value vector stands for one example for one activity (for each acc and gyro axis). With this raw input, we perform 6-channel (6-axes), 1D convolution (Anguita et al., 2013). (When compared to colored images, 3-channel (RGB), 2D convolution is performed.) Table 1 shows the experimental setup.

There are a total of 7352 examples for the training data (from 21 randomly selected subjects), and 2947 examples for the test data (from the remaining 9 subjects). We standardize these values by subtracting the mean and dividing by the standard deviation:

$$z = \frac{x - \bar{x}}{\sigma}. \tag{12}$$

Table 1. Experimental Setup

Parameter	Value
The size of input vector	128
The number of input channels	6
The number of feature maps	10~200
Filter size	1x3~1x15
Pooling size	1x3
Activation function	ReLU (rectified linear unit)
Learning rate	0.01
Weight decay	0.00005
Momentum	0.5~0.99
The probability of dropout	0.8
The size of minibatches	128
Maximum epochs	5000

4.2. Results and Discussion

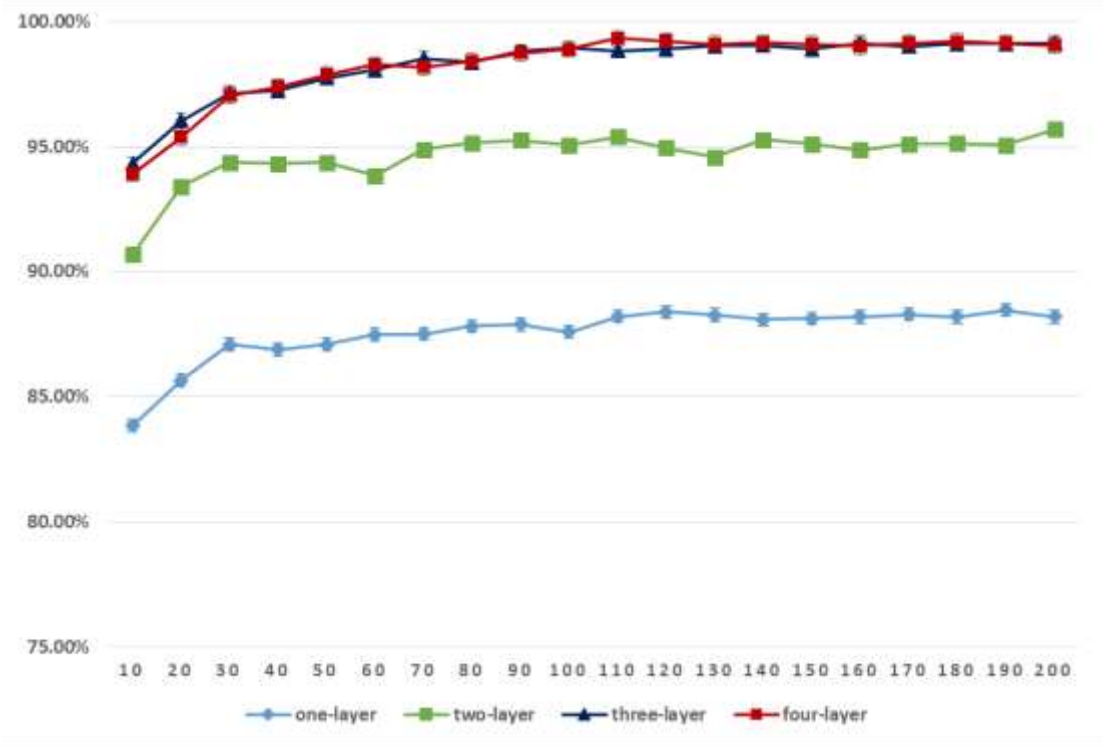
We incorporate a greedy-wise tuning of hyperparameters wherein we adjust the number of layers, number of feature maps, filter size, and pooling size (in that order), and retain the best configuration from the previous step. In this phase, we incorporate max-pooling, a learning rate of 0.01, padding for 'full' convolution, and a Gaussian initialization of U(0.05, 0.05). We used a weight decay value of 0.00005, and increased the momentum from 0.5 to 0.99. Training was done for 5000 epochs, with an early stopping criterion of halting training when there is no decrease in error during the last 100 epochs (Bengio, 2012). The model that achieves the lowest error rate on the validation set is saved.

