Research Article

Jozsef Suto*

The effect of hyperparameter search on artificial neural network in human activity recognition

https://doi.org/10.1515/comp-2020-0227
received June 12, 2020; accepted April 13, 2021

Abstract: In the last decade, many researchers applied shallow and deep networks for human activity recognition (HAR). Currently, the trending research line in HAR is applying deep learning to extract features and classify activities from raw data. However, we observed that, authors of previous studies have not performed an efficient hyperparameter search on their artificial neural network (shallow or deep)-based classifier. Therefore, in this article, we demonstrate the effect of the random and Bayesian parameter search on a shallow neural network using five HAR databases. The result of this work shows that a shallow neural network with correct parameter optimization can achieve similar or even better recognition accuracy than the previous best deep classifier(s) on all databases. In addition, we draw conclusions about the advantages and disadvantages of the two hyperparameter search techniques according to the results.

Keywords: human activity recognition, artificial neural network, deep learning, Bayesian optimization, random hyperparameter search

1 Introduction

Human activity recognition (HAR) is a principal factor in human–machine interaction that made it possible to recognize muscle stimuli of humans. Now the modern technology offers a large amount of different sensing devices for capturing human's activity and behaviour. Depending on the sensor type, there are two principal research directions in HAR: vision-based and wearable sensor-based activity recognition. Due to the limitations and disadvantages of vision-based data capture, sensor-based activity recognition has received higher attention. In this work, we are focusing only on wearable sensor-based HAR.

HAR is a typical classification problem where inputs are inertial sensor signals (multi-variate time-series) and the output is an activity class label. Our final goal is to classify activities from the portions of sensor data streams. Since activity recognition is a time series problem, the main task for analysing sensor data streams is often to extract useful features from segmented data frames. Thereafter, the trained classifier model should classify those portions of data that cover patterns of interest. For portion designation, the predominant approach is the sliding window procedure, where the signal stream will be divided into frames by a fixed length sliding window [1,2]. Adjacent frames sometimes overlap to some degree but they are processed separately.

As in any pattern recognition problems, the keys of successful classification are the appropriate feature representation of the raw sensor signal, determination of the right classifier, and finding an efficient hyperparameter combination (classifier dependent). Although, many data capture devices exist and more information is provided to classifier algorithms, generally accepted rules of thumb for the aforementioned key points do not exist in the literature.

The recognition accuracy of HAR greatly depends on features extracted from raw signals. It is an important part of HAR, which has been studied for many years. In earlier works, features have been either time domain features, frequency domain features, or symbolic representations of sensor signals (typically auto regressive model coefficients). Features are calculated on the raw sensor data, independently for every frame extracted by a sliding window. This static or hand-crafted feature extraction approach is the conventional way of dealing with HAR features. However, suitable feature representation discovery requires fundamental application-specific knowledge.

The appearance of deep neural networks (DNNs) caused breakthroughs in numerous fields of machine learning such as object recognition and document analysis [3]. These methods can completely substitute

* **Corresponding author: Jozsef Suto,** Department of IT Systems and Networks, Faculty of Informatics, University of Debrecen, Debrecen, 4028, Hungary, e-mail: suto.jozsef@inf.unideb.hu

manually handcrafted feature extraction without any domain knowledge. In addition, they capture local dependencies of incoming signals and handles feature scale invariance. The advantages of deep techniques have become attractive for the HAR community very quickly and motivated HAR research to turn toward deeper classifier techniques such as convolutional neural network (CNN) and recurrent neural network [4]. Many researchers claimed that the new layers (e.g. convolutional) can replace static features [5,6]. However, the effectiveness of DNNs against a well-constructed shallow method is not clear in HAR. The studies of Gjoreski et al. [7] and Nguyen and Nguyen [8] reflect this uncertainty. In the first study, the authors compared efficiency of a CNN to other shallow techniques like random forest and support vector machine while in the second study more different CNN models have been compared with a shallow method. In both cases, the recognition accuracy between shallow and deep classifiers was rather small. In addition, we observed that most of the researchers have not applied detailed hyperparameter search in the case of parametric models. According to the aforementioned reasons, we compared the efficiency of a shallow (two layers) artificial neural network (ANN) with previously used DNNs on five public HAR databases. In order to find an appropriate parameter combination, we applied both random and Bayesian hyperparameter optimizations.

The rest of the paper is organized as follows. Section 2 gives a short description about the five used HAR databases. In Section 3, several earlier results will be presented on the five databases and at the end their weakness(es) will be summarized. Section 4 provides a detailed description about the used shallow ANN, and the two hyperparameter optimization methods. Section 5 contains the experimental results with the random and Bayesian parameter optimization on all datasets. Section 6 summarizes and analyses the result achieved by the two parameter search techniques. Finally, Section 7 concludes the article.

2 Databases

Currently, numerous public datasets are available for scientific HAR purposes. Those databases provide an excellent chance for qualitative comparison of machine learning techniques. Our goal was to use such public databases where data acquisition devices were inertial sensors, include daily activities, ensure different representations of data, and enough previous results exist on them. According to those criteria, five datasets have been used as data sources.

2.1 Skoda mini checkpoint

The Skoda dataset consists of ten manipulative gestures plus one unknown (null class) activity. Those gestures are a subset of 46 activities of quality control checkpoints. It has been recorded with 10–10, 3-axis accelerometers placed on the left and right arm of one person who performed car maintenance. The sampling frequency of each sensor was approximately 98 Hz. The recording was about 3 h long, while the person performed 70 repetitions per gesture. In this dataset, measurement values are arranged in matrix format where each line represents a sample. In the original data files, a line consists of a label, a sensor id, and the measurement values of all sensor x, y, and z axes.

This dataset is freely available for HAR research purposes (http://har-dataset.org/doku.php?id=wiki:dataset).

2.2 HAR using smartphone dataset

The second dataset has been downloaded from the University of California, Irvine (UCI) Machine Learning Repository. In this database, the activities have been carried out with 30 volunteers within an age range of 19–48 years. Each person performed six elementary activities such as sitting, standing, and walking. All of them wore a smartphone with one built-in 3-axis accelerometer and one 3-axis gyroscope on their waist. The sensor signals are divided into frames with a 2.56-s-long, fixed-width sliding windows. In the rest of the article, we will refer to this database as HAPT. Unlike other databases that were used in this work, it contains both raw sensor data and extracted features. In the second case, the dataset provides 10,929 instances with 561 attributes (real values).

