AI Bricks: A Microservices-based Software for a Usage in the Cloud Robotics

1st Patrik Sabol
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Kosice, Slovakia
patrik.sabol@tuke.sk

2nd Peter Sincak
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Kosice, Slovakia
peter.sincak@tuke.sk

***Abstract*—In this paper, the structure of cloud-based modular software that employs artificial intelligence algorithms, called as AI Brick, is proposed. Two AI Bricks were implemented, Kohonen Brick implementing Kohonen Self-organized map neural network and MLP Brick implementing Multi-Layered Perceptron. The proposed structure of AI Brick is based on two main ideas of cloud robotics: sharing knowledge and offloading computationally intensive tasks from robot's onboard execution to the cloud environment. As a testbed, we implemented NARA model (Neural network design on Approximate Reasoning Architecture) as a cloud service, which combines the functionality of implemented AI Bricks to solve the most advanced task. We present the way of arbitrary AI bricks interconnection, which allows us to employ a distinct algorithm to simulate or control different systems.**

***Index Terms*—AI Bricks, Cloud Robotics, Cloud Computing, Kohonen, MLP, Microservices**

I. Introduction

The term "*Cloud Robotics*" was first introduced in 2010 by James Kuffner. He used the term to describe robotic system based on cloud computing. This term is broadly defined in [1] as follows "*Any robot or automation system that relies on either data or code from a network to support its operation, i.e., where not all sensing, computation, and memory is integrated into a single standalone system.*"

This definition reveals many benefits of the cloud robotics. One of the advantages is the capability of migrating computationally exhaustive tasks from robots onboard execution to the cloud environment. Therefore, robots do not require such powerful hardware configuration, which reflects on the lighter and less costly robots construction with longer battery life [2]. Moreover, the cloud can host a database or library, which provides a medium for the robots to share information and learn new skills and knowledge form each other. Furthermore, besides the robots sensing system it is also possible to use sensing systems out of the robots body; the robot may receive information about the environment from additional sources, such as cameras in the room. Robots equipped with described feature are known as ambient robots [3].

RoboEarth is a cloud robotic project, whose objective is to develop the Internet of Robots. Its infrastructure consists of World-Wide-Web style database, which stores knowledge generated by humans - and robots - in a machine-readable format and allows robots to store and share information and also to collaborate and achieve common tasks [4]. The RoboEarth Cloud Engine Rapyuta [5], an open source Platform-as-a-Service (PaaS) framework, which follows modular design, allows robots to offload their heavy computation to secure computing environments in the cloud. The computing environments also allow robots to easily access the RoboEarth knowledge repository. It is set up to run any process that is a ROS (robotic middleware) node, which is considered as a software module. Rapyuta's computing environment is implemented using Linux Containers [6] that run on the cloud infrastructure.

In [7], a new concept, AI (Artificial Intelligence) Brick, was proposed in cloud robotic, which provides a way to build specialized tools responsible for solving specific tasks using artificial intelligence. These tools can be easily expanded among developers as a cloud-based software library that can be attached to their new cloud service. It follows microservice architecture pattern. [8]

In this paper, the structure of the AI Brick is proposed and implemented as the modular software for neuro-fuzzy model NARA (Neural Network design on Approximate Reasoning Architecture), which consists of two cooperating AI Bricks together; Kohonen Brick implementing Kohonen neural network [9] and MLP Brick implementing Multi-Layered Perceptron.

The content of the paper is organized as follows: the concept of the AI Brick is described in Section 2. The implementation of two AI Bricks is presented in Section 3. The proposal and implementation of the NARA model as a clouod-based modular software is located in Section 4. Experiments are presented in Secion 5. Finally, conclusion and future work are located in Section 6.

II. AI Brick concept

In [7], Ferrate suggests creating the new high-level layer application for robots and all kind of machines with many specialized tools hosted in the cloud, called as "AI Brick". For communication, he proposed to use standard protocols, which are independent of technology and operation system.

"High-level layer application" is considered as modular software that consists of cooperating AI bricks as modules, which together forms more complex cloud services that are used by agents (see Fig. 1).

