Machine Learning Algorithms Implementation into Embedded Systems with Web Application User Interface

Kamil Židek, Ján Piteľ, Alexander Hošovský
Department of Industrial Engineering and Informatics
Technical University of Košice, Faculty of Manufacturing Technologies with a seat in Prešov
Bayerova 1, Prešov, Slovakia
kamil.zidek@tuke.sk, jan.pitel@tuke.sk, alexander.hosovsky@tuke.sk

***Abstract*—The paper presents research in usability of web technologies for implementation of machine learning and clustering algorithms into embedded systems. The research work is divided into two main parts. The first part is devoted to designing backend system with fast C++ application for learning execution model. The second part of application is frontend based web application with PHP and AJAX to provide interface for virtual laboratory access via internet. This solution is implemented and tested on selected embedded systems (Orange PI Lite, Raspberry PI3).**

***Keywords*—web application; machine learning; clustering; virtual laboratory**

I. Introduction

Nowadays web applications technologies begin to replace standard desktop applications. HTML language is not very suitable for web applications but in combination with client/ server scripting languages it provides a good tool for creating dynamic and interactive application. Client languages provide scripting for user interface (JavaScript) and server scripting (PHP, ruby, ASPX, python) execution of computational intensive code on server (C, C++ binary code) and provides security of private code. The fast response and partial refresh of web page is acquired by AJAX technology. Dynamic web application development using different approaches and software tools are described for example in [1], [2]. The current progress in dynamic web application is described in [3].

In present time storing large dataset of data with web access on clouds are very popular and can be processed online. The designed embedded web nodes can be used in concept of Industry 4.0 for data mining and knowledge extraction from production process. This is very useful and often used in power engineering for example, for monitoring of combustion processes [4], [5], heat production and distribution [6] with possibilities using computational intelligence for process evaluation [7]. Web applications are also used in virtual laboratories for sharing scientific interface with researches and students for free [8], [9], [10]. This article introduces that it is possible to create web application for high computationally intensive application with combination of C++ language and deploy them to embedded system with low consumption and relative small dimension.

II. Used Algorithms

The processing of large dataset in machine vision systems consists of two basic operations: clustering of acquired data and creating of model for classification of new data.

A. Clustering algorithms

Main group of algorithms used in clustering are quantization of vectors (K-mean) [11], hierarchical clustering (extended modification FLANN) [12] and density based algorithm (DBSCAN) [13]. FLANN and DBSCAN algorithms need calculate distances between samples and K-means algorithm computes centroid for every cluster.

Basic clustering algorithms use usually Euclidian distance only, equation (1) is usually used for distance metrics:

$$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}\ . \qquad (1)$$

The most resistant algorithm for samples with noise and very specific distribution is DBSCAN (Density-Based Spatial Clustering of Applications with Noise), which doesn't need initial specification of number of clusters. This algorithm is suitable especially for nonlinear distributed clusters.

Fig.1 shows the example of usability DBSCAN algorithm for clustering data to clusters with specific 2D data distribution and separated noise data (gray points) which are excluded from clusters.

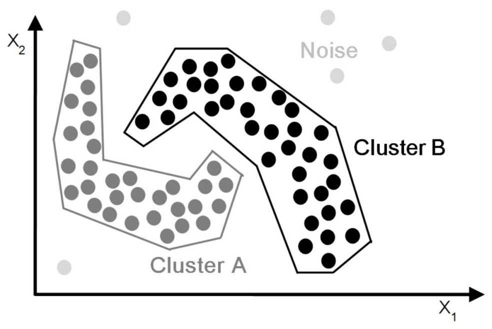


Fig. 1. Example of DBSCAN algorithm for clustering.

B. *Machine learning algorithms for classification*

After clustering procedure we have output represented by set of clusters. The next task is to design some mathematical model which describes these clusters and can be used for fast sorting of new samples without repeated process of clustering. This part of process is divided into two tasks: training and testing of classification. After this evaluation process we can use the designed system for data mining in the cloud.

