Applied Soft Computing 62 (2018) 915–922

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Real-time human activity recognition from accelerometer data using Convolutional Neural Networks

Andrey Ignatov

Swiss Federal Institute of Technology in Zurich (ETHZ), Switzerland

ARTICLE INFO

Article history:
Received 21 June 2016
Received in revised form 16 August 2017
Accepted 13 September 2017
Available online 21 September 2017

Keywords:
Activity recognition
Deep learning
Convolutional Neural Networks
Time series classification
Feature extraction

ABSTRACT

With a widespread of various sensors embedded in mobile devices, the analysis of human daily activities becomes more common and straightforward. This task now arises in a range of applications such as healthcare monitoring, fitness tracking or user-adaptive systems, where a general model capable of instantaneous activity recognition of an arbitrary user is needed. In this paper, we present a user-independent deep learning-based approach for online human activity classification. We propose using Convolutional Neural Networks for local feature extraction together with simple statistical features that preserve information about the global form of time series. Furthermore, we investigate the impact of time series length on the recognition accuracy and limit it up to 1 s that makes possible continuous real-time activity classification. The accuracy of the proposed approach is evaluated on two commonly used WISDM and UCI datasets that contain labeled accelerometer data from 36 and 30 users respectively, and in cross-dataset experiment. The results show that the proposed model demonstrates state-of-the-art performance while requiring low computational cost and no manual feature engineering.

1. Introduction

The current generation of portable mobile devices, such as smartphones, music players, smart watches or fitness trackers incorporates a wide variety of sensors that can be used for human activity and behavior analysis. This opens up new areas of intelligent applications that use this data for making inferences about different aspects of human life. Among the traditional examples here are healthcare monitoring, life logging, fitness tracking and security applications. Another emerged and rapidly evolving field is an unobtrusive user activity recognition in adaptive mobile applications that adjust their behavior and setup to the current mode of use. One common property of these applications is that they usually need to work out of box for an arbitrary user in an arbitrary environment, since in most cases there is no way of asking the user for training data. Another common challenge is a real-time activity recognition that is especially crucial for security and adaptive apps.

The task of human activity recognition can be generally divided into two main steps. The first step is time series segmentation, and the basic approach to this problem is to use a sliding window of a fixed length and split each time series into equal segments. The question that can arise here is how the recognition accuracy depends on the window length, however it was not covered in previous works. Particularly, for WISDM dataset a sliding window of size 10 s was used in all studies [1–5] except for [6], where an adaptive time series segmentation technique was proposed.

The second step is to extract effective features from the obtained raw segments and then perform their classification. This task is extremely crucial in HAR problem since the quality of the features primarily determines the overall system accuracy. One approach widely used in existing works is to rely on various hand-crafted measures such spectral entropy, energy of different frequency bands, auto-regressive and FFT coefficients, etc. Though in practice this approach often shows good performance, it relies on domain-specific knowledge and its generalization to new data sources and experimental setups is usually mediocre. A different approach is based on deep learning, and the main idea behind it is to automatically learn the required feature representation directly from the data. Besides high accuracy and good generalization, one main advantage of this approach is that after a deep learning model is created, it is trained in an end-to-end fashion, thus completely removing the need of manual feature engineering.

In our problem we are dealing with quasi-periodic accelerometer time series, where the form and size of the periods is determined by the activity type. These periods contain essential information about the corresponding activity and thus the structure of the considered data is primarily local. Among different deep learning

E-mail address: ihnatova@vision.ee.ethz.ch

https://doi.org/10.1016/j.asoc.2017.09.027
1568-4946/

models especially attractive in this context are Convolutional Neural Networks due to their specific architecture. CNNs learn filters that are applied to small sub-regions of the data, and therefore they are able to capture local data patterns and their variations. This unsupervised feature learning is performed inherently in the convolutional layers, and the produced features are then passed to the fully-connected layers where the classification takes place. There is no need to train convolutional layers separately – since they contribute to the CNN's output, they can be optimized with a standard backpropagation algorithm, thus the CNN is trained as a whole to minimize the overall prediction error. Additionally, due to a small number of connections and high parallelism the amount of computations and running time of CNNs is significantly lower compared to other deep learning algorithms. The only weakness of these networks is that they fall behind in capturing global properties of the signal, and in this work we eliminate this problem by augmenting CNNs with some basic statistical features that comprise these aspects of the data.