Fig. 4 shows the effect of increasing number of feature maps on the performance of one-layer (L_1), two-layer (L_2), three-layer (L_3), and four-layer (L_4) architectures. As can be seen in the figure, there is a steady increase in performance on validation data with increasing layers.

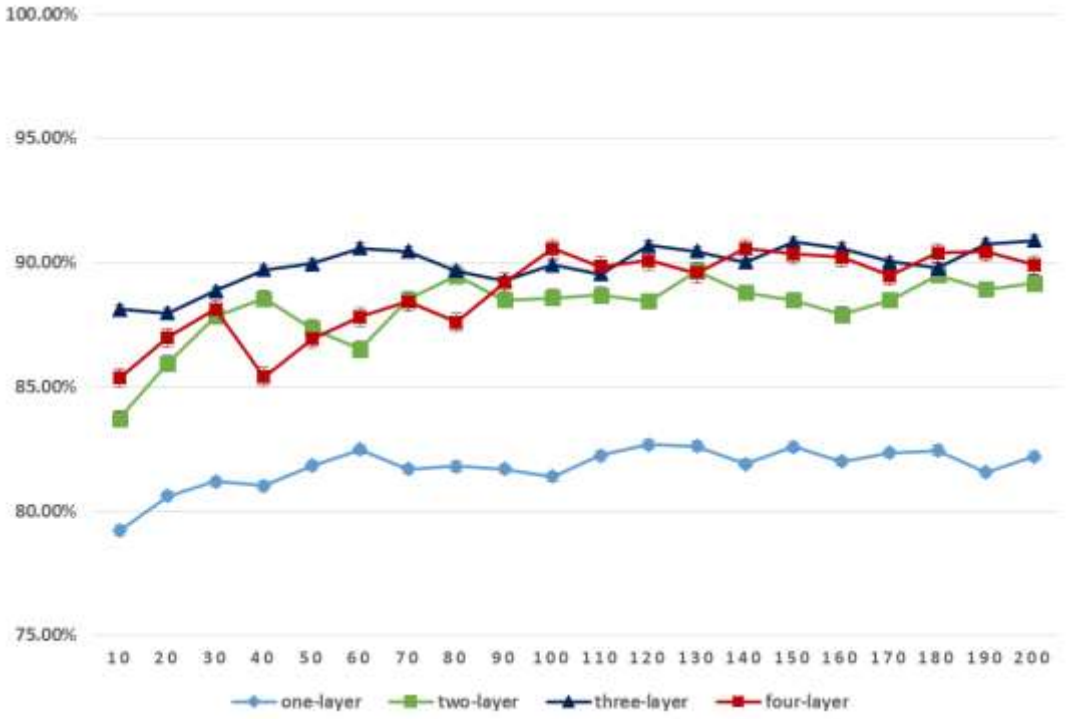
However, on validation data, the increase in performance from L_3 to L_4 is much smaller. On test data, the increase in performance from L_2 to L_3 is much smaller and there are only four small increasing points in performance from L_3 to L_4. Furthermore, we have found that adding a fourth layer results in a decrease in performance from L_3. This shows that as layers are added, more complex features are indeed extracted, but there is a decrease in level of complexity compared to the previous layer.

In addition to that, the graphs show that increasing the number of feature maps does not necessarily translate to increase in performance. For L_1, L_2, and L_3 configurations, the number of feature maps that achieved the best performance on the test set are $J_1 = 120$ (82.68%), $J_2 = 130$ (89.67%), and $J_3 = 200$ (90.90%). The performance does not increase when the number of feature maps is greater than 130, and the product of the number of features in the input should be roughly constant with each additional layer.

From the previous best result configurations, we then increase the filter size. Fig. 5 shows the effect of increasing filter size on performance. Filter sizes that achieved high performance on the test set range from 1x9 to 1x14, an approximate time span of 0.18 to 0.28 seconds. This implies that we can exploit this much larger range of temporal local dependency to achieve better results, as opposed to considering only the immediate neighbors of one time step, 1x3, which is the usual configuration applied.