2.3 Actitracker

This database was created with the consent of 59 subjects who performed 6 distinct daily activities such as walking, jogging, sitting and standing. The sensing device was a mobile phone placed in the subjects' pocket. The sampling

frequency was 20 Hz. For data acquisition, the authors used their own Actiracker Android-based application and recorded the raw sensor signals of the phone 3-axis accelerometer. This dataset includes 1,098,207 raw time series measurements with six attributes. The dataset can be downloaded from the Wireless Sensor Data Mining (WISDM) laboratory's webpage (http://www.cis.fordham.edu/wisdm/dataset.php).

2.4 PAMAP2

The PAMAP2 dataset is also freely available from the UCI machine learning repository. It consists of 12 lifestyle activities such as standing, walking lying down, and sport exercises like running and Nordic walking. The whole dataset is made up of accelerometer, gyroscope, magnetometer, temperature, and heart rate samples which have been recorded from wearable sensors located on the hand, chest, and ankle over 10 h. In this database, all rows contain 54 columns with the following information:

- 1 timestamp (real),
- 2 activity ID (int),
- 3 hearth rate (real or Nan),
- 4–20 IMU hand (real or Nan),
- 21–37 IMU chest (real or Nan),
- 38–54 IMU ankle (real or Nan).

2.5 Opportunity

The Opportunity dataset contains human activities related to usual kitchen activities using multiple sensors. Data are recorded by 4 subjects with 12 wearable sensors attached to various positions of the body at a frequency of 30 Hz. For each subject, the dataset contains six different runs: five runs about activities of daily living (ADL) and one drill session. During ADL runs subjects performed morning activities, like drinking coffee, preparing sandwich or cleaning up without any restriction. In the drill sessions, each subject performed 20 repetitions of a pre-defined sequence of activities. The used sensors include some object sensors, body-worn sensors, and ambient sensors. There are 72 sensors of 10 modalities in total. Five sensors were on the subject's upper body and two were mounted on their shoes. The acceleration sensors are placed on the upper body, hip, and leg. In the database all .dat files contain data matrices in text format. All lines consist of approximately 250 columns of timestamp, labels, and sensor readings. This dataset has been released for the Opportunity Challenge where organizers defined four different tasks targeting the recognition of locomotion (Task A), activity spotting (Task B1), arm gestures (Task B2), and robustness to noise (Task C).

3 Related work

To recognize human activities, several classifier techniques have been tested for specific application purposes. In the last few years typically, deep methods have been used to automatically recognize what and when an activity has been performed by an observed person. Many researchers claim that deep methods provide an effective tool for multi-level feature extraction from high-dimensional data and eliminate the need for designing static features.

In several previous HAR works, authors have not specified hyper-parameters. Probably they did not pay enough attention for parameter search. In other works, parameters were constant values or they came from a partial grid. However, there are some more sophisticated parameter optimization techniques than those [9]. The performance of shallow and deep networks significantly depends on their parameters. The parameter tuning process often depended on personal experience that is hard to quantify or describe. Hammerla et al. [10] also pointed this issue. They observed that authors usually omit the details about parameter search thus overall training process remains unclear and difficult to replicate.

In the article of Jiang and Yin [6], the CNN contained five 5×5 kernels on the first convolutional layer, followed by a 4×4 mean-pooling. The second convolutional layer contained ten 5×5 kernels, followed by 2×2 mean-pooling. Finally, the network ended in two fully connected layers. The input of the classifier was an "activity image" in which raw signals have been stacked row by row onto each other. With this classifier the authors measured 95.18% recognition accuracy (number of correctly classified test samples divided by all test samples) on the HAPT dataset.

In the work of Ronao and Cho [11], the CNN input was 1D sensor signal. Unlike several other studies in their work a detailed description can be found about CNN's hyperparameters. They applied a greedy-wise tuning on hyperparameters wherein they modified the number of layers, number of feature maps, filter size, and pooling size. Other parameters remained constant values. The learning rate was 0.01, the weight decay value was

0.00005, and the number of epochs was 5,000. The authors have found that after 130 feature maps and three convolutional layers the performance does not increase. With the best parameter combination, they achieved 95.75% recognition rate on the HAPT dataset.

Zeng et al. [12] among the first researchers tested an own CNN architecture which consists of three disjoint convolutional and max-pooling layers and three fully connected layers on the Skoda, Opportunity, and Actitracker datasets. In the case of Opportunity and Skoda datasets, they used only one inertial sensor (from the right arm) to identify activities. Moreover, in the Opportunity database they focused only on 11 activities including 10 low-level and 1 unknown activities. Their CNN model achieved 88.19%, 76.77%, and 96.88% classification accuracy on the Skoda, Opportunity, and Actitracket datasets, respectively. Later Zeng et al. [5] attempted semi-supervised learning from both labelled and unlabelled data because they supposed that semi-supervised learning provides better predictions for similar activities compared to supervised learning using only labelled data. As classifier evaluation metrics the authors used accuracy and mean F-measure or in other words the mean $F1$ score (1) where P_i and R_i refer to the precision and recall of the ith class and C indicates the number of classes. They used a reference CNN to the study which achieved 93.84% recognition accuracy on the Actitracker and 0.922 F_m score on the PAMAP2 dataset. In the latest article, Zeng et al. [19] developed a RNN model (from Long Short-Term Memories – LSTM) for HAR because this type of model has a considerable pattern recognition capability in sequential data streams. Their model had 1 LSTM layer with 128 dimensions hidden representation. The model training was performed by ADAM with 0.05 learning rate. They used three databases as data sources including PAMAP2 and Skoda. On the Skoda dataset, the authors focus only on ten accelerometers placed on the right arm while on PAMAP2 they used the data of subjects 5 and 6 for validation and test, respectively. Under these circumstances they achieved 0.899 and 0.938 F_m scores on PAMAP2 and Skoda.