The main contributions of this paper are as follows:

- We propose an architecture that combines a shallow CNN for unsupervised local feature extraction together with statistical features that encode global characteristics of the time series.
- We study how time series length affects the recognition accuracy and limit it up to 1 s in order to enable real-time activity classification.
- We show that the proposed model outperforms the existing solutions, establishing state-of-the-art results on both WISDM and UCI HAR datasets.
- To ensure user- and platform-independency of the model, we perform a cross-dataset evaluation and compare its performance to the alternative approaches. We test the solution on both desktop and mobile devices to guarantee acceptable running time.
- Finally, we make the source code of the model and the whole pipeline publicly available.[1]

The rest of the paper is arranged as follows. Section 2 introduces related works on human activity recognition and deep learning methods. Section 3 gives an overview of Convolutional Neural Networks and presents an architecture of the proposed system. Section 4 provides the detailed experimental results obtained on two HAR datasets and in cross-dataset experiment, and compares them to the existing solutions. Section 5 gives an information about system computational performance and Section 6 summarizes our conclusions.

2. Related work

The task of human activity recognition using smartphone's built-in accelerometer has been well addressed in literature. When it comes to practical applications, one challenge that arises here is real-time classification of user activity. Though a number of papers proposed online HAR systems, they used recognition intervals that are generally quite long for online classification. In particular, the existing works considered time segments of size 128 [7], 200 [2], 250 [8], 300 [9] and 512 [10], which corresponds to interval duration of 2.56–10 s. Smaller time intervals were used in [11], and while this work shows quite good performance, a very small private dataset obtained from 4 users and a limited range of activities makes its results incomparable to any existing solution. Furthermore, all mentioned systems were based on hand-designed features.

A different approach to feature extraction task is based on deep learning/CNNs, and several works have been conducted to adapt it to HAR problem. The first difference between the proposed solutions is how the input signals are treated. In [12–14] the authors were focused on using multiple sensors and proposed to stack signals from them row-by-row into one "sensor image" that is further passed to a Convolutional Neural Network. In [15], instead of dealing with a raw sensor image, a Discrete Fourier Transform was applied to this image and the obtained features were used for the classification. In [4,16,17], where human activity recognition was performed using accelerometer data from one device, the authors learned feature maps for x-, y- and z-accelerometer channels separately that is similar to how an RGB image is typically processed by CNN.

The architecture of CNNs also varied among the studies. In [4] one convolutional and two fully-connected layers were considered, in [12,15,14] – two and one respectively. In [16,17,13] the authors have proposed even deeper architectures that consisted of three and four convolutional layers. Though deeper architectures are theoretically able to learn more abstract features, they often lead to data overfitting and therefore an appropriate balance should be maintained here. In this work we will show that appropriately tuned shallow CNN can yield an accurate classification while requiring less computational resources.

Nowadays smartphones became an integral part of our daily lives and go with us everywhere, becoming a perfect tool for the analysis of human daily activities. For this reason we have chosen open WISDM [25] and UCI [26] datasets for training and performance evaluation of our model. These datasets contain accelerometer data from Android cell phones that was collected while users were performing a set of different activities, such as walking, jogging, stair climbing, sitting, lying and standing.

Another advantage of these datasets is that they were already used in several research works. For WISDM dataset all previous works developed user-specific solutions, and only [5] considered a user-independent model and proposed using a combination of hand-crafted features and Random Forest or Dropout classifiers on top of them. UCI dataset has a version that is already split into training and test sets that contain data from different participants, therefore user-independent solution was dominant in this case. For UCI dataset, manual feature engineering was the prevailing approach [5,19–23,7], though several deep learning methods were also proposed [15–17,23,24]. In [23] Deep Boltzmann Machines were adapted for unsupervised feature extraction, and though they are not targeted on capturing local data structure their performance was superior to the other hand-crafted solutions. In [16] the authors used deep CNNs with three convolutional layers, but according to the experiments this caused significant data overfitting. A better performance was obtained with two-layered CNNs at the expense of using FFT features instead [15] or in addition [17] to the raw time series data. Another promising solution with low computational cost [24] is based on Recurrent Neural Networks, though it is difficult to compare its accuracy to the previous solutions since a custom split of the dataset into training and test parts was used. The best results for UCI dataset were achieved using 561 hand-designed features proposed in [7] and various classifiers on top of them. Further experimental results obtained on WISDM and UCI datasets are presented in Table 1.

3. Algorithms

In this section, we describe the structure of Convolutional Neural Networks and present the system architecture proposed in this paper.

[1] https://github.com/aiff22/HAR.

Table 1
Classification results of HAR methods proposed for WISDM and UCI datasets.