(a)



(b)

Fig. 4. Accuracies of one-layer (L_1), two-layer (L_2), three-layer (L_3) and four-layer (L_4) convnets on (a) validation data and (b) test data, with increasing number of feature maps

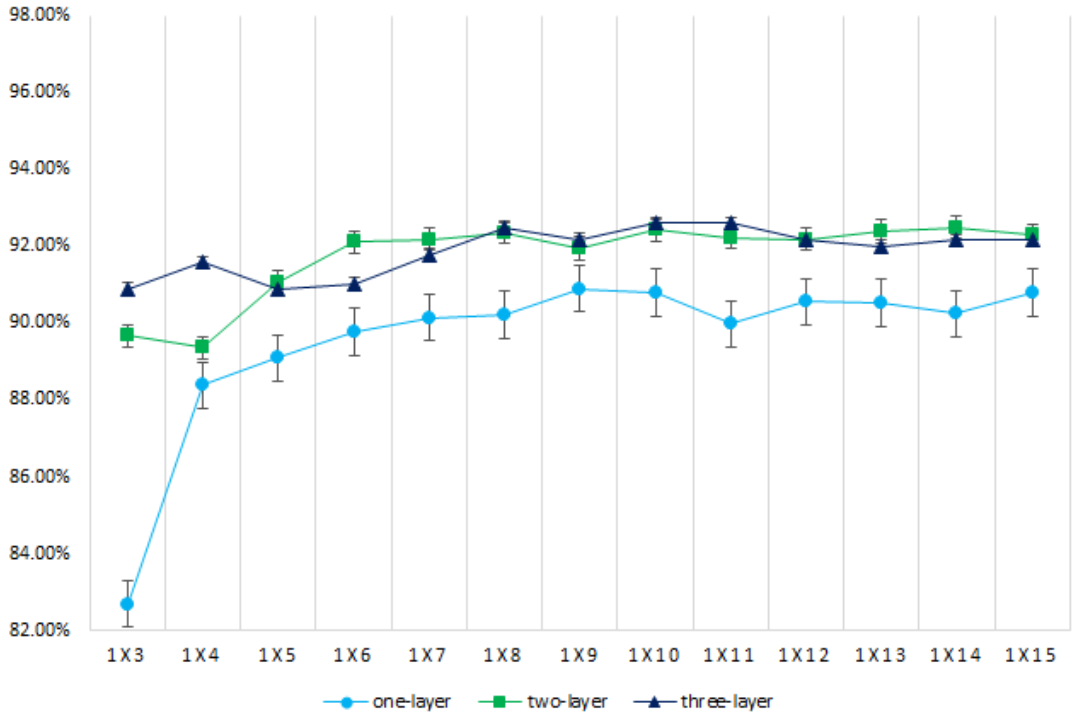


Fig. 5. Performance of convnet with increasing filter size *M*

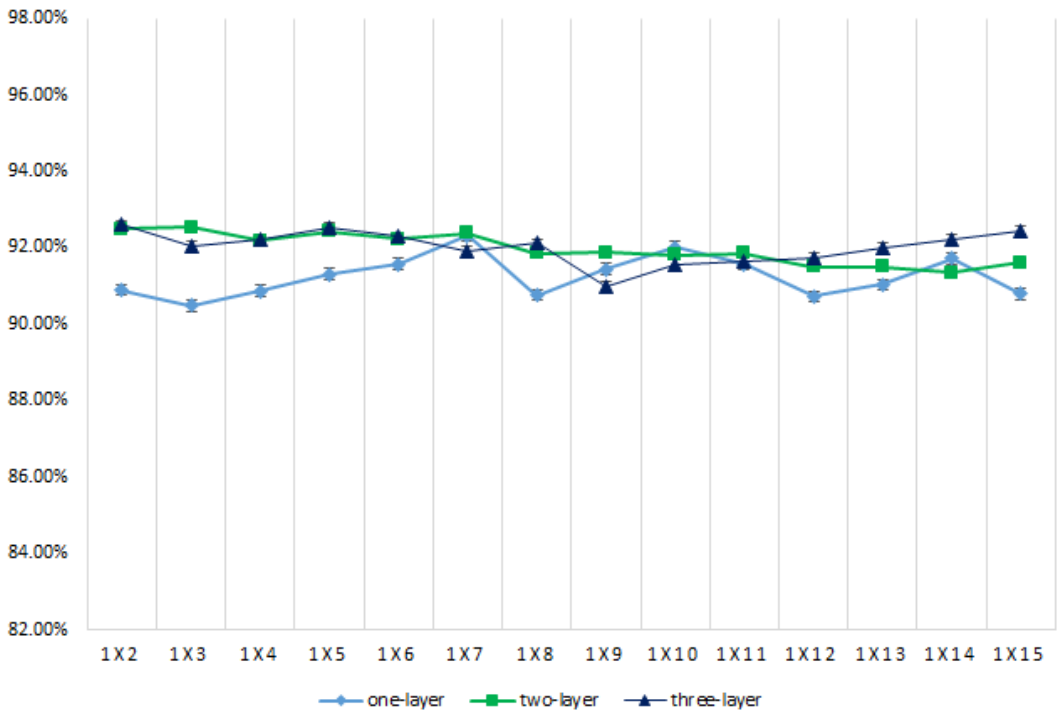


Fig. 6. Performance of convnet with increasing pooling size R

Fig. 6 shows the effect of pooling size on performance. Unlike filter size, pooling size does not have much potential in increasing the performance of the over-all classifier. Based on our multiple runs, a setting of 1x2 or 1x3 is enough. After this run, the current best convnet has hyperparameter settings of $J(L_3) = 200$, $M = 11$, and $R = 2$, with 92.60% accuracy on the test set. The results of tuning the learning rate μ are shown in Table 2. With the current best convnet, now partnered by a multilayer perceptron with a 1000-node fully-connected layer, a μ of 0.006 improves performance on the test set by 1.42%.

Further experimenting with inverted-pyramid architectures yielded a convnet configuration of 96-192-192-1000-6 ($J(L_1) = 96, J(L_2, L_3) = 192, J(L_h) = 1000$, $M = 9$, $R = 3$), wherein the first three values denote the number of feature maps in the convolutional/pooling layers, and the last two values indicate the number of nodes in the fully-connected layer and the softmax layer. With a learning rate of $\mu = 0.02$, we achieve an overall accuracy on the test set of 94.79%.

Table 3 and 4 show the confusion matrices of the best convnet and SVM, respectively. Convnet achieved almost perfect classification for moving activities (99.66%), especially very similar activity classes like walking upstairs and walking downstairs, which were previously perceived to be very difficult in classification (Bao and Intille, 2004; Kwapisz et al., 2010; Wu

