# Machine learning methods in Smartphone-Based Activity Recognition

Istvan Pintye[1,2]
[1]Laboratory of Parallel and Distributed Systems,
Institute for Computer Science and Control
(SZTAKI), Hungary
[2]Doctoral School of Applied
Informatics and Applied Mathematics,
Óbuda University, Hungary
istvan.pintye@sztaki.hu

***Abstract*— In this paper, we present a system for human physical Activity Recognition (AR) using smartphone with embedded sensors. This paper addresses the question whether there is a comfortable way to predict human activities based on collected data from smartphone embedded gyroscope and accelerometer. Computational background of this work based on self-learning machine learning methods. In order to train the machine learning algorithms, The University of California, Irvine (UCI) dataset was used and the different models were compared. After selecting the best model further modifications were suggested in order to improve the accuracy of the model. At the end 96.88% accuracy was reached.**

***Keywords*— *Human activity recognition, Machine Learning, Artificial Neural Net, Python, Sci-Kit Learn, Keras***

## I. Introduction

Due to the evolution of microelectronic and computer systems sensors are now built in almost all smartphones. This fact leads to the recognition that these smart devices can actively monitor our daily lives. Moreover, monitoring the vital signs of patients have become one of the top highlighted research areas in the last few years. The increased life expectancy and thus the increasing elderly population in many countries, additionally the sometimes not satisfying medical care in rural areas led to the need for monitoring the physical activity of the patients.

Today's incurable diseases also pose a question to the medical society i.e.: till how much extent the aforementioned diseases are genetically predestined, or they are just consequences of our way of living. Enormous amount of studies [1], [2], [3] focuses on human activity recognition which can help to discover the correlation between everyday life and certain diagnosis. This study can be regarded as a base for future medical surveys.

The aim of this study is to show that with the help of today widespread used tools available at affordable prices and sophisticated machine-learning methods easily monitorable data can serve as a base in determining objectively our daily activity. The study uses the Human Activity Recognition dataset [4], [1] that is freely available in an already pre-processed format. The data was collected with smartphones fixed at participants' waist.

This paper presents a basic research carried out with a systematic investigation of different machine learning methods in a human activity recognition scenario. Systematic investigation or grid search works rely on high computation resources. We delivered our application of such computing resources onto cloud computing environment that provide flexible and affordable solutions. Moreover, we are now witnessing the worldwide spreading of lightweight web-based research technologies such as Jupyter Notebook with Python kernel. Such technologies enable easy portability and management of complex software stacks and long-running simulation scenarios, such as finetuning of numerical modelling. Outputs can be accessed by any web browser, with even hand-held mobile devices, or even on remote sites with low band data link coverage.

Extending this work later with more sophisticated data collection mechanisms the relationships/correlation of our lifestyle, way of living and the potential risk for certain illnesses can be concluded.

The structure of the paper is as follows: First the data set will be described in section II., followed by the description of the applied machine learning models and fine-tuning solutions in details in Section III. The evaluation of the results is presented in section IV. Finally, we conclude our work and also some further application has been suggested. The Jupyter Notebook program code can be obtained from the corresponding author.

## II. Data set

### A. Human Activity Recognition Using Smartphones Dataset

The Human Activity Recognition database was built from the recordings of 30 volunteer participants aged between 19 and 48 performing activities of normal daily living (ADL) while carrying a waist-mounted smartphone which was equipped with embedded inertial sensors. Everybody performed six different activities (WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphones Samsung Galaxy S II on their waist. The dataset does not contain any information about the gender, age or socio-economical data of the volunteers. All volunteers used the same Galaxy S II equipment. To better understand the experiment the data collections looked like showed in this short video [5]. During the experiments with the help of the built-in accelerometer and a gyroscope the students of University of California captured triaxial linear acceleration and triaxial angular velocity at a constant rate of 50 Hz. The experiments were video recorded and later labelled manually. The collected dataset has been randomly partitioned into training set and test set, with 70% of the volunteers generating the training data and 30% of them produced the test data.