Ordonez and Roggen [4] created a deep network which combines convolutional and recurrent layers. The convolutional layers acted as feature extractors and provided abstract representations of the input sensor data, while the recurrent layers modelled temporal dynamics of feature map activations. They segmented both datasets into mini-batches of a size of 100 data segments. The learning rate was 0.001, the weight decay factor was 0.9, and the dropout probability of randomly selected units was 0.5. They evaluated their model on the Skoda and Opportunity datasets. Their experiments were restricted to ten sensors placed on the right arm. The original sample rate was decimated to 30 Hz on Skoda. Under these conditions they measured 0.958 weighted $F1$ (or F_w) score (2) where $w_i = n_i/N$ (N is the total number of samples, while n_i is the number of samples in the class) is the proportion of the ith class. Here P_i, R_i, and C are the precision, recall, and number of classes similarly as in (1). On the Opportunity database, they recognized locomotion and gesture with and without null class according to the opportunity challenge. Their results were 0.895 and 0.93 F_w scores on locomotion recognition with and without null class and 0.915 and 0.866 F_w on gesture recognition with and without null class:

$$F_m = \frac{1}{C}\sum_{i=1}^{C} 2\frac{P_i R_i}{P_i + R_i}, \tag{1}$$

$$F_w = \sum_{i=1}^{C} 2w_i \frac{P_i R_i}{P_i + R_i}. \tag{2}$$

In the work of Nguyen and Nguyen [8], the CNN model consisted of two convolutional and pooling layers and some fully connected layers. In the model, the number of filters was 64 and 128 on the first and second layer, respectively. The filter width was five samples and the stride was one sample on each convolutional and pooling layer. The dimension of fully connected layers was 500. The probability of dropout was 0.5. They also worked on the Skoda dataset but they restrict their experiments to a single sensor, worn on the right arm. In addition, in their work data have been down-sampled to 48,Hz. With those conditions they achieved 0.871 F_m score on Skoda.

Ha et al. [13] thought that the existing CNN approaches which are considered 1D kernel do not appropriate to take advantage of multimodal sensors. Therefore, they proposed a CNN with 2D kernel because 2D kernels can effectively capture spatial dependencies over sensors. This CNN model reached 97.92% recognition accuracy on Skoda.

Alsheikh et al. [14] tested a Deep Belief Network (DBN) on the Skoda and Actitracker databases. In their model hidden units are formed from Restricted Boltzmann Machines (RBM) and trained in a layer-by-layer fashion. The classifier was trained with stochastic gradient decent with a mini-batch size of 75. On the first layer, the learning rate was 0.001 and the number of epochs was 150. On the next layers, the number of epochs was 75 with 0.01 learning rate. Finally, the fine-tuning learning rate was 0.1 and the number of epochs was 1,000. This model reached 98.23 and 89.38% recognition accuracy on Actitracker and Skoda. In the case of Skoda, the authors used only one accelerometer for the experiment.

Hammerla et al. [10] tried CNN, RNN, and a multilayer shallow network with different architectures on the Opportunity and PAMAP2 datasets. The authors performed a grid search on learning rate, learning rate decay, dropout probability, etc. On PAMAP2 the validation set was the 5'th subject's first and second recordings, while the test set was the same recordings of the 6th subject. The remaining data have been used for training. In the case of Opportunity, the data source was accelerometer recordings from the upper limbs, back, and IMU data from both feet. The validation set was run 2 from subject 1 while the test set was runs 4 and 5 from subject 2 and 3. The remaining data were used for training. They achieved 0.937 F_m score on PAMAP2 with a CNN and 0.745 F_m and 0.927 F_w scores on Opportunity with an RNN.

Münzner et al. [15] tested different sensor fusion options and achieved 0.86 ± 0.034 (standard deviation) F_m score on the PAMAP2 dataset with a two-layered (convolutional) CNN. Kasnesis et al. [16] arranged raw sensor signals into 2D form and by a conventional CNN they achieved 97.25% and 88.56% accuracy on the HAPT and PAMAP2 datasets. Finally, Murahari and Plötz [17] extended the DeepConvLSTM model of Ordonez and Roggen with an attention layer and they measured 0.875 ± 0.002, 0.707 ± 0.003, and 0.913 ± 0.004 (±statistical significance) F_m scores on PAMAP2, Opportunity, and Skoda, respectively. They applied learning rate decay, dropout, and the learning rate was fixed to 0.001.

For better clarity, the aforementioned results are summarized in Table 1. In the case of Opportunity only one study has been used as reference because just this work aligned (approximately) to the Opportunity Challenge [18] and inside it to locomotion recognition (Task A) without null class. From the detailed literature review, some important statements can be drawn:

- Although data sources were the same public databases, the dataset usage was distinct.
- Different authors used different hyperparameters (number and types of layers, architecture deepness, filter size and stride, weight regularization, initial learning rate, learning decay, etc.) in their deep network design.
- Only a few research tried grid hyperparameter search. Most of them did not use any parameter search.
- Without a proper hyperparameter search, we cannot be sure that, we have reached the maximum performance of the classifier model.
- There are much more sophisticated parameter search techniques than hand and grid search.

4 Methods

4.1 Data segmentation and feature extraction

Movement and gesture data collected with inertial sensors are multivariate time-series data with relatively

Table 1: Previous results on the five datasets

Database	Reference	Classifier	Acc. (%)	F_m score	F_w score
Skoda	Zeng et al. (2014) [12]	CNN	88.19	—	—
Skoda	Zeng et al. (2018) [19]	RNN	—	0.938	—
Skoda	Ordonez and Roggen (2016) [4]	DeepConvLSTM	—	—	0.958
Skoda	Nguyen and Nguyen (2017) [8]	CNN	—	0.871	—
Skoda	Ha et al. (2015) [13]	CNN	97.92	—	—
Skoda	Alsheikh et al. (2016) [14]	DBN	89.38	—	—
Skoda	Murahari and Plötz (2018) [17]	DeepConvLSTM + Att.	—	0.913 ± 0.004	—
HAPT	Jiang and Yin (2015) [6]	CNN	95.18	—	—
HAPT	Ronao and Cho (2016) [11]	CNN	95.75	—	—
HAPT	Kasnesis et al. (2018) [16]	CNN	97.25	—	0.972
Actitracker	Zeng et al. (2017) [5]	CNN	93.84	—	—
Actitracker	Zeng et al. (2014) [12]	CNN	96.88	—	—
Actitracker	Alsheikh et al. (2016) [14]	DBN	98.23	—	—
PAMAP2	Zeng et al. (2017) [5]	CNN	—	0.922	—
PAMAP2	Zeng et al. (2018) [19]	RNN	—	0.899	—
PAMAP2	Hammerla et al. (2016) [10]	CNN	—	0.937	—
PAMAP2	Münzner et al. (2017) [15]	CNN	—	0.86 ± 0.034	—
PAMAP2	Kasnesis et al. (2018) [16]	CNN	88.56	—	0.887
PAMAP2	Murahari and Plötz (2018) [17]	DeepConvLSTM + Att.	—	0.875 ± 0.002	—
Opportunity	Ordonez and Roggen (2016) [4]	DeepConvLSTM	—	0.93	—