et al., 2012). Upon close look, the few confusion cases on moving activities were from subject 13, indicating that this particular subject has a very different style of walking compared to the rest of the 29 subjects in the data set. However, the lowest score achieved was with Laying (87.71%), with the accuracy for stationary activities resulting in only 89.91%. This may be attributed to the lesser waveform frequencies of sensor data from stationary activities than from moving ones, of which convnet is also sensitive to, as was found in speech (Swietojanski et al., 2014). On the other hand, SVM performed better on stationary activities (94.91%). However, like most other classifiers, it failed in differentiating between very similar activities (WD = 88.33%).

Lastly, we compare the best convnet with other state-of-the-art methods in HAR as well as deep learning (in the area of automatic feature extraction), as seen in Table 5. According to the results, convnet outperforms other state-of-the-art data mining techniques in terms of performance on the test set. Also, using additional information of the temporal fast Fourier transform of HAR data set on an L_1 convnet improves performance further by almost 1%, showing that more complex features were indeed derived from additional information of FFT. The features are merged to the first convolutional layer as follows: ([acc_x, acc_x_time-fft], [acc_y, acc_y_time-fft], [acc_z, acc_z_time-fft], [gyr_x, gyr_x_time-fft] , [gyr_y, gyr_y_time-fft] , [gyr_z, gyr_z_time-fft]).