978-1-7281-7377-1/20/$31.00 ©2020 IEEE

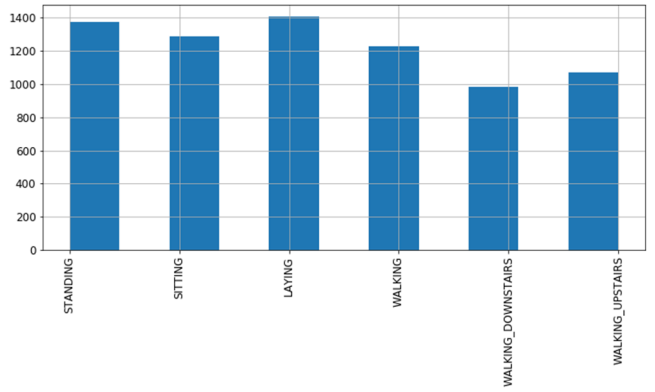


Fig. 1. Distribution of the different activities in the training samples, values of observations (n)

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cut off frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

### B. *Attribute Information*

For each record in the dataset the following is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- An identifier of the subject who carried out the experiment.
- Its activity labels.
- Features are normalized and bounded within [-1,1].

### C. *Short Data Analysis*

A quick F-Test analysis on the data set showed us that there are a lot of features which show significant differences between the activities. Without selecting the possible candidates all the features were used in my further analysis as explanatory variable. Because the data was very well prepared – there was no missing labels and features were transformed into the range of [-1;+1], analysis started with discovering the distribution of thevariables. The distribution of the training samples was quite well equalized as shown in Fig. 1.

Fig. 2 shows the statistics of each activity counted by study volunteers. It can be easily observed that walking downstairs and walking upstairs moves in much smaller range than any other activity. What does that mean? We have seen earlier that the number of activities is almost equally distributed. But while laying, standing and sitting varies in a much larger range than the other three activities.

TABLE I. DISTRIBUTION OF ACTIVITIES BY SUBJECTS

| | *average* | *median* | *std.deviation* |
|---|---|---|---|
| laying | 66.85 | 70.50 | 11.72 |
| standing | 65.75 | 64.00 | 11.75 |
| sitting | 61.20 | 61.00 | 11.38 |
| walking | 58.05 | 57.00 | 10.43 |
| walking down | 46.20 | 46.50 | 5.06 |
| walking upstairs | 50.40 | 51.00 | 5.83 |

Probably it does not have to affect our prediction, but keep in mind the fact that some participants have a lot of laying activity record while others have only just a few. If we look at the distribution of the walking downstairs than we can see that all the subjects have a small amount of records of this kind of action and the variation between the participants is much smaller.

## III. APPLIED MACHINE LEARNING METHODS

Six machine learning methods were tested with default settings. These were all part of the Python Sci-kit Learn package [6], namely: Decision Tree Classifier, K-Neighbors Classifier, Support Vector Classifier, Gaussian Naïve Bayes Classifier, Quadratic Discriminant Analysis Classifier, Multilayer Perceptron Classifier. As far as the target variable was discreet and categorical, accuracy (=number of correct predictions / numbers of all predictions) was used as a measurement of the goodness of the models.

The Dummy classifier, which is also a built-in function in machine-learning frameworks, was used to measure baseline performance. Dummy Classifier is a classifier that makes predictions using simple rules that can be set by the user. Below is the baseline performance using the "most_frequent" strategy, which always predicts the most frequent label in the training set. The model that is developed in this study is expected to show better performance than the 0.182219 (18.22%) accuracy.