high spatial and temporal resolution (e.g. 10–250 Hz). The classification of these data is typically following a machine learning chain approach. The first link in the chain is to segment the time series data into discrete frames (or windows) through a sliding-window segmentation process. Although the segmentation process is slightly different in deep and shallow classifiers, the main idea is the same in both cases. Thereafter, the following step in a typical HAR machine learning chain is the feature extraction. One of the key differences between deep and shallow machine learning techniques is their feature extraction method. Shallow techniques require different hand-crafted features in order to they be efficient. In this case, window's content feeds feature extractors where the goal is to characterize the raw content of the window. From all frames, a set of features is extracted and most commonly the set consists of statistical and/or frequency domain features. According to the characteristic of windows they will be classified as belonging to different types of activities.

In earlier papers, researchers applied different feature extraction approaches because the HAR literature does not suggest a generally accepted feature group [1]. In this study, we used the following statistical, frequency domain, and auto-regressive features (on those databases where it was relevant): *mean, variance, mean absolute deviation, interquartile range, 75'th percentile, kurtosis, signal magnitude area, min–max difference, spectral energy, spectral centroid, principal frequency, and first and second auto-regressive coefficients*. This decision has been motivated by the work of Suto et al. [20], in which the authors collected and analysed the most popular features in HAR. Explanation and equations of features can be found in their article. Features were normalized with the standard scaling because it transforms features onto the same scale.

4.2 ANN architecture

Neural network theory proposes guidelines to the network design but more possible architectures exist. For example, in the current state of the art the cross-entropy, quadratic cost, and log-likelihood functions are the most popular error (cost) functions [21]. For weight regularization also more possibilities exist such as $L1$ normalization, $L2$ normalization, and dropout. Adapted to the error function, different activation functions can be used on the hidden and output layers.

In the work of [22], the authors compared three shallow ANN architectures on two public HAR databases with different activation and error functions. They observed that hyper-parameters are more important than the network architecture because all architectures achieved similar recognition accuracy while a given architecture produces significantly different recognition rates with different hyper-parameter combinations. According to their results, we used the following settings in our implementation. A structure of the network can be seen in Figure 1.

- Number of layers: 2 (hidden and output)
- Number of hidden neurons: 50, 65, or 80
- Learning algorithm: stochastic gradient descent
- Weight initialization: according to (3) normal distribution where $\mathbf{W}^l$ is the weight matrix and κ is the number of inputs of a neuron on the lth layer
- Activation functions: tangent (4) and linear (5) on the hidden and output layers, respectively, where η is the cumulative input of a neuron
- Learning decay: exponential (6) where ϕ is the decay factor, ε is the epoch counter, and α_0 is the initial learning rate
- Error function: mean square error with $L2$ regularization (7) where M is the number of output neurons, N is the number of samples, y_j is the target, a_j is the activation of the output neuron, while λ and ω refer to the regularization strength and weights
- Epoch limit: 1,000
- Mini batch size: 10 samples
- Stop condition: no improvement in 10 epochs

$$\mathbf{W}^l \in N\left(0, \frac{1}{\sqrt{\kappa}}\right), \tag{3}$$

$$\sigma_h(\eta) = \frac{\exp(\eta) - \exp(-\eta)}{\exp(\eta) + \exp(-\eta)}, \tag{4}$$

$$\sigma_o(\eta) = \eta, \tag{5}$$

$$\alpha = \alpha_0 \exp(-\phi\varepsilon), \tag{6}$$

$$C(\boldsymbol{y}, \boldsymbol{a}) = \sum_{j=1}^{M} (y_j - a_j)^2 + \frac{\lambda}{2N} \sum_{\omega} \omega^2. \tag{7}$$

4.3 Hyperparameter search

Although there is a large body of literature on global optimization, in machine learning, hyperparameter optimization is typically carried out by hand, by grid search, by random search, or by Bayesian optimization. The most important parameters of a shallow network are number of hidden neurons (h), initial learning rate (α_0), learning rate decay factor (ϕ), and regularization strength (λ). In 2012, Bergstra and Bengio [23] empirically and

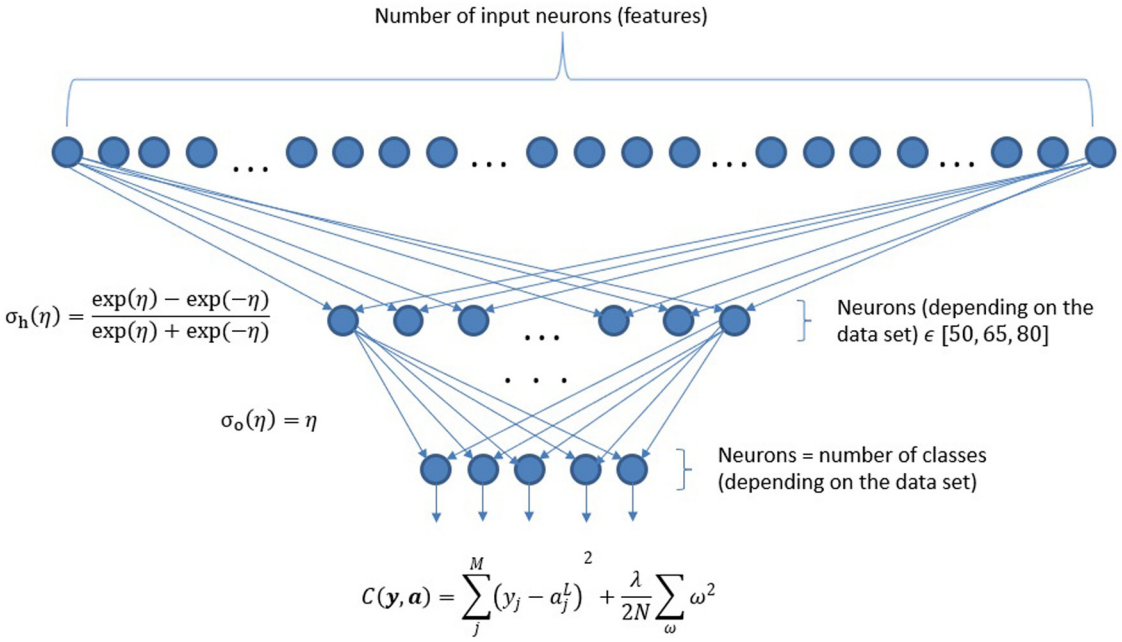

Figure 1: Structure of the ANN.