There are many algorithms for machine learning and next classification: Support Vector Machines (SVM) [14], Normal Bayesian Classifier (NBC) [15], K-Nearest Neighbor (KNN) [16], Trees Gradient Boosted (GBT) [17], Random Trees (RT) [18] and Artificial Neural Networks (ANN) [19].

The simplest one is support vector machine algorithm which can be used only for simple clusters distribution in data space. The most computationally expensive are algorithms based on artificial neural networks, but they have very low tendency to error creation in new sample classification. Their main disadvantage is memory size difficulty with bigger numbers of neurons in network, because embedded systems are very limited in this feature (usually they have 0.5-1 GB RAM integrated without possibility to extend).

Support Vector Machine algorithm is based on searching an optimal boundary for dividing data into clusters. In simple case a boundary line can be used which is described by equation (2):

$$w \cdot x - b = 0\,. \tag{2}$$

Naive Bayes Classifier is based on statistical theory of probability Gaussian element distribution according to equation (3):

$$p(C_k|x) = \frac{p(C_k) \cdot p(x|C_k)}{p(x)}, \tag{3}$$

where $p(x/C_k)$, $p(\mathrm{C_k})$ and $p(x_k)$ are calculated using training data.

Example of implementation of clustered data to input and output of neural network is shown in Fig. 2.

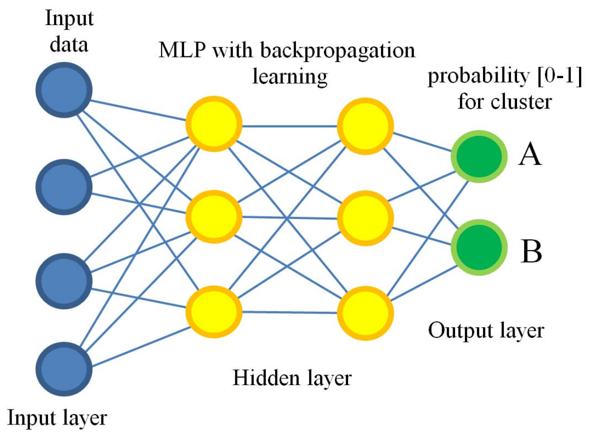

Input data
MLP with backpropagation learning
probability [0-1] for cluster
A
B
Output layer
Hidden layer
Input layer

Fig. 2. Using of artificial neural network for classification.

III. Principle of Implemented Algorithm

We used essential set of clustering and classification algorithms and its implementation in C++. Detailed description of whole implementation process can be distributed to 3 main processes [20]:

A. *Data import*

The first task is to collect data, usually from realtime production process, databases and cloud storages.

B. *Data clustering*

Then there is need of basic data distribution into groups which have some similar properties. It requires to test clustering algorithms, to compare reliability of used algorithms and to select the most suitable algorithm for clustering used data.

C. *Classification / Prediction*

This process involves:

- learning and creating of mathematical model by data from clustering with set of classification methods,
- comparison of classification methods and selection the most reliable to clustered data,
- testing of new samples using classification model for new reliable prediction.

Fig. 3 shows concept of combination all proposed algorithms to one part.

The web application allows import data from external file or it is possible to create simple data set for testing in loaded image with randomization automatically. It is supposed that you can use data in 2D space and position [x, y] with extra parameters, for example width, length, orientation. Currently we can integrate five separate parameters by user interface in web application.

In Fig. 4 there is simple graphical representation of usability of clustering and machine learning algorithms for data mining process which can be implemented into software of embedded device.

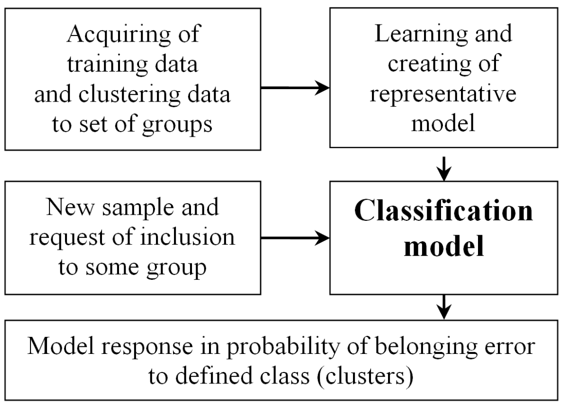

Acquiring of training data and clustering data to set of groups
Learning and creating of representative model
New sample and request of inclusion to some group
Classification model
Model response in probability of belonging error to defined class (clusters)

Fig. 3. The technique of clustering, classification and prediction.