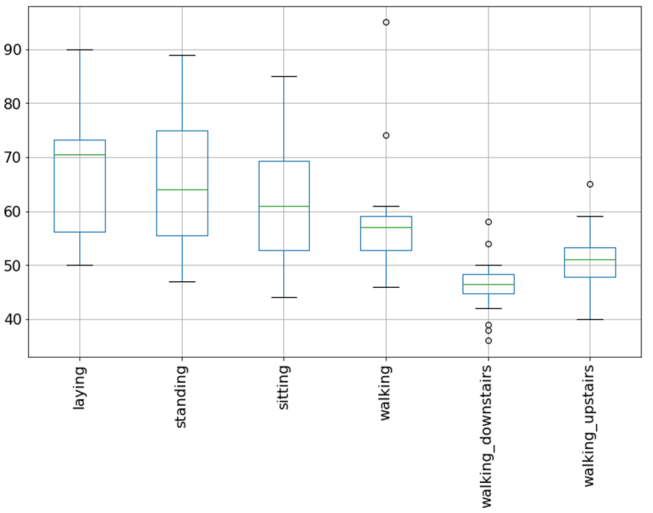


Fig. 2. Samples count by activities of the training set

TABLE II. ACCURACY OF THE MACHINE LEARNING MODELS

| Rank | | |
|---|---|---|
| | *Machine learning method* | *Accuracy* [a] |
| 1 | MultyLayerPerceptron Calassifier | 0.94095 |
| 2 | Support Vector Classifier | 0.93077 |
| 3 | Decision Tree Classifier | 0.86155 |
| 4 | K-Neighbors Classifier | 0.80726 |
| 5 | Gaussian Naïve Bayes Classifier | 0.77027 |
| 6 | Quadratic Discriminant Analysis Classifier | 0.73566 |

[a] Accuracy measured on training set.

The results of the six machine learning methods mentioned above were compared by their accuracy using the same training set. The results were sorted by the accuracy in Table II. Based on the values represented in the table the Multilayer Perceptron Classifier reached the highest accuracy on the training set, while Support Vector Classifier also showed a reasonable accuracy value. Therefore, this study tries to find the best parameter settings for these two Classifiers.

### A. Hyper Parameter fine tuning

Fine-tuning of a machine learning model requires immense computational resources; however, such capacities are usually available on non-homogeneous IT platforms. In addition, development and operational application are typically performed on different, heterogeneous systems (from laptops to cloud computing environments). For this experiment MTA Cloud infrastructure was used with one Virtual Machine having 8 vCPU core, 16 GB RAM, 500 MB HDD with Ubuntu 16.04 Operational system was deployed by Occopus [7] orchestration system .

The goal of our hyperparameter searching is to find the best hyperparameters for MLP and SVC respectively. Python Sci-kit learn package gives us an opportunity to define a grid search in a quite easy way. After the parameters were defined the GridSearchCV [8] module did the rest for us. Various numbers of parameters were tested for both MLP and SVC models.

TABLE III. MULTY LAYER PERCEPTRON CLASSIFIER

| Rank | **Best parameter set found: {'activation': 'logistic', 'hidden_layer_sizes': (250, 100)}** | | |
|---|---|---|---|
| | *'hidden layer'* | *'activation'* | *Accuracy* [b] |
| 1 | (250, 100) | logistic | 0.951 (+/-0.054) |
| 2 | (250,100) | ReLU | 0.937 (+/-0.030) |
| 3 | (100,) | logistic | 0.929 (+/-0.049) |
| 4 | (50, 20) | logistic | 0.929 (+/-0.058) |
| 5 | (20,) | logistic | 0.928 (+/-0.049) |
| 6 | (20,) | ReLU | 0.928 (+/-0.060) |
| 7 | (100,) | ReLU | 0.926 (+/-0.049) |
| 8 | (50,20) | ReLU | 0.923 (+/-0.051) |

[b] Accuracy measured on training data set.

TABLE IV. SUPPORT VECTOR CLASSIFIER

| Rank | **Best parameter set found: {'C': 1, 'kernel': 'linear'}** | | |
|---|---|---|---|
| | *'C'* | *'kernel'* | *Accuracy* [a] |
| 1 | 1 | linear | 0.936 (+/-0.056) |
| 2 | 0.1 | linear | 0.936 (+/-0.054) |
| 3 | 20 | linear | 0.933 (+/-0.057) |
| 4 | 100 | linear | 0.931 (+/-0.055) |
| 5 | 100 | rbf | 0.888 (+/-0.108) |
| 6 | 20 | rbf | 0.887 (+/-0.102) |
| 7 | 1 | rbf | 0.885 (+/-0.052) |
| 8 | 0.1 | rbf | 0.800 (+/-0.153) |

[a] Accuracy measured on training set.