theoretically showed that randomly chosen trials are more efficient for hyperparameter optimization than grid or manual search. Grid search suffers from the curse of dimensionality, while the major drawback of manual search is the reproduction difficulty. Reproducibility is important for the progression of scientific research and for the ease of application of machine algorithms by non-expert users.

From the aforementioned optimization techniques, probably the Bayesian is the most efficient. It is a powerful method for finding the extrema of an objective function when its evaluation is costly. This optimization technique estimates the objective function with a surrogate function and follows the exploration and exploitation approach during sampling of the search space. Before we perform Bayesian optimization, we have to make two decisions. First, about the prior over functions that will express assumptions about the function being optimized. Second, we must choose an acquisition (sampling) function which is used to determine the next point to evaluate. In our work, the Bayesian optimization is strongly based on the tutorial of Brochu et al. [24]. According to their work we used a Gaussian process (GP) (8) as prior distribution which can be specified by its mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ [25]:

$$f(\boldsymbol{x}) \sim \mathrm{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')). \tag{8}$$

Initially, the prior mean was the zero-function $m(\mathbf{x}) = \mathbf{0}$ while the covariance function was the popular squared exponential (9). The squared exponential kernel (Gaussian kernel) measures similarity between two real-valued parameters $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$. In the kernel l determines the smoothness of the function while θ controls the vertical variation:

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \theta \exp\left(-\frac{1}{2l^2}\|\boldsymbol{x_i}, \boldsymbol{x_j}\|\right). \tag{9}$$

At the beginning of the optimization process some training points must be available $\boldsymbol{D}_{1:n} = \boldsymbol{x}_{1:n}, y_{1:n}$ and the optimizer has to decide what $\boldsymbol{x}_{n+1}$ should be chosen next. Then according to the properties of GP, $y_{1:n}$ and $y_{n+1} = f(\boldsymbol{x}_{n+1})$ are jointly Gaussian. So, the posterior distribution can be described with the following formula (10):

$$P(y_{n+1}|\boldsymbol{D}_{1:n}, \boldsymbol{x}_{n+1}) = N(\mu(\boldsymbol{x}_{n+1})\sigma^2(\boldsymbol{x}_{n+1})). \tag{10}$$

The acquisition function should take consideration of regions where both mean and variance of the surrogate function are high (maximum search). The literature offers more possible acquisition functions such as Probability of Improvement, Expected Improvement, and GP Upper Confidence Bound. We used the popular Expected Improvement [26]. Once a new result is available, the model is updated and the acquisition function is also recomputed. After that, the next iteration of the optimization loop can be started.

Although the efficiency of Bayesian optimization is indisputable, there are some limitations that have prevented it from becoming a widely used hyperparameter

optimization method in machine learning [24,27]. For instance, it has a time-consuming optimization procedure. The determination of sampling points may vary significantly in duration and this should be taken into account. Furthermore, the design of prior strongly influences the final outcome. Finally, we should find a trade-off between exploration and exploitation. Therefore, in this work we used both Bayesian and random searches.

5 Experimental results

5.1 Experimental conditions

For each database, we adjusted the most common experimental conditions. We utilized the whole database (and not just its fraction) of Skoda, HAPT, and Actitracker. Independently of the database we have not used overlap between windows.

In the PAMAP2 dataset, temperature and hearth rate readings were omitted from the study because we focused only on inertial signals. In the Opportunity dataset, we tried to classify locomotion from the full set of body-worn sensors similarly as Task A in Opportunity Challenge. It is a 4-class activity recognition problem. The test dataset was ADL4 and ADL5 from subjects 2 and 3. The training dataset was all ADL runs of subject 1 and ADL1, ADL2, and Drill sessions from subjects 2 and 3. The validation set was ADL3 of subjects 2 and 3.

In the case of HAPT, the creators of the database applied 2.56-s-long windows (128 samples) and they extracted several features that were similar features as listed in Section 4.1. In addition, they defined test and training datasets. In our work, we used 85% of the original training set for training, 15% of the original training set for validation, and the test set as it is.

The window size in Skoda and PAMAP2 dataset was 128 samples wide. In the Actitracker, it was 256 samples wide, while in the Opportunity database the sliding window was 500 ms (16 samples) long. The size of windows was a power of two in all databases according to the fast Fourier transformation's requirement.

5.2 Random parameter search

As mentioned earlier, the most important parameters of a shallow networks are the number of hidden neurons (h), learning rate decay factor (ϕ), initial learning rate (α_0), and regularization strength (λ). At first, random hyperparameter search has been used. The test framework (ANN and parameter search) was written in Python (without any acceleration), and all experiments were performed on a notebook with Intel Core I7-4510U CPU and 16GB RAM. The search for α_0 and λ took place on a 10-base exponential scale where the exponent comes from uniform distribution (11). The number of hidden neurons was drawn randomly from another uniform distribution $h \in U(10, 500)$. For visualization purpose, the hyperparameter search has been restricted to two parameters in both parameter search techniques. Therefore, the learning decay factor was kept on the constant 0.003 value. It means that a learning period is limited by 1,000 epochs and a constant learning decay. In the first scenario, the initial learning rate and the number of hidden neurons were randomly selected while weight regularization was kept on 0.001.

$$\alpha_0, \lambda \in 10^{U(-6,1)}. \tag{11}$$

As an example, the outcome of 100 test runs on the HAPT dataset can be seen in Figure 2. We should highlight that the initial learning rate axis on the above figures is a 10-base logarithmic scale due to (11). After the examination of our first random parameter search, two interesting observations can be drawn (those can also be observed in Figure 2):

- Initial learning rate has a higher effect on the performance than the number of hidden neurons in a two-layer ANN independently of the dataset.
- Above 80 hidden neurons, the performance improvement of the two-layer ANN is very small independently of the dataset.