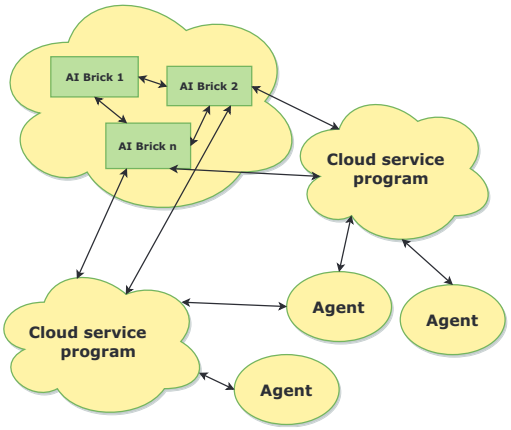

Fig. 1. Example of communication between agents and cloud services, between cloud service and AI bricks and between AI bricks

In other words, AI Brick is considered as a software module (library) deployed in the cloud environment. In [10], requirements for implementation of AI Bricks was proposed as follows:

1) Each AI brick has to be implemented as a software library attached to a cloud service.
2) Each AI brick has to solve its task completely. Task can solve itself or can collaborate with other AI bricks.
3) Each AI brick should have the capability to communicate with other AI bricks. It means that AI bricks should be able to receive inputs from other AI bricks and sent them outputs.
4) Each AI brick must have exactly defined desired inputs and outputs.
5) Each AI brick must be scalable. It means that program should be implemented in the way to run multiple instances on different virtual machines or use multiple threads running on various CPU cores.
6) Data stored in cloud storage must be synchronized between instances of the same AI brick deployed on different virtual servers.

III. Proposed AI Brick's structure

A. Communication with AI Brick

The main idea of AI Brick is to encapsulate AI algorithm using microservice architecture pattern. Hence, it should communicate with the client only via inputs and outputs of the AI Brick (as seen in Fig. 2)

As mentioned in the Introduction section, one of the benefits of cloud robotics is the ability to store and share knowledge among all robotic systems. For this purpose, Central Knowledge Base (CKB) was added between AI brick and client (as shown in Fig. 3). Therefore, communication flow can be described as follows: The client sends the input to the AI Brick, which processes the input and output stores into CKB. Afterwards, the client is able to read and share knowledge from CKB.

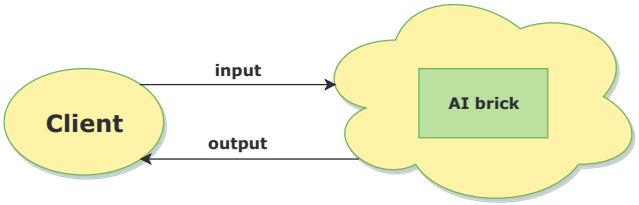

Fig. 2. Communication between the AI brick and the client

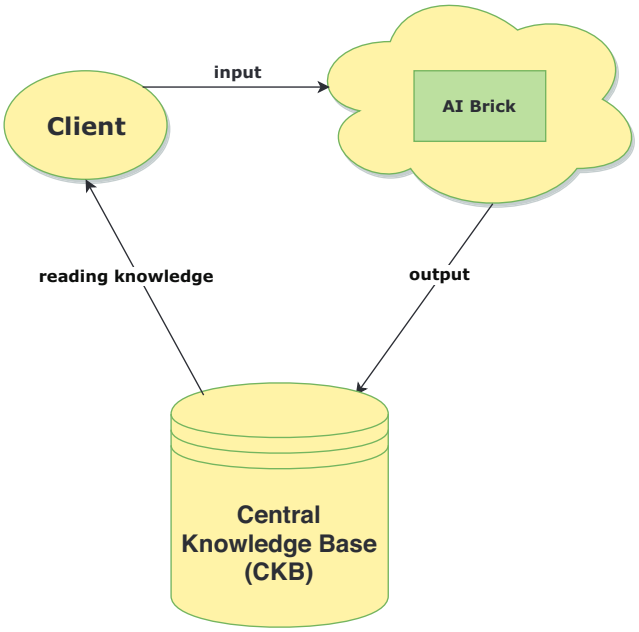

Fig. 3. Centralized knowledge storing approach

B. Internal structure of AI Brick

The proposed internal structure of AI brick (see Fig. 4) consists of the communication interface, internal storage, and block of AI algorithm. The role of communication interface is to communicate with the client, to receive, preprocess and pass input data and parameters through internal storage to the block of AI algorithm, which process the data and store knowledge into CKB.