In the case of the Multilayer Perceptron model such as the number of the neurons on the hidden layer, the number of the hidden layers and the activation function that can be Rectified Linear Unit or Logistic (in another framework is named Sigmoid) varied while in the case of the Support Vector Classifier the value of the 'c' and the type of the kernel functions {linear, radial} were the possible hyperparameters. Table III and IV shows the results respectively

Table III. shows the results of the fine tuning of the MLP model. Results are sorted by accuracy. The higher the value the better the estimation. The values in the brackets show the variation of the accuracy from test to test. In this case a 10-fold cross validation experiment was executed so the accuracy value can be interpreted as a mean of the 10 experiments, while the values between the brackets are the variance of the mean accuracy. Table IV. shows the similar results in the case of the SVM model and it can be observed that all the Multilayer Perceptron solutions with different hyperparameters resulted over 92 percent accuracy.

Comparing the two tables (Table III. with Table IV.) reveals that the MultiLayer Perceptron model with 250 neurons on the first and 100 neurons on the second hidden layer had a slightly better accuracy than the SVC model.

Since the MLP model gave the best result in the following part of the paper we focus further enhancement possibilities of this model.

The original data set was split into training and test set. Training set contains 7352 sample with labels, the test data set has 2947 samples.

Confusion matrix is a very powerful tool in a hand of a data analyst. It compares the predicted values against the 'ground truth' values. For example, in this case 53 'sitting' action were classified or predicted by the Multilayer Perceptron classifier as a 'standing' action. Of course, this is a huge mistake. In other words, 10.8 percent of all the 'sitting' action was misclassified. On the contrary, 'laying' was predicted perfectly. The following confusion matrix in Fig. 3 shows the result of this best configuration on the test set. The overall accuracy on the test dataset was 95.147 percent.

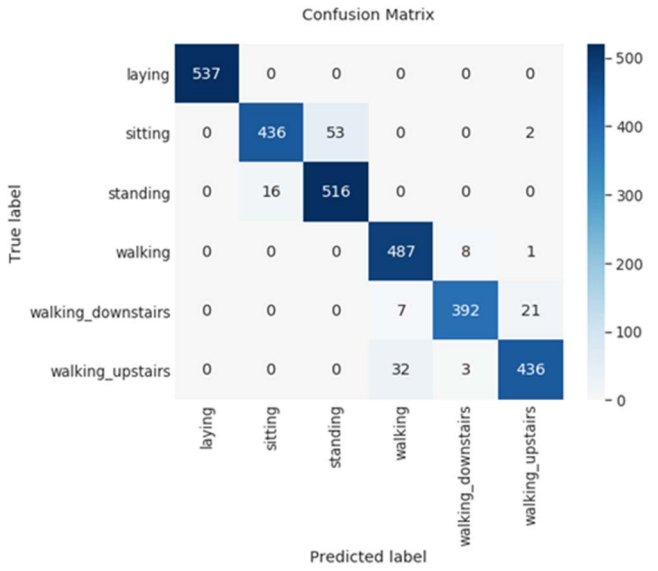


Fig. 3. Confusion Matrix calculated on the test data set (NN(250,100))

Multilayer Perceptron Classifier - reached more than 95% accuracy on the training set. After adjusting the parameter of the Neural Network, further enhancements are available.

### *B. Avoiding overfitting the model*

Finally, turned out that the best parameter settings led to worse result than the fourth-best parameter (NN (50,20)) setting on the test set. This phenomenon simply can be caused by the bigger number of parameters of the bigger model which also can lead to the problem of overfitting. Overfitting a model means that the model fits too closely to the training data and therefore the performance on a new (test) data set will be much worse. To eliminate the problem of overfitting the use of dropout layers seems to be a good idea. Therefore, two additional dropout layers were introduced into the model. The dropout layers were added to the final model based on Table V. and it really turned out to have better accuracy.

In order to see how to evolve the performance of the neural network I had to change the Sci-Kit learn module to the Keras – TensorFlow module. In the following section I will show how to improve this accuracy, how to make it even better than 95%.