Paper	Dataset	Method	Testing technique	Accuracy
[5]	WISDM	Handcrafted features + Random Forest	Leave-one-out	83.46
[5]	WISDM	Handcrafted features + Dropout	Leave-one-out	85.36
[5]	UCI	Handcrafted features + Dropout	Leave-one-out	76.26
[5]	UCI	Handcrafted features + Random Forest	Leave-one-out	77.81
[19]	UCI	Hidden Markov Models	21 training/9 testing	83.51
[20]	UCI	Dynamic Time Warping	21 training/9 testing	89.00
[21]	UCI	Handcrafted features + SVM	21 training/9 testing	89.00
[16]	UCI	Convolutional Neural Network	21 training/9 testing	90.89
[22]	UCI	Hidden Markov Models	21 training/9 testing	91.76
[23]	UCI	PCA + SVM	21 training/9 testing	91.82
[23]	UCI	Stacked Autoencoders + SVM	21 training/9 testing	92.16
[18]	UCI	Hierarchical Continuous HMM	21 training/9 testing	93.18
[17]	UCI	Convolutional Neural Network	21 training/9 testing	94.79
[24]	UCI	Recurrent Neural Network	$\frac{3}{4}$ training/$\frac{1}{4}$ testing	95.03
[15]	UCI	Convolutional Neural Network	21 training/9 testing	95.18
[17]	UCI	FFT + CNN features	21 training/9 testing	95.75
[7]	UCI	Handcrafted features + SVM	21 training/9 testing	96.37

3.1. Convolutional Neural Networks

CNN is a hierarchical feed-forward neural network which structure is inspired by the biological visual system. Its principal difference from standard neural networks is that apart from fully-connected layers it has a number of convolutional layers, where it learns filters that are sliding along the input data and applied to its sub-regions. The overall structure of CNNs is described below:

- *Convolutional layer*. In one-dimensional case, a convolution between two vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{f} \in \mathbb{R}^m$ is a vector $\mathbf{c} \in \mathbb{R}^{n-m+1}$, where each element $c_i = \mathbf{f}^T \mathbf{x}_{[i:i+m-1]}$ is computed as a scalar product between the vector $\mathbf{f}$ and the corresponding subsegment of $\mathbf{x}$. In other words, a vector $\mathbf{f}$, which is also called a convolutional filter, is sliding along vector $\mathbf{x}$, a dot product is computed at each step and the obtained values form the outputs of the convolutional layer.
- *Nonlinearity*. To learn non-linear decision boundaries, convolutional layer is typically followed by non-linear activation function that is applied point-wise to its outputs. Three commonly used activation functions are *sigmoidal*, *hyperbolic tangent* and *ReLU*. The third one is defined as $ReLU(x) = max(0, x)$, which is a simple thresholding operation.
- *Pooling layer*. This layer usually follows a convolutional layer and its goal is to reduce and summarize the obtained representation. Two conventional choices to do this is to take an average or maximum of small rectangular blocks of the data.
- *Fully-connected layer*. After several convolutional and max-pooling layers, the output of these layers is flattened into a one-dimensional vector and used for the classification. At this stage additional features can be stacked together with this vector. To learn non-linear dependencies, CNN has one or more fully-connected layers on top of it that perform the classification.
- *Soft-max layer*. Finally, the output of the last layer is passed to a soft-max layer that computes probability distribution over the predicted classes.

All mentioned layers are stacked together and form one Convolutional Neural Network, that can be trained as a whole. One common way do this is to use a back propagation algorithm and optimize training parameters with stochastic gradient descent.

3.2. System architecture

In this work, we propose a CNN architecture that is presented in Fig. 1. The processing of the centered accelerometer data begins in the convolutional layer with 196 convolutional filters that are learned in parallel to create a rich feature representation of the data. The size of each filter is 1×16, and the step of the convolution is 1. Then, after *ReLU* function is applied to the resulting 196 feature maps, max-pooling of size 1×4 is used to reduce feature representation by 4 times. The output of the max-pooling layer is then flattened and stacked together with additional statistical features: mean, variance, sum of the absolute values and the histogram of each input data channel. The joint vector is subsequently passed to a fully-connected layer that consists of 1024 neurons. We use a dropout technique in this layer with dropout rate 0.05 to avoid overfitting. Finally, the outputs of the fully-connected layer are passed to a soft-max layer that computes probability distribution over six activity classes. The model is trained to minimize cross-entropy loss function which is augmented with l_2-norm regularization of CNN weights. The parameters of the network are optimized with Adam [27] modification of stochastic gradient descent using backpropagation algorithm to compute the gradients.

4. Experiments and evaluation

To evaluate the performance of the proposed model, we carried out a set of experiments described in this section. Experiments were performed on WISDM [25] and UCI [26] datasets that contain accelerometer time series data obtained from Android smartphones. These data were collected from 36 and 30 different users respectively while they were performing a specific set of six activities: walking, jogging, stair climbing, sitting, lying and standing. The results obtained on these datasets are summarized in the following sections.

4.1. WISDM dataset

The only work that developed a user-independent solution for this dataset is [5], where leave-one-out validation technique was applied to test the model. Using this technique for performance evaluation of CNN is rather computationally expensive, and therefore we decided to split this dataset into training part with data from users 1–26 and test part with data from the rest 10 users. Our preliminary experiments have shown that the number and the users used in the training dataset greatly affect the recognition accuracy, and after trying several different splits we have chosen one with the highest test error to see to what extent CNN will improve the results in this case.