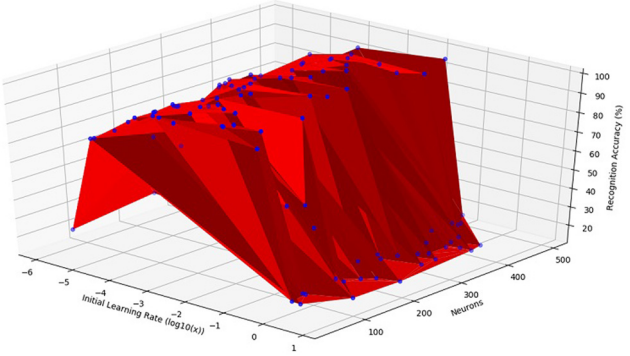

Figure 2: Random parameter search of hidden neurons and initial leaning rate on the HAPT dataset.

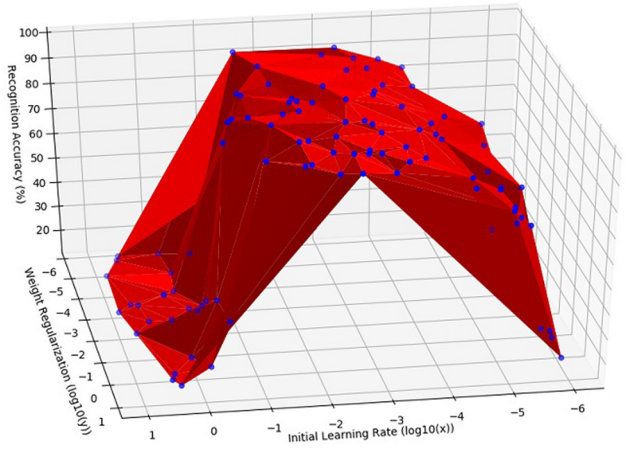

Figure 3: Random parameter search of initial leaning rate and weight regularization on the HAPT dataset.

In the second random parameter search scenario, the initial learning rate and the weight regularization strength were randomly selected. Taking into account the previous observations the number of hidden neurons was 50 in the case of Skoda, and PAMAP2 while it was 65 in the Actitracker, HAPT, and 80 in the Opportunity datasets. In the second random parameter search (again 100 runs), the tendency was similar as before – the initial learning rate had a higher effect on the recognition rate than weight regularization strength independently of the dataset. Again, as an example Figure 3 shows the outcome of the second random search on HAPT dataset. Now the weight regularization axis is also a 10-base logarithmic scale due to (11).

After the second random search, the best (caused the highest performance) learning rate and weight regularization combination have been selected. With the best parameter combination ten test runs have been performed on each database where the only difference between runs is the weight initialization. After test runs, each classifier evaluation metric has been calculated and this result (and the training time in seconds) can be found in Table 2.

5.3 Bayesian optimization

In our optimization framework, the l and θ kernel parameters (9) were 0.2 and 1.0, respectively. Moreover, we supposed that the samples from the objective function are corrupted with a very small noise $y_n = f(\boldsymbol{x}_n) + \varepsilon_n$ where $\varepsilon_n \sim N(0, \sigma^2_{\text{noise}})$. With this assumption the posterior distribution can be described in the following form (12), where $\mu(\boldsymbol{x}_{n+1})$ and $\sigma^2(\boldsymbol{x}_{n+1})$ are also influenced by noise:

$$P(y_{n+1}|\boldsymbol{D}_{1:n}, \boldsymbol{x}_{n+1}) = N(\mu(\boldsymbol{x}_{n+1}), \sigma^2(\boldsymbol{x}_{n+1}) + \sigma^2_{\text{noise}}). \quad (12)$$

Bayesian optimization can be seen as "black box" optimization, which typically requires that all dimensions have bounds on the search space. In our case, we supposed that the search space is a hyper-rectangle of dimension 2 (initial learning rate, weight regularization strength). Our experiences have shown that the number of possible elements inside the search space (test points) is a very important point of this optimization method. At first, we tried to choose test points according to the exponential scale as in (11). Unfortunately, in this case the performance of GP regression was poor. Thereafter, 150 evenly spaced numbers were selected as test points from the $[10^{-6}, 10^{0}]$ interval on both dimension (150×150 test points). The 150 evenly spaced points may seem small but the posterior distribution determination is very memory-intensive, so we should set our search space scale according to the available hardware resources. Before the real optimization steps, ten points were randomly selected from the search space where the objective function has been sampled. Those points were the initial training points. For example, the result of Bayesian optimization on the HAPT database can be seen in Figure 4. Now both axes are on normal scale. The number above points on the figure indicates the iteration step of the optimizer (from 0 to 9 are the initial points).

Summary about the optimization process on all databases can be found in Table 3. The iteration column contains the number of iteration steps until the best point

Table 2: Evaluation metrics of the two-layer ANN classifiers

Database	Acc. (mean ± std)	Max Acc.(%)	F_m (mean ± std)	Max F_m	F_w (mean ± std)	Max F_w	Time (s)
Skoda	96.7 ± 0.8	97.8	0.953 ± 0.016	0.967	0.967 ± 0.009	0.978	6,597
HAPT	96.2 ± 0.4	97.0	0.957 ± 0.005	0.964	0.965 ± 0.005	0.969	23,157
Actitracker	98.3 ± 0.4	98.9	0.966 ± 0.008	0.977	0.983 ± 0.004	0.988	4,997
PAMAP2	89.8 ± 0.6	90.6	0.831 ± 0.024	0.873	0.896 ± 0.007	0.905	18,076
Opportunity	73.4 ± 15.22	92.7	0.634 ± 0.217	0.884	0.687 ± 0.207	0.923	78,018

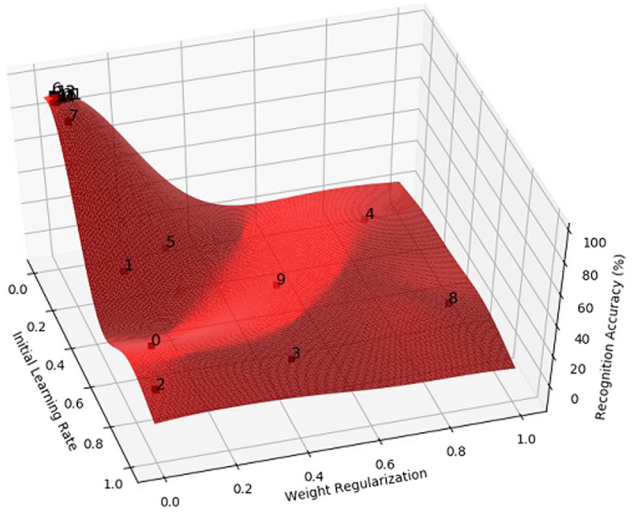

Figure 4: The result of Bayesian optimization on the HAPT dataset.