IV. Implementation of AI Bricks

A. Used technologies for a development of AI Bricks

Proposed AI Bricks are built on Microsoft Azure platform, which offer a wide range of products and technologies; therefore it is suitable for developing and testing of any kind of services.

There are many communication technologies, which can be used as a communication interface, such as Simple Object Access Protocol (SOAP), REST API, Windows Communication Foundation (WCF), Service Bus or cloud data storage. In [10], Bus technology was proposed to use as a communication

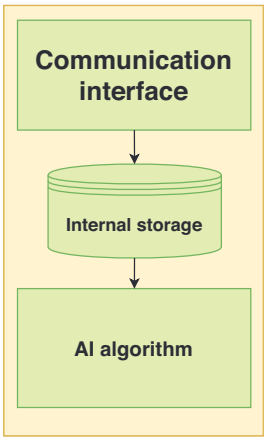

Fig. 4. Internal structure of AI Brick

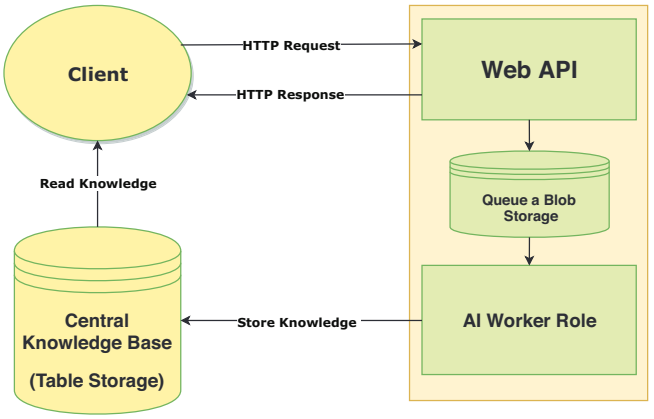

Fig. 5. Structure of communication between AI brick, client and Central Base of Knowledge

interface. In this paper, we propose to use ASP.NET Web API, a framework that is used for building a RESTful application on the .NET Framework. This framework uses request-response message system and makes it easy to create HTTP services that reach a broad range of clients, including browsers and mobile devices and all kind of robotic systems, which is suitable for cloud robotics.

The client sends input encapsulated in JSON object to AI Brick via HTTP requests. By return, via HTTP response, the client receives globally unique identifier (GUID) that is used to uniquely identify a knowledge in the CKB.

The Central Knowledge Base is implemented as an Azure Table Storage that is used to store structured NoSQL data without relations. Microsoft Azure also provides Blob Storage to store unstructured data and Queue Storage for messaging queues. AI algorithm is implemented using Azure Worker Role, which is generally used to perform intensive background processes. Communication bridge between Web API and AI Worker Role is represented by a combination of Queue and Blob storage. Input data from the client is stored in the internal Blob storage as a JSON object, whose name is sent to AI Worker Role as a message through Queue storage. This solution is used because Queue storage supports messages of the maximum size of 64 KB. Whole communication flow is illustrated in Fig. 5

B. Kohonen Self-organized Map (SOM) Brick

Self-organized map (SOM), also known as Kohonen map is an artificial neural network that is trained using unsupervised learning to produce a low-dimensional, discretized representation of the input space of the training samples. It is usually used for clustering [9].