To establish some baseline results we have implemented three different methods for HAR problem and applied them to this dataset. The first method was based on 40 statistical features

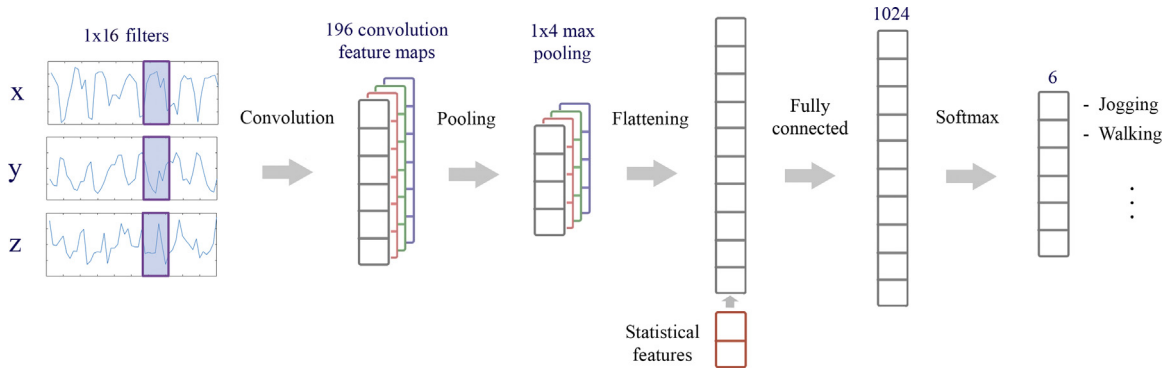

Fig. 1. The architecture of the proposed system.

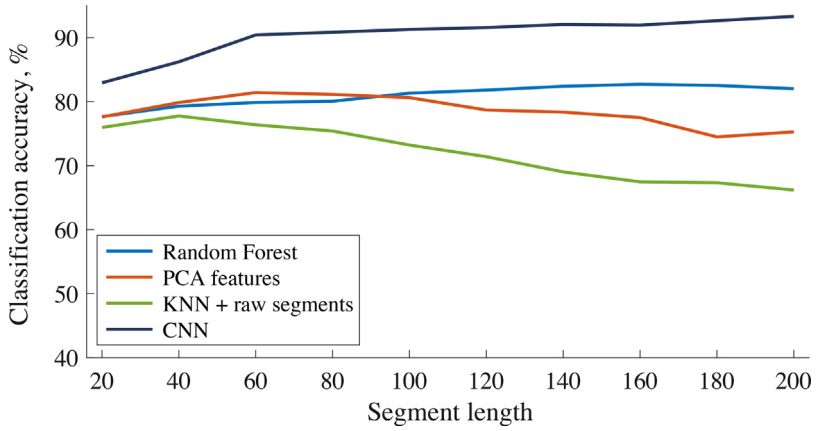

Fig. 2. Dependency between the classification accuracy and the size of the recognition interval.

described in [2] and Random Forest classifier (RF) on top of them. The second method used 26 features that were generated using PCA analysis, and the third one was based on the plain accelerometer time series classification using k-nearest neighbor method.

The first question that we were interested in was how the size of the recognition interval affects the performance of the classifiers. For this purpose we varied this size between 20 and 200 (1 and 10 s respectively) with a step size of 20, and for each value the accuracy of all methods was estimated. The results of this experiment are presented in Fig. 2, and they reveal two interesting findings. The first one is that larger segments do not necessarily lead to better recognition results. While the increase of the recognition interval from 20 to 40–60 gains significant performance boost for all methods, its further growth introduces only moderate improvements for CNN and Random Forest, whereas the accuracy of PCA- and KNN-based methods degrades. Since the conventional segment length for this dataset is 200, it may be reasonable considering smaller recognition intervals along with the standard one. Secondly, CNN demonstrates a strong advantage over the baseline methods for all segment lengths.

The detailed classification results for recognition intervals of size 50 and 200 are presented in Tables 2 and 3 respectively. As one can see, CNN achieves an accuracy of 90.42% and 93.32%, outperforming the baselines (p-value < 0.0001) by more than 10% in both cases. These results are also 1.5% higher (p-value = 0.0009) compared to the original paper [2] that develops a user-specific model for this dataset. Per-class analysis shows that CNN introduces considerable improvements for the first four dynamic activities, while the results for static activities are comparable to the baselines. The reason for this is that the local shape of accelerometer time series corresponding to sitting and standing activities is very similar, therefore learned convolutional filters cannot add much information to statistical features.

4.2. UCI dataset

The next set of experiments was conducted on UCI dataset using the same setup as in the previous section. A conventional partition of the dataset into training and test sets was used unless stated otherwise. The summary of the obtained results for recognition intervals of size 50 and 128 (1 and 2.56 s respectively) is presented in Table 4. For 2.56 s segments – the standard for this dataset – the proposed CNN demonstrates an accuracy of 97.63%, thus outperforming (p-value = 0.0156) by 1.2% all previously proposed solutions including [7], where 561 complex hand-designed features were proposed. With a decrease of interval size to 1 s, which allows nearly instantaneous activity classification, the accuracy lowers to 94.35% that is still in the line with other CNN-based methods.