Table 2. Effects of learning rate on convnet performance

Learning rate	Accuracy	Learning rate	Accuracy
0.1	91.882%	0.009	93.716%
0.09	92.323%	0.008	93.648%
0.08	93.173%	0.007	93.716%
0.07	92.799%	0.006	**94.022%**
0.06	92.969%	0.005	93.886%
0.05	92.629%	0.004	93.580%
0.04	93.818%	0.003	93.376%
0.03	93.682%	0.002	93.648%
0.02	93.376%	0.001	93.546%
0.01	93.886%		

Table 3. Confusion matrix of the convnet

		Predicted Class						
		W	**WU**	**WD**	**Si**	**St**	**L**	**Recall**
ual Cla	**Walking**	491	3	2	0	0	0	98.99%

	W. Upstairs	0	471	0	0	0	0	100.00%
	W. Downstairs	0	0	420	0	0	0	100.00%
	Sitting	0	0	0	436	34	21	88.80%
	Standing	0	1	0	24	496	11	93.23%
	Laying	0	0	0	43	23	471	87.71%
	Precision	100.00%	99.16%	99.53%	86.68%	89.69%	93.64%	**94.79%**

Table 4. Confusion matrix of SVM

		Predicted Class						
		W	**WU**	**WD**	**Si**	**St**	**L**	**Recall**
Actual Class	**Walking**	483	7	6	0	0	0	97.38%
	W. Upstairs	12	458	1	0	0	0	97.24%
	W. Downstairs	12	37	371	0	0	0	88.33%
	Sitting	0	1	0	440	48	2	89.61%
	Standing	0	0	0	26	506	0	95.11%
	Laying	0	0	0	0	0	537	100.00%
	Precision	95.27%	91.05%	98.15%	94.42%	91.34%	99.63%	**94.61%**

Table 5. Comparison of convnet to other state-of-the-art methods. HCF is the hand-designed features (Anguita et al., 2012). tFFT is temporal fast Fourier transform from (Sharma et al., 2008).

Method	**Accuracy on test set**
PCA+MLP	57.10%
HCF+NB	74.32%
HCF+J48	83.02%
SDAE+MLP (DBN)	87.77%
HCF+ANN	91.08%
HCF+SVM	94.61%
Convnet (inverted pyramid archi)+MLP	**94.79%**
tFFT+Convnet ($J(L_1) = 200$)	**95.75%**