The input JSON object for Kohonen Brick should contain following atributes:

- size of Kohonen's map
- learning rate γ
- height of the Mexican hat wavelet neighbourhood function

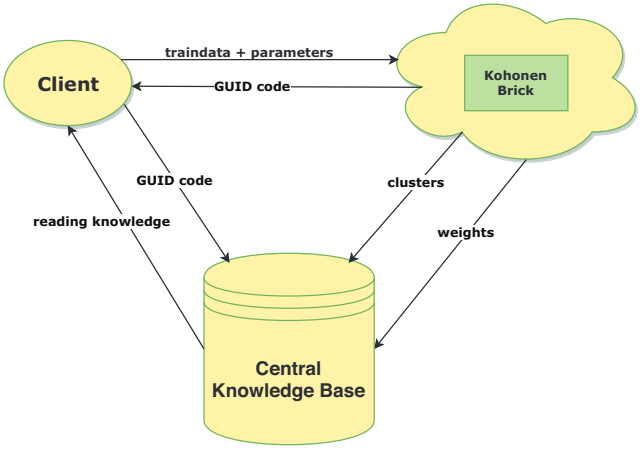

Fig. 6. Visualization of the Kohonen Brick

- radius of the Mexican hat wavelet neighbourhood function
- number of learning iteration
- training data set

Training data set is represented as a two-dimensional jagged array of doubles, where the size of the first dimension is a number of training samples and size of the second dimension is a number of inputs for Kohonen neural network.

Outputs from the Kohonen Brick are indexes of clusters for each user's data samples and Kohonen neural network's weights. (see Fig. 6)

C. Multi-Layered Perceptron Brick

Multi-Layered Perceptron (MLP) is a feed-forward neural network with one or more layers between input and output layer. Proposed MLP Brick uses Backpropagation learning algorithm for training MLP.

The input JSON object for MLP Brick should contain following atributes:

- number of input neurons
- number of output neurons

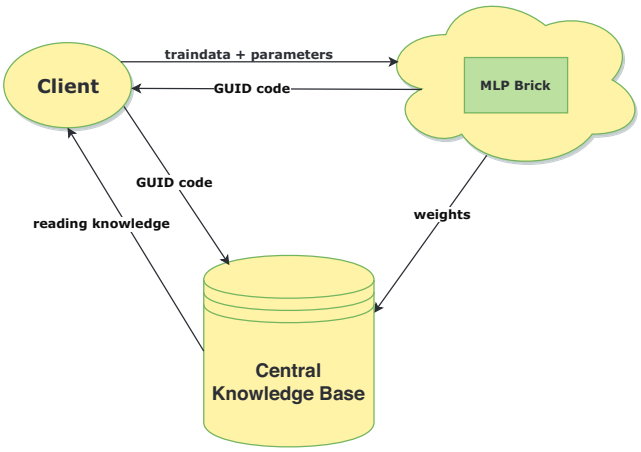

Fig. 7. Visualization of the MLP Brick

- number of neurons in first hidden layer
- number of neurons in second hidden layer
- learning rate γ
- steepness of sigmoid function
- number of learning iteration
- training data set

Training data set is also represented as a two-dimensional jagged array of doubles, where the size of the first dimension is a number of training samples. The second dimension consists of inputs and desired outputs. Therefore, the size of the second dimension should be equal to the sum of input and output neurons count.

Output from the MLP Brick are weights of the feed-forward neural network (see Fig. 7).

V. Neuro-fuzzy model NARA

The NARA (Neural network design on Approximate Reasoning Architecture) architecture, proposed in [11], belongs to a class of hybrid neuro-fuzzy systems, which combines artificial neural network and fuzzy logic. The main strength of NARA system is that it is an universal approximator with the ability to solicit interpretable IF-THEN rules.

In [12], the NARA architecture was used for development of a tool for a water management system to simulate conditions, which are able to prevent legionelosis outbreaks in a drinking water system.

A. Structure of the NARA model

The NARA model is organized and constructed by the structure of IF-THEN fuzzy rules. It consists of two types of neural networks (NN) (sub-networks). First NN partitions the input space and identifies the clusters. This input space does not have to be divided crisply. It is similar to the membership function of the fuzzy set. Second NN consists of many parallel NNs which are experts for identified clusters in the input space.

In the proposed implementation of the NARA model, Kohonen NN is used as a memory type NN (NSmem). This network corresponds to the IF part (antecedents) of fuzzy inference rules. For parallel sub-networks $(NS_1, ..., NS_n)$ are used MLPs that correspond to the THEN part (consequent) of the n-th fuzzy inference rule (see Fig. 8).