We have additionally performed a 10-fold user-based cross validation to explore the variability of results on this dataset. The accuracy was ranging between 95.54% and 99.54% with an average value of 97.47% and a standard deviation of 1.17%, which is close to what was observed on the conventional test subset. It should be noted that this subset was used to measure the results reported in Table 1, except for work [5] that also applied a cross-validation technique.

4.3. CNN analysis

To analyze how the proposed combination of CNN with statistical features and how various data preprocessing mechanisms

Table 2
Classification results on WISDM dataset for recognition interval of size 50.

Activity type	Basic features + RF	PCA + RF	Segments + KNN	CNN + stat. features
Jogging	94.03	93.64	86.49	**97.58**
Walking	80.64	88.08	89.23	**97.05**
Upstairs	62.38	57.57	51.45	**62.89**
Downstairs	42.87	25.28	30.94	**76.68**
Sitting	**84.97**	81.96	73.95	82.32
Standing	94.28	91.42	93.45	**95.71**
Overall	79.85	79.86	77.76	**90.42**

Bold values are the best results achieved for each activity type (among the four considered methods).

Table 3
Classification results on WISDM dataset for recognition interval of size 200.

Activity type	Basic features + RF	PCA + RF	Segments + KNN	CNN + stat. features
Jogging	94.72	95.99	72.38	**97.87**
Walking	83.56	87.71	85.88	**98.50**
Upstairs	66.67	29.19	12.42	**72.22**
Downstairs	49.82	14.97	2.72	**87.00**
Sitting	82.47	82.47	74.23	**82.63**
Standing	**95.76**	61.18	96.47	93.33
Overall	82.66	75.28	66.19	**93.32**

Bold values are the best results achieved for each activity type (among the four considered methods).

Table 4
Classification results on UCI dataset for recognition intervals of size 50 and 128.

Activity type	CNN + stat. features, intervals of size 50		CNN + stat. features, intervals of size 128	
	Accuracy	F1-score	Accuracy	F1-score
Walking	92.71	94.90	99.40	99.60
Upstairs	97.31	95.95	100.00	99.47
Downstairs	96.63	95.41	98.81	99.04
Sitting	86.24	89.23	90.04	93.61
Standing	93.49	90.77	98.20	94.71
Lying	99.66	99.83	100.00	99.82
Overall	94.35	94.29	97.63	97.62

influence the classification results, we applied standard and augmented CNNs to plain, centered and normalized accelerometer data. Table 5 shows the results of this experiment. The conventional CNN without any data preprocessing demonstrates an accuracy of 95.31%, and performing time series centering or normalization leads in this case to a significant performance drop. While the structure of accelerometer time series is mainly local, the global form of these signals also contains an important information about the activities, which is irretrievably lost during data preprocessing, causing worse results. In particular, this explains an unusually low CNN's accuracy in [16], where the authors performed time series normalization. Augmenting CNN with statistical features enhances its performance by 0.7%, and performing data centering leads to a further dramatic increase by 1.5%. The explanation of this effect is that time series centering standardize the input data, making the task for CNN easier, while statistical features preserve all the lost information. Data normalization does not help in this situation since it significantly distorts time series shape, removing magnitude information which is critical for activities differentiation.

In addition, we have analyzed how CNN structural parameters affect the classification results. For this purpose we varied the number and size of the convolutional filters, the number of neurons in the hidden layer and the dropout rate to estimate CNN performance in each case. The results are presented in Figs. 3–6. It can be observed that the accuracy is notably improved when the number of neurons and filters is increased to 32 and 64 respectively, while further growth causes only marginal improvements. Particularly, a tiny CNN that consists of 64 convolutional filters and 32 neurons has demonstrated an accuracy of 96.62% on this dataset, which is only 1% lower compared to the larger best-performing network. But using considerably smaller amount of filters will not give sufficient results, that is also supported by [15] – despite having two convolutional layers, the proposed CNN consisted of only 5 and 10 convolutional filters, which resulted in 95.18% accuracy. As for the size of the convolutional filters, the network is not very sensitive to this parameter: while the best accuracy was obtained for filters of size 16, the accuracy does not drop significantly till this size becomes smaller than 4 or greater than 30. Another crucial parameter of CNN is the dropout rate. According to Fig. 6, its extreme values between 0.04 and 0.1 turned to be the most efficient in this task, yielding a performance improvement of 1.5%. We should also note that neither increasing the number of convolutional layers (97.35%) nor the number of fully-connected layers (97.11%) gave better results – while theoretically this can give higher performance, in this task the network just starts to overfit the data, and additional regularization only leads to a drop in

Table 5
UCI classification results for various data preprocessing approaches.