TABLE V. NEURAL NETWORK ARCHITECHTURE

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Dense1 | (None, 50) | 28150 |
| dropout1 | (None, 50) | 0 |
| Dense2 | (None, 20) | 1020 |
| dropout2 | (None, 20) | 0 |
| Dense3 | (None, 6) | 126 |
| Total params: 29,296<br>Trainable params: 29,296<br>Non-trainable params: 0 | | |

TABLE VI. MULTI LAYER PERCEPTRON CLASSIFIER

| ***Label*** | ***precision*** | ***recall*** | ***support*** [d] |
|---|---|---|---|
| Laying | 1.00 | 0.99 | 537 |
| Sitting | 0.97 | 0.89 | 491 |
| Standing | 0.89 | 0.98 | 532 |
| Walking | 0.94 | 0.98 | 496 |
| Walking Downstairs | 0.98 | 0.94 | 420 |
| Walking Upstairs | 0.96 | 0.94 | 471 |

There were 50 neurons used in the first hidden layer and 20 neurons in the second hidden layer with sigmoid activation function. After each hidden layer a 10% dropout layer was inserted. because this dropout layer provided the best results. Adaptive Momentum (Adam) optimizer [9] and softmax cost function were used. In this case the Keras library was used instead of SciKit-Learn package, and a model looks as it is shown in Table V. shown.

## IV. EVALUATION OF THE FINAL MODEL

The input of the first dense layer was 562 the same as the number of features. Fig. 4 shows the values of the accuracy on the training set and on the test (or validation set) during the training. Finally, this model reached 96.88% accuracy on the test set.

Accuracy is only one aggregated measurement to express the goodness of an estimation, but that are also other metrics to analyze the prediction by the categories of target variable as well. In Table VI. the precision(also called positive predictive value is the fraction of relevant instances among the retrieved instances), the recall (refers to the percentage of total relevant results correctly classified by your algorithm) and support (support is the total number of each activity) values can be compared for the six different activities.

.

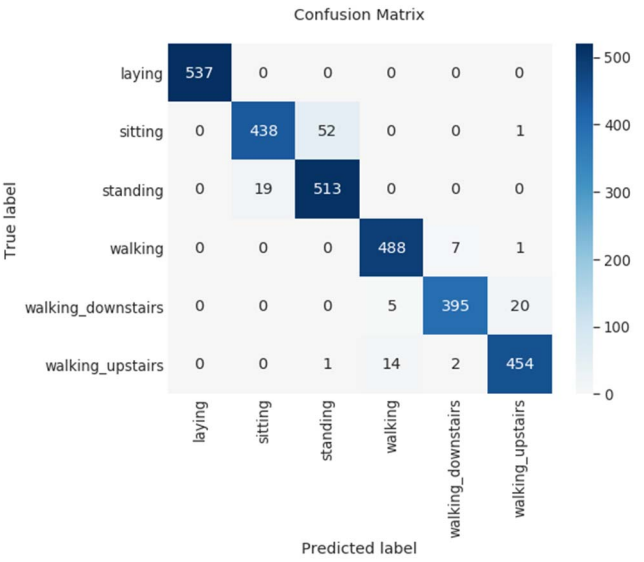


Fig. 4. Confusion Matrix for the test with NN(50,D(0.1),20,D(0.1))

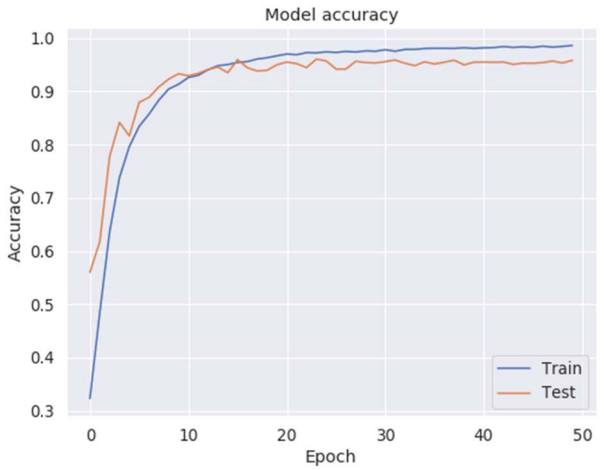


Fig. 5. Accuracy on the training and test sets (NN (Sig(50), dropout (0.1), Sig(20), dropot (0,1), softmax(6))

Fig. 4 also represents a confusion matrix after the new model and the dropout layers were introduced. It can be seen that this time the values in the blue squares are higher than in the case of Fig. 3, which means that this model predicted the activities with more accuracy.