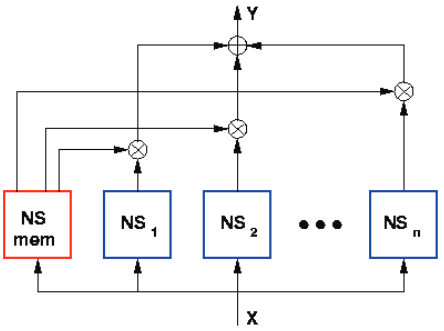

Fig. 8. The conceptual scheme of a NARA neural network with neural sub-networks

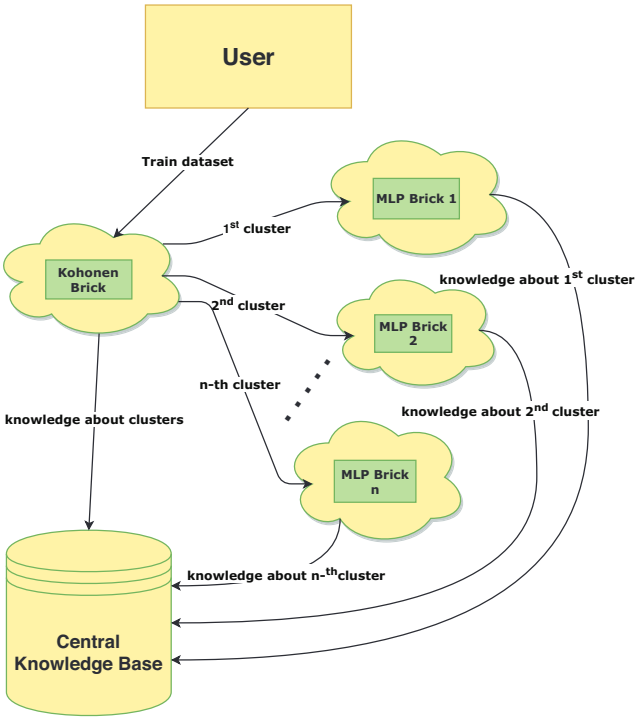

Fig. 9. Visualization of the NARA cloud service

B. NARA model as a modular software on the cloud

In this paper, the NARA model is implemented as a modular software deployed in the cloud environment, which consists of AI Bricks as modules, more concretely Kohonen Brick and MLP Brick. Together, they form modular cloud service NARA.

In Fig. 9, logic flow of the NARA cloud service is visualized. First, user uploads training data set that are sent to the Kohonen Brick, which divides training data to clusters. Afterwards, for each cluster, new instances of the MLP Brick are initialized.

The NARA cloud service is implemented as a Azure Web Role service using ASP.NET MVC 5. User's data are stored

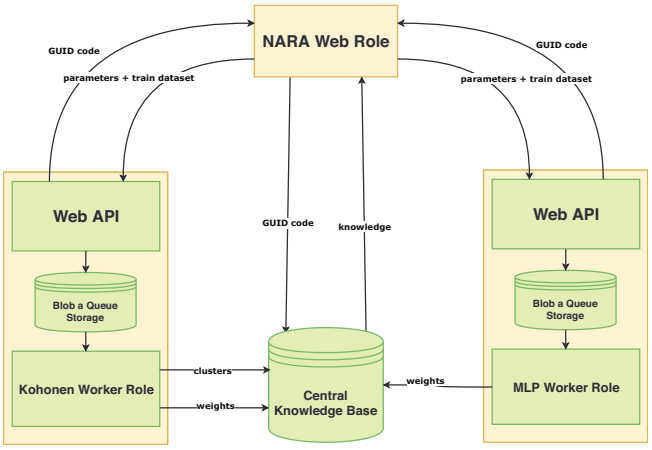

Fig. 10. Implementation of the NARA system as a set of modules on cloud

in Azure Storage. ASP.NET Identity implements user authentication using Azure SQL Database.

Fig. 10 shows a implementation diagram of the cloud-based NARA model. It shows a communication between the NARA Web Role, Kohonen and MLP Bricks and the Central Knowledge Base.