Method	Accuracy, %
CNN + Stat. features + data centering	97.63
CNN + Stat. features	96.06
CNN + Stat. features + data normalization	95.48
CNN	95.31
CNN + Data centering	92.35
CNN + Data normalization	90.77

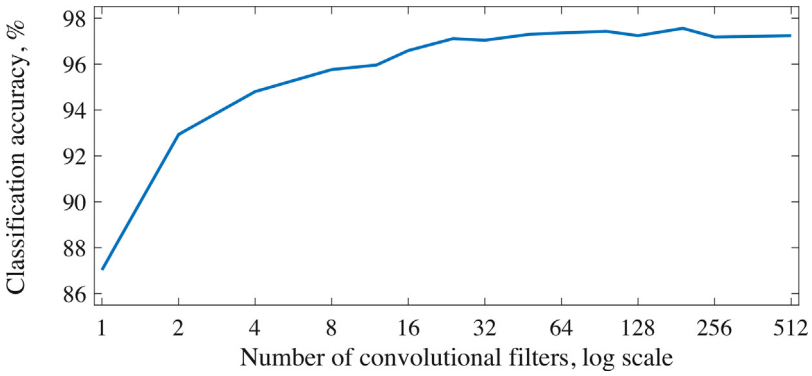

Fig. 3. Dependency between the accuracy and the number of convolutional filters.

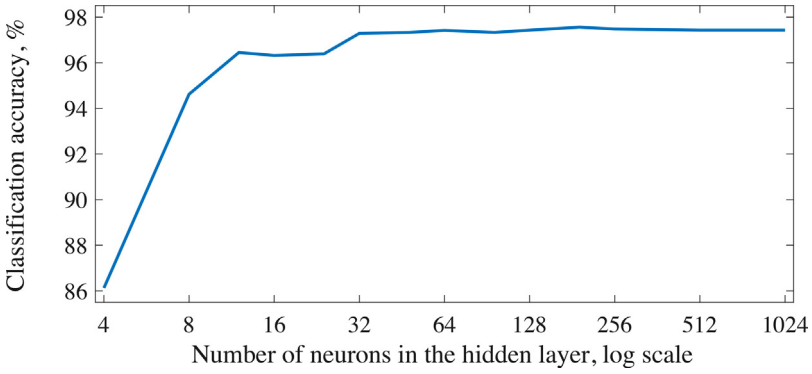

Fig. 4. Dependency between the number of neurons in the hidden layer and CNN accuracy.

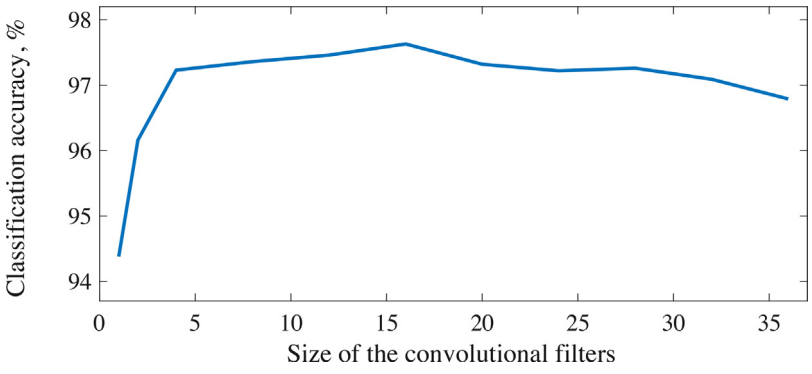

Fig. 5. Dependency between the size of the convolutional filters and CNN accuracy.

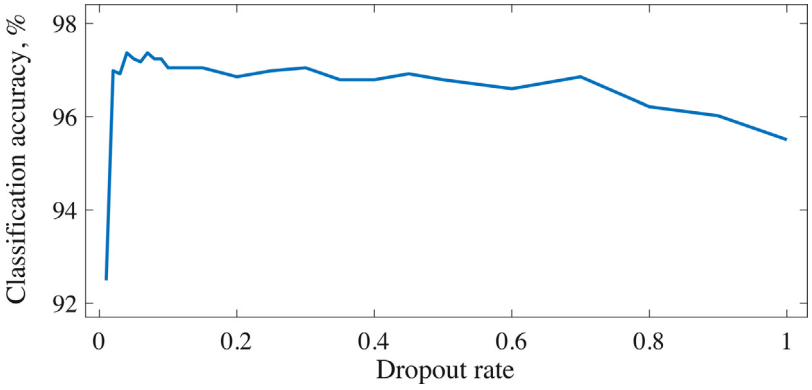

Fig. 6. Dependency between the dropout rate and CNN accuracy on UCI dataset.

Table 6
Classification results for cross-dataset experiment.