Fig. 5 compares the change of the accuracy on both the training and the test set. After 50 epochs the accuracy was around 96.88%. This is a slightly better result and the value of the accuracy could not be increased with longer training period. As Fig. 5 shows the accuracy of the training set (blue) is still raising, while the accuracy of the test set stop increases after 20 epochs. The distance between the accuracy of the test set and the accuracy of training set is stayed constant. There is a clear sign that there is no need to train further the model.

Finally, let us compare the true and the predicted activities for a randomly picked participants. Fig. 6 shows the results for our MLP model with two dropout layers with the real values from the data set and predicted values.

On the vertical axis the activities are presented, while the horizontal axis shows the timeline. Blue line shows the real activity, while black dotted line shows the predicted value. As we can see the prediction is not perfect, but the error is not radically wrong. Most of the time only similar activities are misclassified.

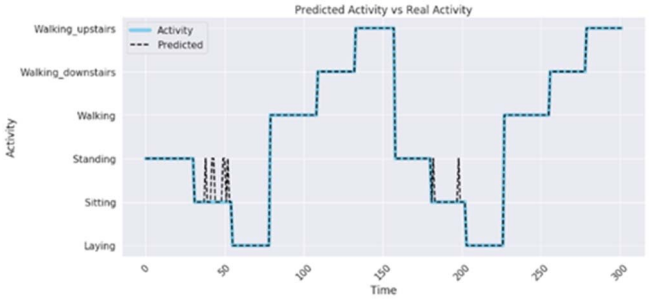


Fig. 6. Predicted activities vs real activities

## V. Conclusion and Future Work

In this paper different machine learning methods were systematically compared on a human activity recognition dataset. After the two best methods were selected [Multi Layer Perceptron Classifier, Support Vector Classifier] further parameter tuning was conducted.

The parameter search lead to a slightly better solution, namely it gave 95.147 accuracy, but it is important to note that the accuracy of the basic model was also quite high. In order to further improve the accuracy of the model we used dropout layers on our best configured MLP model to avoid the so-called overfitting phenomenon. With this extension we managed to reach 96,88 % accuracy. Our computations were carried out on MTA cloud with the help of Jupyter notebook.

The contribution of our work is to support patient and doctors with objective and reliable data about their physical activity because people tend to mislead the doctors and themselves as well with subjective feelings about their way-of-living. Later, this study can be extended by other vital signs in order to obtain better diagnosis or research results.

As our future work we would like to select the most important variables in order to create a more simple or simplified model. During this study the whole dataset containing 561 variables were used and fed them into the neural network. In our next experiment we will wrap these models in a feature selection loop, like genetic algorithm or other feature selection methods based on statistics. Furthermore, these preliminary results could be used to monitor patients with certain illnesses, like old men with heart diseases to know whether he had enough activity during the day or not.

## Acknowledgment

The presented work was partially funded by the European H2020 NEANIAS project under grant No. 863448, and by the Hungarian Scientific Research Fund (OTKA) under project No. 132838. We thank for the usage of MTA Cloud (https://cloud.mta.hu/) that significantly helped us achieving the results published in this paper.

## References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, *A Public Domain Dataset for Human Activity Recognition Using Smartphones*. .

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic."

[3] J. L. Reyes-Ortiz, A. Ghio, D. Anguita, X. Parra, J. Cabestany, and A. Català, *Human Activity and Motion Disorder Recognition: Towards Smarter Interactive Cognitive Environments*. .

[4] "UCI Machine Learning Repository: Human Activity Recognition Using Smartphones Data Set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones. [Accessed: 10-Feb-2020].

[5] "Activity Recognition Experiment Using Smartphone Sensors. - YouTube." [Online]. Available: https://www.youtube.com/watch?v=XOEN9W05_4A. [Accessed: 20-Apr-2020].

[6] "API Reference — scikit-learn 0.22.1 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/classes.html. [Accessed: 10-Feb-2020].

[7] J. Kovács and P. Kacsuk, "Occopus: a Multi-Cloud Orchestrator to Deploy and Manage Complex Scientific Infrastructures," *J. Grid Comput.*, vol. 16, no. 1, pp. 19–37, Mar. 2018.

[8] "sklearn.model_selection.GridSearchCV — scikit-learn 0.22.1 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Accessed: 10-Feb-2020].

[9] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.