VI. Experiments

The focus of experiments is to test the communication between client and Web API. Moreover, cloud environment is able to provide on-demand powerful computing resources. Therefore, learning times of Kohonen NN and MLP are measured on local and cloud environment. Finally, to test performance of the implemented NARA model, one of the benchmark data set was used.

A. Communication between client and Web API

To test the communication between client and Web API, elapsed time between HTTP request and HTTP response was measured, which depends on the size of a sent JSON object and on client's internet connection upload speed. HTTP request was sent from two environments, local computer and cloud. Tests were performed by sending one requests and three asynchronous requests at once.

The size of the sent JSON object was 2.7 megabytes. For local environment, ten measurements was executed with different upload speeds. For the cloud environment, it was not possible to measure upload speed, therefore, in figures, response time is represented as a line, not as a point. AI Bricks was deployed in the same data center as a client cloud service.

Figures 11 and 12 show a strong dependency between response time and client's upload speed of internet connection. The comparison between the cloud and local environment confirmed the hypothesis that time is the lowest in a cloud environment, because cloud data centers are covered with the best internet connection. Results from those experiments show, how important is the quality of internet connection in practical usage of the cloud robotic.

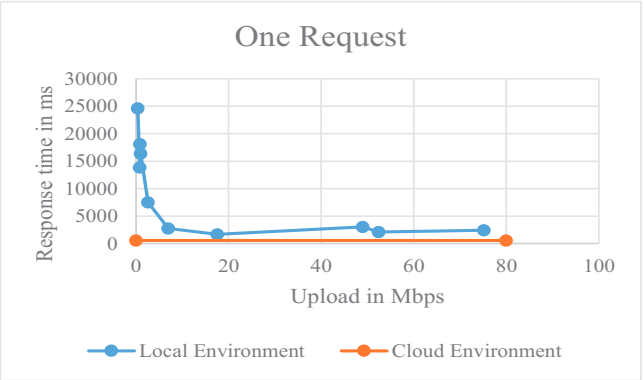

Fig. 11. Graph of the response time between one clients request and WebAPI. The y-axis shows response time in milliseconds and x-axis shows client's upload speed of the internet connection in megabits per seconds

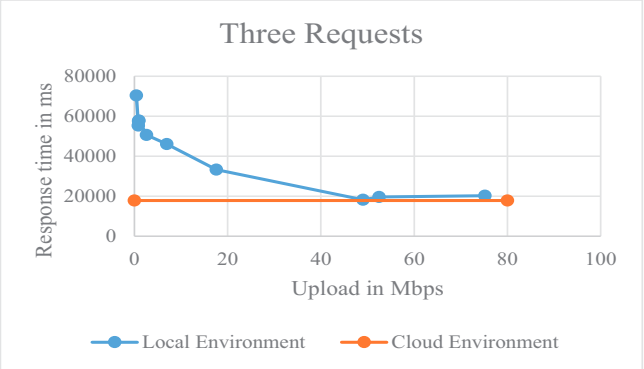

Fig. 12. Graph of response time between three clients requests and WebAPI. The y-axis shows response time in milliseconds and x-axis shows client's upload speed of the internet connection in megabits per seconds

B. Learning time of Kohonen NN and MLP

Learning time of Kohonen NN and MLP was measured in two environment, local PC and cloud. The local environment was featured by CPU *Intel Core i5-5200U @ 2.20GHz* and the cloud with *Intel(R) Xeon(R) CPU E5-2660 @ 2.20.GHz*.

Tables I and II clearly show that cloud environment, equipped with more powerful CPU than the ordinary laptop, is able to run learning of NNs much more faster.

C. Performance of the proposed cloud-based NARA model

For the performance experiment, well-known IRIS data set from the UCI machine learning repository was used. The average accuracy from runs of fivefold cross-validation was $98\% \pm 1.24\%$, which clearly proves the correctness of the NARA model implementation.