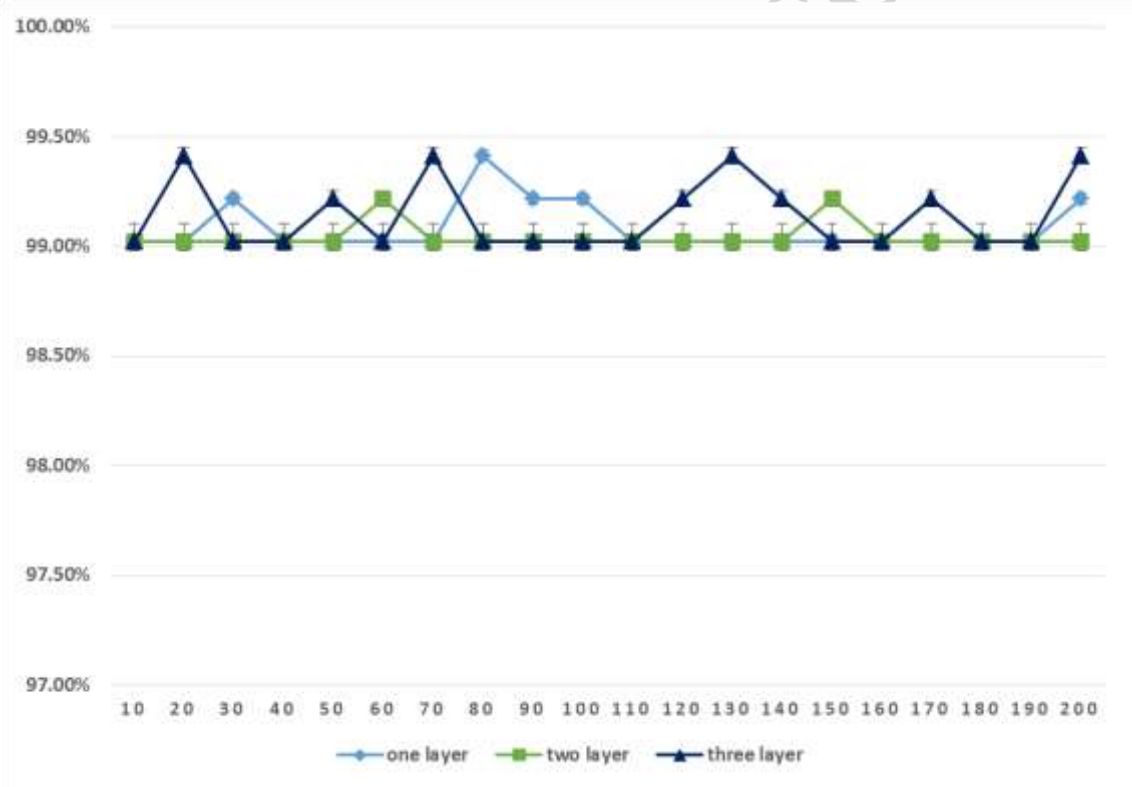
4.2. Additional Experiment

We have experimented another activity dataset that was collected from three graduate students between 20-30 years old (Lee et al., 2011). They grasped the Android smartphone by hand for data collection. The sensor data were separated into windows of 128 values, with 50% overlap; the 128-real value vector stands for one example for one activity. The activities composed with

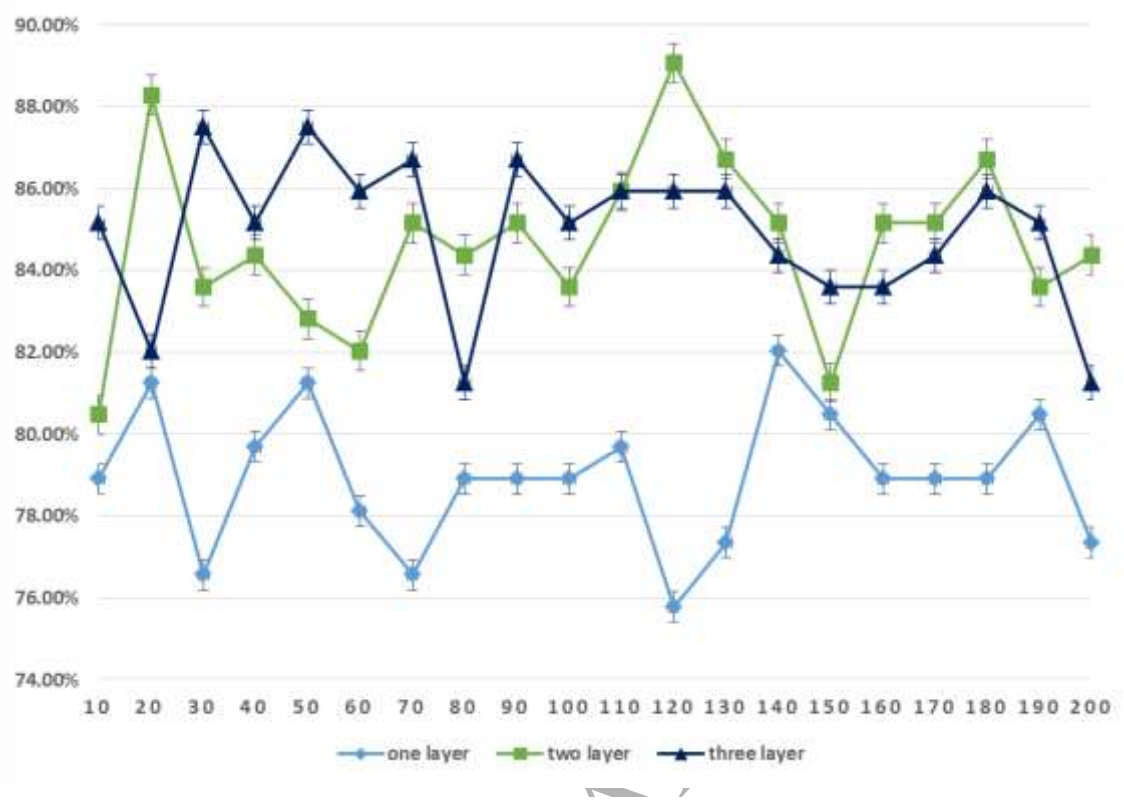
'stand', 'walk', 'stair up', 'stair down' and 'run'. There are a total of 592 examples for training data and 251 examples for the test data. We standardize these values by subtracting the mean and dividing by the standard deviation.