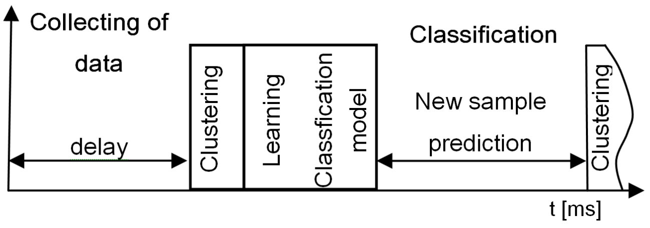


Fig. 4. Graphical representation of algorithms usability.

IV. Frontend, Backend for User Interface

The main server with Linux OS provides support for connection from local network to web nodes as virtual workplaces and connect embedded devices to one subnetwork. We use Apache web server with proxy to connect all embedded systems to one local network. Embedded systems have own web server to provide user interface for computational core written in C++. Machine learning and clustering method are implemented in OpenCV module machine learning (ML), which provides command line text output.

User interface uses HTML technology to show input and output from computational core in graphical form. PHP technology creates bridge between user interface and C++ software with combination of AJAX technology to refresh only part of web page. JavaScript is used for simple tasks on client side (web-browser).

Fig. 5 shows implemented software in block diagram how virtual laboratory is connected to group of embedded systems and how HTML, PHP, AJAX and JavaScript programming languages are used.

Virtual laboratories need access to all systems from one network. We implemented open source VPN network openVPN server into HP Proliant server and openVPN clients into connected embedded systems. This private network is encrypted by 256bit private key and any client must connect by its own code, which identifies every active user on this network. The web user interface monitors any client with data bandwidth on server and device.

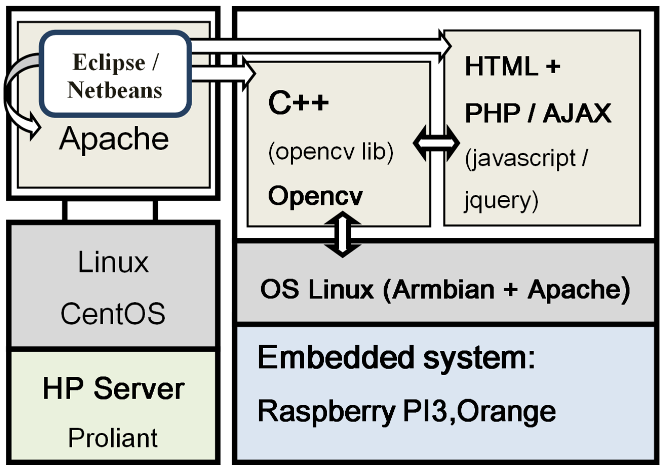


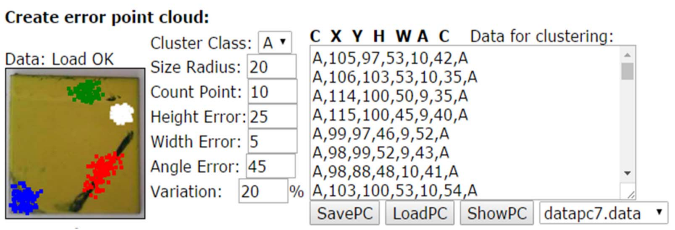
Fig. 5. Block diagram of virtual laboratory.

Fig. 6 depicts manual and automatized data import of 2D data to clustering and classification process. The application can integrate 5 dimensions of input data. The data are marked by alphabetical symbols x, y, w, h, a. Position in 2D space [x, y], width and height [w, h] and angle of element [a] for example in grades. Test data can be written to textbox or create data manually with mouse click in JavaScript Canvas component which convert to text. There is possibility to load any picture as background for data selection.