TABLE I
Learning time of the Kohonen NN

Environment	Training samples	Elapsed time
Cloud	9900	11 2013 ms
Local	9900	25 990 ms

TABLE II
LEARNING TIME OF THE MLP

Environment	Training samples	Elapsed time
Cloud	3856	25 331 ms
Local	3856	103 441 ms
Cloud	1925	12 930 ms
Local	1925	62 187 ms

VII. CONCLUSION

In this paper, we presented the proposal and the implementation of the structure of cloud-based modular software that employs artificial intelligence algorithms, called as AI Brick. The proposed structure of AI Brick is based on two main ideas of cloud robotics: sharing knowledge and offloading computationally intensive tasks from robot's onboard execution to the cloud environment. We have implemented two AI Bricks employing two common learning algorithms for artificial intelligence, learning algorithm for Kohonen Self-organized Map neural network and Backpropagation learning algorithm for Multi-Layer Perceptron. The main benefit of our implementation of AI brick is employing Web API as a communication interface, which allows being query by simple HTTP methods. Moreover, proposed architecture allows a simple knowledge sharing just by sharing GUID code used to uniquely identify a knowledge in the Central Knowledge Base.

As a testbed, we implemented NARA model (Neural network design on Approximate Reasoning Architecture) as a cloud service, which combines the functionality of implemented AI Bricks to solve the most advanced task.

In experiments, we have demonstrated the functionality of communication between client and Web API in AI Brick. Moreover, we have proved some advantages of cloud robotics. We have demonstrated that compute-intensive tasks, such as learning MLP using backpropagation method, were solved faster in the cloud environment.

For the future work, AI Brick should be implemented using more scalable technology, for example Azure Container Service.

ACKNOWLEDGMENT

This research work was supported by the Slovak Research and Development Agency under the contract No. APVV-015-0731, supported from 07-2016 to 06-2020.

REFERENCES

[1] James Kuffner. Cloud-enabled humanoid robots. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on, Nashville TN, United States, Dec.*, 2010.

[2] Guoqiang Hu, Wee Peng Tay, and Yonggang Wen. Cloud robotics: architecture, challenges and applications. *IEEE network*, 26(3), 2012.

[3] Kyuseo Han, Jaeyoung Lee, Sangik Na, and Wonpil You. An ambient robot system based on sensor network: Concept and contents of ubiquitous robotic space. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2007. UBICOMM'07. International Conference on*, pages 155–159. IEEE, 2007.

[4] Markus Waibel, Michael Beetz, Javier Civera, Raffaello d'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Häussermann, Rob Janssen, JMM Montiel, Alexander Perzylo, et al. Roboearth. *IEEE Robotics & Automation Magazine*, 18(2):69–82, 2011.

[5] Dominique Hunziker, Gajamohan Mohanarajah, Markus Waibel, and Raffaello D'Andrea. Rapyuta: The RoboEarth cloud engine. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 438–444, Karlsruhe, Germany, 2013.

[6] Infrastructure for container projects. https://linuxcontainers.org/, 2018. Accessed: 2018-04-20.

[7] Cloud roboting new paradigm is near [online]. http://www.robotica-personal.es/2013/01/cloud-robotics-new-paradigm-is-near.html, 2013. Accessed: 2018-04-20.

[8] Mario Villamizar, Oscar Garcés, Harold Castro, Mauricio Verano, Lorena Salamanca, Rubby Casallas, and Santiago Gil. Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. In *Computing Colombian Conference (10CCC), 2015 10th*, pages 583–590. IEEE, 2015.

[9] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1-3):1–6, 1998.

[10] How to build ai bricks? [online]. http://ouraicit.tumblr.com/post/115761079238/how-to-build-ai-bricks, 2015. Accessed: 2018-04-20.

[11] Hideyuki Takagi, Noriyuki Suzuki, Toshiyuki Koda, and Yoshihiro Kojima. Neural networks designed on approximate reasoning architecture and their applications. *IEEE Transactions on Neural Networks*, 3(5):752–760, 1992.

[12] Peter Sinčak, Jaroslav Ondo, Daniela Kaposztasova, Maria Virčikova, Zuzana Vranayova, and Jakub Sabol. Artificial intelligence in public health prevention of legionelosis in drinking water systems. *International journal of environmental research and public health*, 11(8):8597–8611, 2014.