Table 3: Summary about Bayesian optimization process on all databases

Database	Iterations	Max Acc. (%)	Time (s)
Skoda	14	97.3	8,116
HAPT	7	96.7	4,163
Actitracker	10	98.7	5,466
PAMAP2	11	89.8	7,288
Opportunity	14	89.1	3,6850

has been reached. Independently of the data set, the convergence of the optimizer was rather fast. In addition, if we compare the maximum classification accuracies (other evaluation metrics were omitted for the sake of clarity) in Tables 2 and 3, we can see that the difference is very small except in the case of Opportunity dataset. Finally, if we take a look at the time requirement of random search and Bayes optimization, we can conclude that in the case of bigger datasets the Bayesian optimization is a more time-efficient choice while in other cases 100 random trials took less time.

6 Discussion

In this work, we highlighted the importance of hyperparameters in neural network design. If we compare our results with previous results in Table 1, we will see that our shallow networks achieved similar performance to the previous best deep network on all databases. Our results also demonstrate that, a two-layer shallow network in combination with random or Bayesian optimization (on four databases out of five with our settings) can be a better choice for activity classification than a deep method. Based on the experimental results (depicted in figures), we can conclude that the following hyperparameters can be a good initial setting of such an ANN that we used: $h < 100$, $\alpha_0 < 10^{-3}$, and $\lambda \sim 10^{-2}$. However, a Bayesian or random parameter search (depending on the dataset) is necessary for the hyperparameters to be correct.

Deep networks already outperformed shallow techniques in several machine learning problems but this is not fulfilled now. Maybe one possible reason can be the lack of parameter search. Another possible reason is the small data size. Deep networks are very sensitive to training set size and they require much bigger training data set in order to their weights be correctly set. Unfortunately, data acquisition in HAR is inconvenient and time-consuming process, and it is especially true for elder population. Our opinion is that most of the current HAR articles follow a "brute force tendency." It means that, researchers create deeper and deeper networks and they expect performance improvement from complexity. However, increasing the network does not always bring the expected result. For example, Saez et al. [2] and Suto and Oniga [28] noted that an incorrect deep topology significantly affects the result. In addition, the time requirement of a deep architecture would be enormous on the current Internet of Things (IoT) and wearables sensor networks [29]. In many cases, putting more complexity in the architecture does not cause any significant improvement but it requires more training time.

According to the results in previous sections, we can compare random and Bayesian optimization in HAR. After the comparison, we can conclude that the Bayesian optimization is more time-efficient than 100 random trials when the evaluation of the objective function is costly (bigger dataset). The main disadvantage of Bayesian method with GP comes from the heavy memory requirement of posterior distribution generation. It defined a boundary of the search space resolution, thus the outcome of Bayesian optimization was not always as efficient as random search.

7 Conclusion

This study compared the efficiency of a two-layer ANN in combination with random and Bayesian hyperparameter search with previous deep network-based techniques in

the activity recognition problem. The main motivation of this work was an observation about inefficient hyperparameter search in previous articles.

At the beginning of the article, we presented five public databases which were the data sources in this study. Thereafter, several relevant studies have been presented where the authors used deep network as classifier and worked on at least one of the five databases. Later, the used static features, the structure and the parameters of the shallow network, and the parameter optimization methods have been presented.

At first, we used random parameter search. In the first scenario, the initial learning rate and the number of hidden neurons were selected randomly. According to the results, two conclusions have been drawn which are true on each dataset. First, the initial learning rate has a higher effect on the recognition rate than the number of hidden neurons in a two-layer ANN. Furthermore, above 80 hidden neurons the performance improvement of the two-layer ANN is very small. In the second random parameter search scenario, the initial learning rate and the weight regularization strength were randomly selected. The number of hidden neurons was 50 in the case of Skoda, and PAMAP2 datasets, 65 in the Actitracker and HAPT datasets, and 80 in the Opportunity. In the second random parameter search, the tendency was similar as before. The training time requirements of 100 random trials were 6,597 s on Skoda, 2,315 s on HAPT, 4,997 s on Actitracker, 18,076 s on PAMAP2, and finally 78,018 s on Opportunity. With the best parameter combination, ten test runs have been performed on each database and three evaluation metrics have been calculated from its outcome. The highest recognition accuracies were 97.8% on the Skoda database, 97.0% on HAPT, 98.9% on Actitracker, 90.6% on PAMAP2, and 92.7% on Opportunity.

In the second step of the investigation, we used Bayesian optimization for parameter search. Before the real optimization steps, ten points were randomly selected from the search space where the objective function has been sampled. Those points were the initial training points. The convergence of Bayes optimizer was rather fast independently of the dataset. On bigger databases the Bayesian optimization is more time-efficient than 100 random trials, but its memory requirement allowed "low" resolution in search space.

According to our measurements and other previous results, the final conclusion of this study is that, the usage of a shallow two-layer ANN with proper handcrafted features and appropriate hyper-parameters can achieve similar performance to deep networks on public HAR datasets. One of the main reasons for this phenomenon is the relatively small data size. This observation can also be useful in all classification problems where the available data size is similar or less.

Acknowledgment: This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

Conflict of interest: Author states no conflict of interest.

Data availability statement: The datasets analysed during the current study are available in the UCI repository (https://archive.ics.uci.edu), and on the two following websites: http://har-dataset.org/doku.php?id=wiki:dataset, http://www.cis.fordham.edu/wisdm/dataset.php.