We can use these inputs for any data integer or float. If we don't need all dimensions we can fill unused data by zero value or let it blank.

Comparison of clustering methods was implemented as simple conditional system which creates clusters with condition of percentage size of every parameter to clusters (Fig. 7 right).

Fig. 7 shows selection of advanced clustering algorithms and simple conditional system in web application with some basic setup parameters to tune algorithms. Fig. 8 shows result example of clustering for specific algorithm.



Create error point cloud:

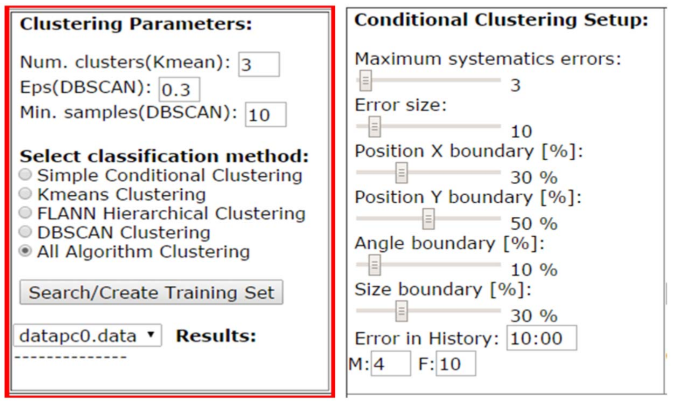
Data: Load OK

Cluster Class: A
Size Radius: 20
Count Point: 10
Height Error: 25
Width Error: 5
Angle Error: 45
Variation: 20 %

C X Y H W A C Data for clustering:

```
A,105,97,53,10,42,A
A,106,103,53,10,35,A
A,114,100,50,9,35,A
A,115,100,45,9,40,A
A,99,97,46,9,52,A
A,98,99,52,9,43,A
A,98,88,48,10,41,A
A,103,100,53,10,54,A
```

Fig. 6. Data import UI example for clustering and classification.



Clustering Parameters:

Num. clusters(Kmean): 3
Eps(DBSCAN): 0.3
Min. samples(DBSCAN): 10

Select classification method:
- Simple Conditional Clustering
- Kmeans Clustering
- FLANN Hierarchical Clustering
- DBSCAN Clustering
- [✓ All Algorithm Clustering]

Results:

Conditional Clustering Setup:

Maximum systematics errors: 3
Error size: 10
Position X boundary [%]: 30 %
Position Y boundary [%]: 50 %
Angle boundary [%]: 10 %
Size boundary [%]: 30 %
Error in History: 10:00
M: 4 F: 10

Fig. 7. Web UI for clustering of data.

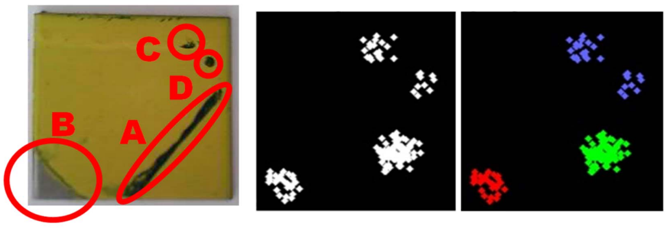


Fig. 8. Graphical output for evalution of clustering reliability.

The first picture in Fig. 8 depicts example of input image with some errors which creates source of data for clustering.

The second picture in Fig. 8 shows input data without clustering, it contains only data which are all in one cluster.

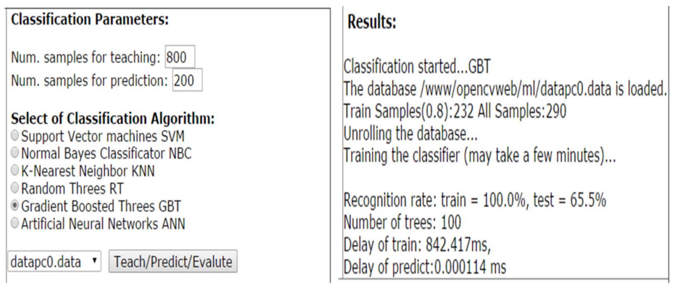
The last picture in Fig. 8 depicts clustered data to some groups (clusters) by color changing of data according their reference to cluster class. The image shows graphical result for evaluation of reliability of algorithms in 2D space. We can also implement next dimension for example angle in 3D space. The main disadvantage of graphical evaluation of algorithm suitability is no possibility to show 4 or more dimensional cluster in some graphical representation.