Fig. 7 shows the effect of increasing number of feature maps on the performance of one-layer (L_1), two-layer (L_2) and three-layer (L_3) architectures. Because of small data set, on validation set, there is no difference of performances. On the test data, the performance of one-layer is lower than two- and three-layers, but there is no difference between two- and three-layers. The graphs show that the number of feature maps is not strongly related to the performance.

The results of tuning the learning rate μ are shown in Table 6. With the current best convnet, 0.03 of μ, achieved the accuracy of 93.75%. Table 7 shows the result of comparing the performance with the other competitive methods, which confirms the superiority of the proposed method in accuracy.



(a)



(b)

Fig. 7. Additional experiment: Accuracies of one-layer (L_1), two-layer (L_2) and three-layer (L_3) convnets on (a) validation data and (b) test data, with increasing number of feature maps

Table 6. Additional experiment: Effects of learning rate on convnet performance

Learning rate	Accuracy	Learning rate	Accuracy
0.04	87.50%	0.006	89.84%
0.03	**93.75%**	0.005	89.06%
0.02	90.63%	0.004	89.84%
0.01	92.97%	0.003	89.84%
0.009	90.63%	0.002	88.28%
0.008	91.41%	0.001	87.50%
0.007	88.28%		

Table 7. Additional experiments: Comparison of convnet to other state-of-the-art methods.

Method	Accuracy on test set
HCF+NB	79.43%
HCF+J48	82.62%

SDAE+MLP (DBN)	60.94%
HCF+ANN	82.27%
HCF+SVM	77.66%
Convnet (inverted pyramid archi)+MLP	**93.75%**

5. Conclusions

In this paper, we propose deep convolutional neural networks (convnets) to perform efficient, effective, and data-adaptive human activity recognition (HAR) using the accelerometer and gyroscope on a smartphone. Convnets not only exploit the inherent temporal local dependency of time-series 1D signals, and the translation invariance and hierarchical characteristics of activities, but also provides a way to automatically and data-adaptively extract relevant and robust features without the need for advanced preprocessing or time-consuming feature hand-crafting. Experiments show that more complex features are derived with every additional layer, but the difference in level of complexity between adjacent layers decreases as the information travels up to the top convolutional layers. A wider filter size is also proven to be beneficial, as temporal local correlation between adjacent sensor readings has a wider time duration. In addition to that, the adoption of a low pooling size is better since it is very important to maintain the information passed from input to convolutional/pooling layers.

Comparison of convnet performance with other state-of-the-art data mining techniques in HAR showed that convnet easily outperforms the latter methods, achieving an accuracy of 94.79% with raw sensor data and 95.75% with additional information of FFT from the HAR data set. The achieved high accuracy is mostly due to the almost perfect classification of moving activities, especially very similar activities like walking upstairs and walking downstairs, which were previously perceived to be very hard to discriminate. However, comparing convnet's confusion with SVM's, SVM performed better in classifying stationary activities.

Future works will include experimenting with a combination of convnet and SVM, incorporating frequency convolution together with time convolution, using a different error function, or including cross-channel pooling in place of normal max-pooling. Moreover, we need further study for the analysis of the features extracted automatically by the convent and compare them with the well-known hand-crafted features. Even though the deep convolutional neural networks might be the dominant technique for the HAR, further study on the characteristics of the method and utilizing larger dataset should be conducted.

References

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. Int'l. Conf. on Ambient Assisted Living and Home Care (IWAAL), 216-223.

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. European Symposium on Artificial Neural Networks (ESANN), 437-442.

Bao, L., & Intille, S. (2004). Activity recognition from user-annotated acceleration data. Pervasive Computing, LNCS 3001, 1-17.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. arXiv:1206.5533v2.

Bengio, Y., Goodfellow, I. J., & Courville, A. (2015). Deep Learning, Book in preparation for MIT Press, http://www.iro.umontreal.ca/~bengioy/dlbook.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., & Bengio, Y. (2010). Theano: A CPU and GPU math expression compiler. Proc. of the Python for Scientific Computing Conference (SciPy), 3.

Bhattacharya, S., Nurmi, P., Hammerla, N., & Plotz, T. (2014). Using unlabeled data in a sparse-coding framework for human activity recognition. Pervasive and Mobile Computing 15, 242–262.