References

[1] O. D. Lara, M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Commun. Surv. Tut.*, vol. 15, pp. 1192–1209, 2013

[2] Y. Saez, A. Baldominos, P. Isasi, "A comparison study of classifier algorithms for cross-person physical activity recognition," *Sensors*, vol. 17, p. 66, 2017.

[3] P. Y. Simard, D. Steinkraus, J. C. Platt, "Best practice for convolutional neural networks applied to visual document analysis," In: *7th International Conference on Document Analysis and Recognition* (*6 Aug. 2003, Washington, USA*), Washington, 2003, pp. 958–962.

[4] F. J. Ordonez, D. Roggen, "Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, pp. 1–25, 2016.

[5] M. Zeng, T. Yu, X. Wang, T. L. Nguyen, O. Mengshoel, "Semi-supervised convolutional neural networks for human activity recognition," In: *2017 IEEE International Conference on Big Data* (*11–14 Dec. 2017, Boston, USA*), Boston, 2017, pp. 522–529.

[6] W. Jiang, Z. Yin, "Human activity recognition using wearable sensors by deep convolutional neural networks," In: *23th ACM International Conference on Multimedia* (*13 Oct. 2015, Brisbane, Australia*), Brisbane, 2015, pp. 1307–1310.

[7] H. Gjoreski, J. Bizjak, M. Gjoreski, M. Gams, "Comparing deep a classical machine learning methods for human activity recognition using wrist accelerometer," In: *25th International Joint Conference on Artificial Intelligence* (*9–15 July 2016, New York, USA*), New York, 2016, pp. 1–7.

[8] T. T. T. Nguyen, N. D. Nguyen, "Experiments on deep learning for wearable activity recognition," *Southeast–Asian J. Sci.*, vol. 5, pp. 101–110, 2017.

[9] F. Hutter, L. Kotthoff, J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*, Springer Nature, UK, 2019.

[10] N. Y. Hammerla, S. Halloran, T. Plots, "Deep, convolutional, and recurrent models for human activity recognition using wearables," In: *25th International Joint Conference on Artificial*

Intelligence (*9-15 July 2016, New York, USA*), New York, 2016, pp. 1533–1540.
[11] C. A. Ronao, S. B. Cho, "Human activity recognition with smartphone sensors using deep learning neural networks," *Expert. Syst. Appl.*, vol. 59, pp. 235–244, 2016.
[12] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, et al., "Convolutional neural networks for human activity recognition using mobile sensors," In: *6th International Conference on Mobile Computing and Services* (*6 Nov. 2014, Austin, USA*), Austin, 2014, pp. 197–205.
[13] S. Ha, J. M. Yun, S. Choi, "Multi-modal convolutional neural networks for activity recognition," In: *IEEE International Conference on Systems, Man, and Cybernetics* (*9-12 Oct. 2015, Hong Kong, China*), Hong Kong, 2015, pp. 3017–3022.
[14] M. A. Alsheikh, A. Selim, D. Niyato, L. Doyle, S. Lin, H. P. Tan, "Deep activity recognition models with triaxial accelerometers," In: *AAAI Workshop: Artificial Intelligence Applied to Assistive Technologies and Smart Environments* (*25 Jan. 2016, Phoenix, USA*), Phoenix, 2016, pp. 8–13.
[15] S. Münzner, P. Schmidt, A. Reiss, M. Hanselmann, R. Stiefelhagen, R. Dürichner, "CNN-based sensor fusion techniques for multimodal human activity recognition," In: *2017 ACM International Symposium on Wearable Computers* (*11–15 Sep. 2017, Hawaii, USA*), Hawaii, 2017, pp. 158–165.
[16] P. Kasnesis, C. Z. Patrikakis, I. S. Venieris, "PerceptionNet: A deep convolutional neural network for late sensor fusion," In: *Intelligent Systems Conference* (*6–7 Sep. 2018, London, UK*), London, 2018, pp. 1–9.
[17] V. S. Murahari, T. Plötz, "On attention models for human activity recognition," In: *International Symposium on Wearable Computers* (*8–12 Oct. 2018, Singapore*), Singapore, 2018, pp. 100–103.
[18] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. R. Millan, et al., "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recogn. Lett.*, vol. 34, pp. 2033–2042, 2013.
[19] M. Zeng, H. Gao, T. Yu, O. J. Mengshoel, H. Langseth, I. Lane, et al., "Understanding and improving recurrent networks for human activity recognition by continuous attention," In: *International Symposium on Wearable Computers* (*8–12 Oct. 2018, Singapore*), Singapore, 2018, pp. 56–63.
[20] J. Suto, S. Oniga, P. Pop-Sitar, "Feature analysis to human activity recognition," *Int. J. Comp. Commun.*, vol. 12, pp. 116–130, 2017.
[21] M. A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015, Ebook, http://neuralnetworksanddeeplearning.com/, Accessed 18 January 2020.
[22] J. Suto, S. Oniga, "Efficiency investigation of artificial neural networks in human activity recognition," *J. Ambient. Intell. Human. Comput*, vol. 9, pp. 1049–1060, 2017.
[23] J. Bergstra, Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.
[24] E. Brochu, V. M. Cora, N. Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," arXiv:http://arXiv.org/abs/arXiv:1012.2599, 2010.
[25] C. E. Rasmussen, "Gaussian process in machine learning," In: *Summer School on Machine Learning*, Springer, Berlin, Heidelberg, 2003, pp. 63–71.
[26] J. Bergstra, D. Yamins, D. D. Cox, "Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures," In: *30th International Conference on Machine Learning* (*16-21 Jun. 2013, Atlanta, USA*), Atlanta, 2013, pp. 115–123.
[27] J. Snoek, H. Larochelle, R. P. Adams, "Practical Bayesian optimization for machine learning algorithms," In: *Advances in Neural Information Processing Systems* (*3–8 Dec. 2012, Lake Tahoe, USA*), Lake Tahoe, 2012, pp. 2951–2959.
[28] J. Suto, S. Oniga, "Efficiency investigation from shallow to deep neural network techniques in human activity recognition," *Cogn. Syst. Res.*, vol. 54, pp. 37–49, 2019.
[29] J. Suto, S. Oniga, C. Lung, I. Orha, "Comparison of offline and real-time human activity recognition results using machine learning techniques," *Neural Comput. Appl.* 32, pp. 15673–15686, 2018, https://doi.org/10.1007/s00521-018-3437-x.