All implemented algorithms are suitable for simple data distribution. Specific distribution of data generates wrong output for basic algorithms (hierarchical, K-mean). The most suitable algorithm was DBSCAN, because it clears some noise from process of clustering.

The next step is process of learning model used later for prediction of reference new data into existing cluster. It is possible to select and test the 6 basic classification algorithms. The main parameters for evaluation of learning reliability and prediction are percentage and time delay for these two steps.

The basic set of data for teaching and prediction is divided in some scale. The result is shown from GBT (Gradient Boosted Tress). The algorithm's suitability depends mainly on the speed (delay) of prediction for new data.

Fig. 9 shows UI example for classification algorithm, Fig. 10 for machine learning.


Classification Parameters:

Num. samples for teaching: 800
Num. samples for prediction: 200

Select of Classification Algorithm:
- Support Vector machines SVM
- Normal Bayes Classificator NBC
- K-Nearest Neighbor KNN
- Random Threes RT
- [✓ Gradient Boosted Threes GBT]
- Artificial Neural Networks ANN

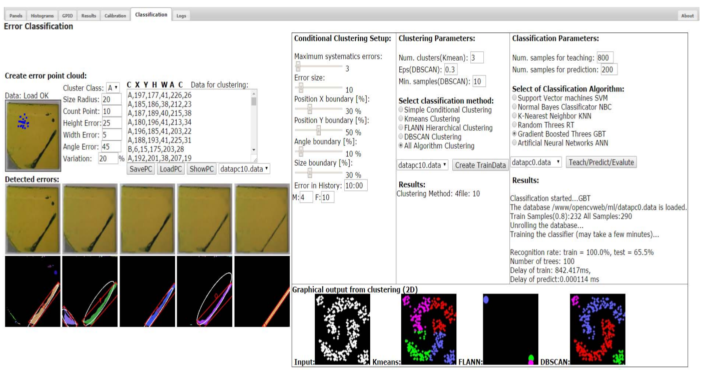
datapc0.data

Results:

```
Classification started...GBT
The database /www/opencvweb/ml/datapc0.data is loaded.
Train Samples(0.8):232 All Samples:290
Unrolling the database...
Training the classifier (may take a few minutes)...

Recognition rate: train = 100.0%, test = 65.5%
Number of trees: 100
Delay of train: 842.417ms,
Delay of predict:0.000114 ms
```

Fig. 9. User interface for classifaction algorithms.


Error Classification

Create error point cloud:

Data: Load OK
Cluster Class: A
Size Radius: 20
Count Point: 10
Height Error: 25
Width Error: 5
Angle Error: 45
Variation: 20 %

C X Y H W A C Data for clustering:

```
A,197,177,41,226,26
A,185,186,38,212,23
A,187,189,40,215,38
A,180,196,41,213,34
A,196,185,41,203,22
A,188,193,41,225,31
B,6,15,175,203,28
A,192,201,38,207,19
```

datapc10.data

Detected errors:

Conditional Clustering Setup:

Maximum systematics errors: 3
Error size: 10
Position X boundary [%]: 30 %
Position Y boundary [%]: 50 %
Angle boundary [%]: 10 %
Size boundary [%]: 30 %
Error in History: 10:00
M: 4 F: 10

Clustering Parameters:

Num. clusters(Kmean): 3
Eps(DBSCAN): 0.3
Min. samples(DBSCAN): 10

Select classification method:
- Simple Conditional Clustering
- Kmeans Clustering
- FLANN Hierarchical Clustering
- DBSCAN Clustering
- [✓ All Algorithm Clustering]

datapc10.data

Results:
Clustering Method: 4file: 10

Classification Parameters:

Num. samples for teaching: 800
Num. samples for prediction: 200

Select of Classification Algorithm:
- Support Vector machines SVM
- Normal Bayes Classificator NBC
- K-Nearest Neighbor KNN
- Random Threes RT
- [✓ Gradient Boosted Threes GBT]
- Artificial Neural Networks ANN

datapc0.data

Results:

```
Classification started...GBT
The database /www/opencvweb/ml/datapc0.data is loaded.
Train Samples(0.8):232 All Samples:290
Unrolling the database...
Training the classifier (may take a few minutes)...

Recognition rate: train = 100.0%, test = 65.5%
Number of trees: 100
Delay of train: 842.417ms,
Delay of predict:0.000114 ms
```

Fig. 10. Web user interface for machine learning.

V. Experiments

We used two embedded platforms for algorithm testing: Raspberry PI3 and Orange PI Lite. Speed of the algorithms running on ARM architecture (ARMv7/ARMv8) was compared with standard x86 server architecture (Intel Xeon processor). Tested platform had to contain unit for floating operation. Intel uses SSE instruction and embedded system NEON with advanced SIMD.

TABLE I. Parameters of tested systems

Desktop / Embeded system	Parameters		
	Core/bits	*CPU*	*Memory*
x86 HP Proliant	Xeon/64	1.8 GHz	16 GB
ARMv7 Orange PI Lite	Cortex A7/32	1.2 GHz	0,512 GB
ARMv8 Raspberry PI3	Cortex A53/64	1.2 GHz	1 GB

The basic set of data with 400 samples was used and it was divided into 4 clusters. Training set was 320 samples and testing set was 80 samples. DBSCAN algorithm doesn't use all data from dataset to create cluster, only according to position parameters. The ANN algorithm uses all values for learning. The used neural network had two layers with 100 hidden neurons and it acquired in learning and testing 100 percent reliability.

In Tab. II there are measured delays of machine learning algorithms in embedded systems with comparison to standard x86 processor.

TABLE II. Comparison of devices delay

Desktop / Embeded system	Delay for algorithms		
	Clustering DBSCAN [ms]	*Teaching ANN [ms]*	*Classification ANN [ms]*
x86 HP Proliant	13.20	7 386.4	8.85
ARMv7 Orange PI Lite	228.20	57 072.4	68.12
ARMv8 Raspberry PI3	82.09	38 445.3	39.73

VI. Conclusion

In the paper there are presented some results of experiments with implementation of computational intensive algorithm of clustering and machine learning into embedded system. The main idea was combination of C++ backend for computation and web application user interface for control and representation of outputs. The most computational intensive algorithms for clustering – DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and for classification – ANN (Artificial Neural Networks) were selected for delay testing. The result of experiments is that currently available embedded system can provide enough of computational resources to create functional system for teaching and classification of data from clouds and they can be suitable as virtual laboratories for testing of sample data online.

The results show that 64 bit ARM computes 4-6 times slower than x86 Xeon server CPU. The performance can increase with new ARM Cortex processor A72 boards with octa or deca cores.

It can be seen from experiments carried out with a selected combination of algorithm for clustering and classification that scientific progress will be directed to embedded systems which provide sufficient computational performance for easier data mining tasks.

The next development will be to test NVIDIA Jetson TX2 which can do some speedup for these algorithms by usability of integrated GPU for parallel computing by shader units with CUDA technology. Our new research and experiments will be targeted to deep learning algorithms from OPENCV module DNN and VISIONWORKS SDK [21], [22].

Acknowledgment

This work was supported by the Agency for Research and Development under the contract no. APVV-15-0602 and also by the Project of the Structural Funds of the EU, ITMS code: 26220220030.

References

[1] D. Parsons, *Dynamic Web Application Development Using XML and JAVA*. London: Cengage Learning EMEA, 2008, 629 p.

[2] S. Stobart and D. Parsons, *Dynamic Web Application Development Using PHP and MySQL*. London: Cengage Learning EMEA, 2008, 633 p.