Chen, L., Hoey, J., Nugent, C., Cook, D., & Yu, Z. (2012). Sensor-based activity recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 42 (6), 790-808.

Duffner, S., Berlemont, S., Lefebvre, G., & Garcia, C. (2014). 3D gesture classification with convolutional neural networks. Int'l. Conf. on Acoustic, Speech, and Signal Processing (ICASSP), 5432-5436.

Duong, T., Phung, D., Bui, H., & Venkatesh, S. (2009). Efficient duration and hierarchical modeling for human activity recognition. Artificial Intelligence (Elsevier), 173 (7-8), 830-856.

Foerster, F., Smeja, M., & Fahrenberg, J. (1999). Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring. Computers in Human Behavior, 15 (5), 571–583.

Goodfellow, I. J., Warde-Farley, D., Lamblin, P., Dumoulin, V., Mirza, M., Pascanu, R.. Bergstra, J., Bastien, F., & Bengio, Y. (2013). Pylearn2: A machine learning research library. arXiv preprint arXiv:1308.4214.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

Khan, A. M. (2013). Recognizing physical activities using Wii remote. Int'l. Journal of Information and Education Technology, 3 (1), 60-62.

Krizhevsky, A., Sutskever, A. I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Neural Information Processing Systems Conference (NIPS), 1-9.

Kwapisz, J., Weiss, G., & Moore, S. (2010). Activity recognition using cell phone accelerometers, SIGKDD Explorations, 12 (2), 74-82.

Lara, O. D., & Labrador, M. A. (2012). A survey on human activity recognition using wearable sensors. IEEE Communications Surveys & Tutorials, 15 (3), 1192-1209.

LeCun, Y., & Bengio, Y. (1998). Convolutional networks for images, speech, and time-series, The Handbook of Brain Theory and Neural Networks, 255-258.

Lee, Y. S., & Cho, S.-B. (2011). Activity recognition using hierarchical hidden Markov models on a smartphone with 3D accelerometer. Hybrid Artificial Intelligent Systems (HAIS), 6678, 460-467.

Li, Y., Shi, D., Ding, B., & Liu, D. (2014). Unsupervised feature learning for human activity recognition using smartphone sensors. Mining Intelligence and Knowledge Exploration, 8891, 99-107.

Plotz, T., Hammerla, N. Y., & Olivier, P. (2011). Feature learning for activity recognition in ubiquitous computing. Int'l. Joint Conference on Artificial Intelligence (IJCAI), 2, 1729–1734.

Sharma, A., Lee, Y.-D., & Chung, W.-Y. (2008). High accuracy human activity monitoring using neural network. Int'l. Conf. on Convergence and Hybrid Information Technology, 430-435.

Shoaib, M., Bosch, S., Incel, O. D., Scholten, H., & Havinga, P. J. M. (2014). Fusion of smartphone motion sensors for physical activity recognition, Sensors, 14 (6), 10146–10176.

Swietojanski, P., Ghoshal, A., & Renals, S. (2014). Convolutional neural networks for distant speech recognition. IEEE Signal Process. Lett., 21 (9), 1120-1124.

Vollmer, C., Gross, H.-M., & Eggert, J. P. (2013). Learning features for activity recognition with shift-invariant sparse coding. Int'l. Conf. on Artificial Neural Networks and Machine Learning (ICANN), 8131, 367–374.

Wu, W., Dasgupta, S., Ramirez, E. E., Peterson, C., & Norman, G. J. (2012). Classification accuracies of physical activities using smartphone motion sensors. Journal of Medical Internet Research, 14 (5), doi: 10.2196/jmir.2208.

Yang, J. B., Nguyen, M. N. San, P. P., Li, X. L., Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. Int'l. Joint Conf. on Artificial Intelligence (IJCAI), 3995-4001.

Zeng, M., Nguyen, L. T., Yu, B., Mengshoel, O. J., Zhu, J., Wu, P., & Zhang, J. (2014a). Convolutional neural networks for human activity recognition using mobile sensors. Int'l. Conf. on Mobile Computing, Applications and Services (MobiCASE), 197-205.

Zheng, Y., Liu, Q., Chen, E., Ge, Y., & Zhao, J. L. (2014b). Time series classification using multi-channels deep convolutional neural networks. Web-Age Information Management (LNICS), 8485, 298-310.