Activity type	Basic features + RF	PCA + RF	Segments + KNN	CNN + stat. features
Walking	50.86	31.08	29.22	**84.17**
Upstairs	35.55	61.27	44.51	**62.45**
Downstairs	17.26	0.00	16.61	**77.89**
Sitting	25.83	**96.82**	92.62	91.25
Standing	**91.84**	0.00	7.93	88.45
Overall	46.56	38.26	38.47	**82.76**
$\frac{segments}{s}$ Throughput,	6700	8900	223	149,600

Bold values are the best results achieved for each activity type (among the four considered methods).

the accuracy. Changing *ReLU* activation function to hyperbolic tangent or sigmoid did not lead to improvements too – 97.26% and 97.33%, respectively. Though the proposed network does not suffer from the gradient vanishing problem, *ReLU* gives two main benefits in our case. First, its constant non-vanishing gradient leads to significantly faster learning – the accuracy of 96.9% on UCI dataset is achieved after 3K iterations, while for sigmoid function this takes almost 26K iterations. Additionally, *ReLU* function is known to be less prone to overfitting since it induces the sparsity in the hidden units, which in our case results in slightly better accuracy compared to other activations.

4.4. Cross-dataset evaluation

To test the generalization ability of the proposed solution, a cross-dataset evaluation was performed: WISDM dataset was used for training the model and UCI dataset for testing. Apart from data centering, accelerometer signals from UCI dataset were additionally divided by 10 to ensure the same range of variability. The classification results for this experiment are presented in Table 6.

As can be expected, baseline methods have shown a mediocre performance since they rely on features that are quite sensitive to variations of the signal form. They were able to determine whether the activity is passive (sitting, standing) or active (walking, stair climbing), but inside each class the results were almost random. Whereas CNN learns features that are in general invariant to signal scaling and small distortions, it has demonstrated substantially better results – 82.16% of correct predictions, therefore outperforming the baselines not only in this cross-dataset experiment, but also on the pure WISDM dataset (Table 3).

5. Computational performance

Convolutional Neural Networks have a highly parallelizable architecture, therefore they can perform data classification in a very efficient way. To estimate the exact velocity of the proposed solution, we used a pre-trained CNN for continuous time series classification and measured the throughput of the system. We have additionally measured the throughput of three alternative techniques described in previous sections. All algorithms were running on a machine with *Intel Xeon E5-2640 v3* 8-Core CPU and *Nvidia Titan X* GPU. The results are presented in Table 6 and they show the number of segments classified per second for each method. It can be observed that the performance of CNN is significantly higher compared to other solutions – the throughput almost reaches 150 thousand segments per second, while in other cases it is less than 10 thousand. The difference can be explained as following: during the prediction phase all operations performed by CNN can be written in terms of matrix multiplications and simple thresholding operations that can be parallelized on GPUs very efficiently, thus utilizing the full power of thousands CUDA cores of the graphics card. Other methods are based on non-matrix operations, and though some solutions for accelerating their performance were proposed in the literature [28,29], all current common implementations rely on CPU only.

Since *Tensorflow* machine learning library that was used for our CNN implementation is available for mobile devices, we have additionally tested the proposed solution in the wild – on a mid-range *Nexus 5X* Android smartphone. The architecture of the application was the following. The measurements from smartphone accelerometer and gyroscope were sampled at the rate 50 Hz similarly to [26], and the last 128 values for each channel were stored in a queue. A CNN pretrained on UCI dataset was embedded into the system and continuously performed time series classification: each time the previous prediction was finished, the last 128 values were taken again from the queue, centered and together with statistical features passed to the CNN for a new inference. The system was able to classify about 28 samples per second, which should be enough for real-time activity recognition where the predictions are updated 1–5 times/s. The throughput of mobile system is significantly lower compared to the results observed on the server, since in this case all computations were performed on a low-power phone CPU which performance is incomparable to both server CPUs and high-end GPUs. However, we should note that for devices equipped with the latest generation of *Snapdragon* mobile SoCs these results should be noticeably increased since they support GPU acceleration for *Tensorflow* models.

6. Conclusion

In this paper we proposed a solution for user-independent human activity recognition problem that is based on Convolutional Neural Networks augmented with statistical features that embrace global properties of the accelerometer time series. It has the benefits of using short recognition intervals of size up to 1 s and requiring almost no feature engineering and data preprocessing. Due to a relatively shallow architecture, the proposed algorithm has a small running time and can be efficiently executed on mobile devices in real time.

To evaluate the performance of the considered approach we tested it on two popular WISDM and UCI HAR datasets. The obtained results demonstrate that the proposed CNN-based model significantly outperforms baseline approaches and establishes state-of-the-art results in both cases. The cross-dataset experiment has further emphasized a platform-independent architecture that can be applied not only to different users, but to devices with different accelerometer calibrations.