[3] S. D. Jang, "Performance Analyses of Web Page Model Using Ajax Technologies," *Advanced Science Letters,* vol. 22, no. 11, 2016, pp. 3437-3440

[4] J. Boržíková, J. Mižák, nad J. Piteľ, „Monitoring of operating conditions of biomass combustion process,“ *Applied Mechanics and Materials*, vol. 308, 2013, pp. 153-158

[5] J. Mižáková, "Monitoring of biomass-based heat production system," *Applied Mechanics and Materials*, vol. 616, 2014, pp. 143-150

[6] I. Čorný, „Overview of progressive evaluation methods for monitoring of heat production and distribution,“ *Procedia Engineering*, vol. 190, 2017, pp. 619-626

[7] I. Čorný, „Possibilities of application of computational intelligence in monitoring of heat production and supply," *Key Engineering Materials*, vol. 669, 2016, pp. 560-567

[8] S. Vrána, B. Šulc, P. Trnka, and M. Kuře, „Experience from various technological concepts applied in virtual control laboratory". in *3rd IFAC Workshop on Internet Based Control Education (IBCE 2015),* IFAC PAPERSONLINE, vol. 48, issue 29, 2015. pp. 277-282

[9] T. Saloky and J. Šeminský, "Project of laboratory of systems with artificial intelligence." *TRANSACTIONS of the VŠB – Technical University of Ostrava*, LI, nr. 2, 2005, pp. 117-120

[10] I. Bukovský, P. Beneš, M. Veselý, J. Kalivoda, J. Piteľ, O. Líška, K. Ichiji, and N. Hamma, "Poznatky z výskumu neuro-regulátorů a z laboratorní praxe." (in Czech), in *Automatizace, regulace a procesy* (*ARaP 2016*), Prague: MM Publishing, 2016, pp. 105-111

[11] G.V. Oliveira, F.P. Coutinho, R. Campello, and M.C. aldi, "Improving k-means through distributed scalable metaheuristics." *Neurocomputing,* vol. 246, 2017, pp. 45-57.

[12] J. Sahoo, L. Mishra, and M.N. Mohanty, "GA Based Optimization of sign regressor FLANN model for channel equalization." in *Recent Developments in Intelligent Systems and Interactive Applications (Iisa2016),* Proceedings Paper vol. 541, 2017, pp. 124-130

[13] J. Hou, H.J. Gao, and X.L. Li, "DSets-DBSCAN: A parameter-free clustering algorithm." *IEEE Transactions on Image Processing,* vol. 25, no. 7, 2016, 12 p.

[14] C.C. Chang and C.J. Lin. "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011, 27:1–27:27

[15] R. Burduk, "On the bounds on optimal bayes error in the task of multiple data sources," in *Image Processing and Communications Challenges 4*, Berlin: Springer-Verlag, 2013, pp. 201-208

[16] S.C. Zhang, X.L. Li, M. Zong, X.F. Zhu, and D.B. Cheng, "Learning k for kNN classification," *ACM Transactions on Intelligent Systems and Technology,* vol. 8, no. 3, 2017, 19 p.

[17] J.H. Friedman, Greedy function aproximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, 2001, pp. 1189-1232

[18] S. Fletcher and M.Z. Islam, "Differentially private random decision forests using smooth sensitivity," *Expert Systems with Applications,* vol. 78, 2017, pp. 16-31

[19] Y. LeCun, L. Bottou, G.B. Orr, and K.R. Muller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, vol. 1524, 2002, pp. 5-50

[20] K. Židek and E. Rigasová, "Diagnostics of products by vision system", *Applied Mechanics and Materials*. vol. 308, 2013, pp. 33-38

[21] T. Gong, T. Fan, J. Guo, and Z. Cai, "GPU-based parallel optimization of immune convolutional neural network and embedded system," *Engineering Applications of Artificial Intelligence*, vol. 62, 2017, pp. 384-395

[22] A. Lovyannikov and S. Zapechnikov, "Machine learning for embedded devices software analysis via hardware platform emulation", in *2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, St. Petersburg: IEEE, 2017, pp. 489-492