References

[1] K. Kuspa, T. Pratkanis, Classification of Mobile Device Accelerometer Data for Unique Activity Identification, 2013.
[2] Jennifer R. Kwapisz, Gary M. Weiss, Samuel A. Moore, Activity recognition using cell phone accelerometers, SIGKDD Explor. Newsl. 12 (2) (2011) 74–82.
[3] J. Lockhart, Mobile sensor for data mining, Fordham Undergrad. Res. J. 1 (2011) 67–68.

[4] M. Zeng, L. Nguyen, B. Yu, O. Mengshoel, J. Zhu, P. Wu, J. Zhang, Convolutional neural networks for human activity recognition using mobile sensors, in: 6th International Conference on Mobile Computing, Applications and Services (MobiCASE), November 2014, 2014, pp. 197–205.
[5] B. Kolosnjaji, C. Eckert, Neural network-based user-independent physical activity recognition for mobile devices, in: IDEAL 2015: 16th International Conference, 2015, pp. 378–386.
[6] Andrey D. Ignatov, Vadim V. Strijov, Human activity recognition using quasiperiodic time series collected from a single tri-axial accelerometer, Multimed. Tools Appl. 1 (2015) 1–14.
[7] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: Computational Intelligence and Machine Learning, 2013.
[8] O. Lara, M. Labrador, A mobile platform for real-time human activity recognition, in: 2012 IEEE Consumer Communications and Networking Conference (CCNC), January 2012, Jan 2012, pp. 667–671.
[9] P. Siirtola, J. Roing, User-independent human activity recognition using a mobile phone: offline recognition vs. real-time on device recognition, in: Distributed Computing and Artificial Intelligence: 9th International Conference, 2012, pp. 617–627.
[10] A. Mannini, A. Sabatini, Machine learning methods for classifying human physical activity from on-body accelerometers, Sensors 10 (2) (2010) 1154–1175.
[11] M. Kose, O. Incel, C. Ersoy, Online human activity recognition on smart phones. Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data, 2012.
[12] 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: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI'15, 2015, pp. 3995–4001.
[13] F. Ordonez, D. Roggen, Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition, Sensors 16 (1) (2016) 115.
[14] S. Ha, J.M. Yun, S. Choi, Multi-modal convolutional neural networks for activity recognition, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics. October 2015, 2015, pp. 3017–3022.
[15] W. Jiang, Z. Yin, Human activity recognition using wearable sensors by deep convolutional neural networks, in: Proceedings of the 23rd ACM International Conference on Multimedia, 2015, pp. 1307–1310.
[16] C.A. Ronao, S.B. Cho, Evaluation of Deep Convolutional Neural Network Architectures for Human Activity Recognition with Smartphone Sensors, Yonsei University, 2015.
[17] C.A. Ronao, S.B. Cho, Human activity recognition with smartphone sensors using deep learning neural networks, Expert Syst. Appl. 59 (2016) 235–244.
[18] Charissa Ann Ronao, Sung-Bae Cho, Recognizing human activities from smartphone sensors using hierarchical continuous hidden Markov models, Int. J. Distrib. Sens. Netw. 13 (1) (2017), 1550147716683687.
[19] Y.J. Kim, B.N. Kang, D. Kim, Hidden Markov model ensemble for activity recognition using tri-axis accelerometer, in: 2015 IEEE International Conference on Systems, Man, and Cybernetics, Oct 2015, pp. 3036–3041.
[20] S. Seto, W. Zhang, Y. Zhou, Multivariate time series classification using dynamic time warping template selection for human activity recognition, CoRR (2015), abs/1512.06747.
[21] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. Reyes-Ortiz, Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine, in: Ambient Assisted Living and Home Care: 4th International Workshop, 2012, pp. 216–223.
[22] C.A. Ronao, S.B. Cho, Human activity recognition using smartphone sensors with two-stage continuous hidden Markov models, in: 2014 10th International Conference on Natural Computation (ICNC), Aug 2014, pp. 681–686.
[23] Y. Li, D. Shi, B. Ding, D. Liu, Unsupervised feature learning for human activity recognition using smartphone sensors, in: Mining Intelligence and Knowledge Exploration: Second International Conference, 2014, pp. 99–107.
[24] I. Masaya, I. Sozo, N. Takeshi, Deep Recurrent Neural Network for Mobile Human Activity Recognition With High Throughput, 2016, arXiv:1611.03607.
[25] WISDM's activity prediction dataset. http://www.cis.fordham.edu/wisdm/dataset.php.
[26] Human activity recognition using smartphones data set. https://archive.ics.uci.edu/ml/machine-learning-databases/00240.
[27] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, CoRR (2014), abs/1412.6980.
[28] Vincent Garcia, Eric Debreuve, Michel Barlaud, Fast k nearest neighbor search using GPU, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPRW'08, IEEE, 2008, pp. 1–6.
[29] Toby Sharp, Implementing decision trees and forests on a GPU, in: European Conference on Computer Vision, Springer, 2008, pp. 595–608.