Robotized application based on deep learning and Internet of Things

Carlos Pascal, Laura-Ofelia Raveica, and Doru Panescu
Department of Automatic Control and Applied Informatics
"Gheorghe Asachi" Technical University of Iași
Iași, Romania
{cpascal, raveica.laura, dorup}@ac.tuiasi.ro

***Abstract*— This paper presents a way to integrate an industrial robot into Internet of Things and to use it with a deep learning application. Besides of manufacturer's restrictions, which usually exist in an industrial scenario, an easy method to extend and merge the sensorial and decisional systems for robots will be required as part of Industry 4.0. Related to this, the proposed method couples IBM Watson IoT cloud-based platform, a Node-RED cloud application, a deep learning mechanism with TensorFlow (this being applied for a computer vision case study), and an old generation industrial robot. Several conclusions highlight the trade-off of using IoT and deep learning solutions for a real manufacturing environment.**

***Keywords*— robotics, Internet of Things, deep learning, computer vision, cloud infrastructure, neural networks**

I. INTRODUCTION

Nowadays industrial applications are in continuous renewal and between the technologies being involved one can frequently note Internet of Things (IoT) [1], [2], deep learning [3], cloud computing [4], computer vision systems, and sensor networks. A common use of deep learning for solving industrial tasks is to couple this method with computer vision techniques [3]. This is the approach considered in this paper, too. Though IoT is in its beginning stage of industrial adoption, it is already valuable for testing and validation of new functional modules. This is the case treated in this research, which had two main goals: to develop and test a deep learning module being linked to a computer vision system, and to establish an interface for an industrial robot based on IoT that should allow the connection to cloud infrastructure.

There are many researches in the IoT field. It is to notice that the application of this paradigm in an industrial environment determines many specific issues, so that now the distinction is made through the notion of Industrial Internet of Things (IIoT) [2], [5]. The overview presented in [2] highlights the problems regarding connectivity in IIOT and also the challenges for this solution. Besides the benefits obtained with this new technology, the problems regarding the security requirements must be discussed, too. In [5], the underlined advantages of IIOT regard the way precision machining can be achieved by the use of embedded sensors and intelligent algorithms that can detect, in real time, deviations from nominal values during the manufacturing processes. A new paradigm is applied in [6], namely the so-called edge computing, which means that computation is done as much as possible close to the sensorial part and only preprocessed information is transferred towards cloud services. In this way, the obtained advantage is about reducing communication load. This is a significant issue, due to the great volume of data determined by sensors, resources and products with embedded intelligence.

Starting from the common case of robotic systems, which operate in repetitive cycles with reduced adaptability (as it is the case of the robotic environment we used in our previous experiments [7]), the purpose was to extend the sensorial and decisional subsystems. Having this in mind, our objective was to get an entire system of the type "see, decide and act", which should be similar to the principles of agent-based systems. In order to validate the proposed approach and understand its specific issues, the problem of recognizing digits from zero to nine in a real environment was adopted; it has the advantage that some results regarding it exist in the literature [8]. Image processing and the trained neuronal network have great importance in this case, as well as the light, shadows and camera specifications. The main issues introduced through this paper refer to the connection of an industrial robot with a deep learning module using an IoT cloud platform, the comparison of learning performances involved in the object recognition scenario, and utilization of some image processing strategies based on OpenCV [9] to improve the recognized rate.

One way to apply deep learning is by means of TensorFlow; this is an open source software library used for numerical computation [10]. It was developed by engineers at Google Brain to explore deep neural networks and machine learning. The tool is flexible and can express a wide variety of algorithms, such as learning and deduction schemes for deep neural networks. It supports both Python and C++ languages. The Python API is more extended and its main advantage regards the way it can run on multiple CPUs and GPUs. To view the created model we can use a web platform called TensorBoard [11]. This allows us to inspect the graph of neural network; one can track the evolution of a tensor (a set of values organized in a multidimensional array) over time through a histogram and can graphically represent different desired characteristics, such as the accuracy of model.

This paper is organized as follows. First, a solution to integrate an industrial robot into an IoT cloud-based platform

978-1-5386-4444-7/18/$31.00 ©2018 IEEE

is presented. After illustrating a simple cloud application devoted to monitoring and controlling the robot, a deep learning application is explained. The paper ends with some results and conclusions.

II. IoT Solution for a Robotized Cell

A. Motivation and the used manufacturing cell

In order to extend the sensorial and decisional subsystems as needed for the object recognition scenario, a solution is required to easily integrate and establish the communication between a data acquisition application, a processing system, a learning module, decisional and execution parts. The used manufacturing environment is presented in Fig. 1 and includes: (1) HD webcam system linked to (2) image processing application (described in Section III) running on a laptop with an Intel Core i5-5200 processor, (3) a driver application running on a modest computer that is connected to the SC4-plus controller for (4) the robot IRB2400L.

Previously, we proposed a solution for the external interaction with an ABB robot controller used in several research projects [7], [12], [13]. Each time the main effort was to facilitate custom communication with other applications, such as Matlab, Java, Python and C++, through local and private networks. Other researchers have adopted the Robot Operating System (ROS) for their robots in order to interact with them [14]. This approach requires certain options to be installed on the robot controller, some of them being not compatible with older versions of robot controllers. That is why this method is not applicable for our robot. A proposed interaction method between (2) and (3) is obtained via Watson IoT platform [15] without any local connection.

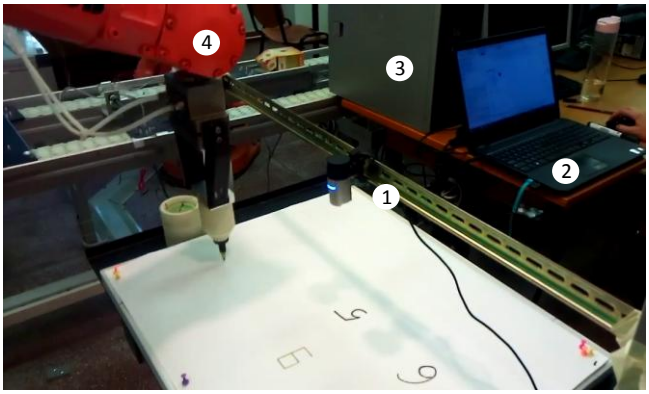

Fig. 1. Manufacturing environment used with the IoT application

B. IoT system architecture

The IoT communication solution is based on publish/ subscribe patterns, using a broker service and the MQTT protocol over TCP/IP. Two primary types of actors are in the system. IoT devices are embedded systems able to publish their states/events and receive commands within existing internet infrastructure through MQTT. IoT applications are developed to receive device/devices information and to optionally publish commands toward devices. The interaction between IoT actors is carried out by the broker service. When a device publishes something with a topic, the broker forwards the message to all applications registered to that topic. Similarly, a command is transmitted through the broker from an application to a specific device having the command registered. This architecture is illustrated in the bottom part of Fig. 2. In our solution, the IBM cloud infrastructure is used: Watson IoT as the broker service and Node-RED as IoT cloud application for testing. IoT integration of industrial equipment involves some manufacturing constraints. Even if some devices are connected to a local network, external interaction is limited by the installed software (no support for MQTT protocol) and/or by the private protocol provided by the manufacturer. In our case, the SC4 plus robot controller has only one solution supported by ABB, namely the Interlink service [16] working with the Robot Application Protocol (RAP) [17]. This service is pulling the controller (> 200 ms) in order to get states of controller, I/O channels, robot arm pose and persistent variables of running program. It also supports several

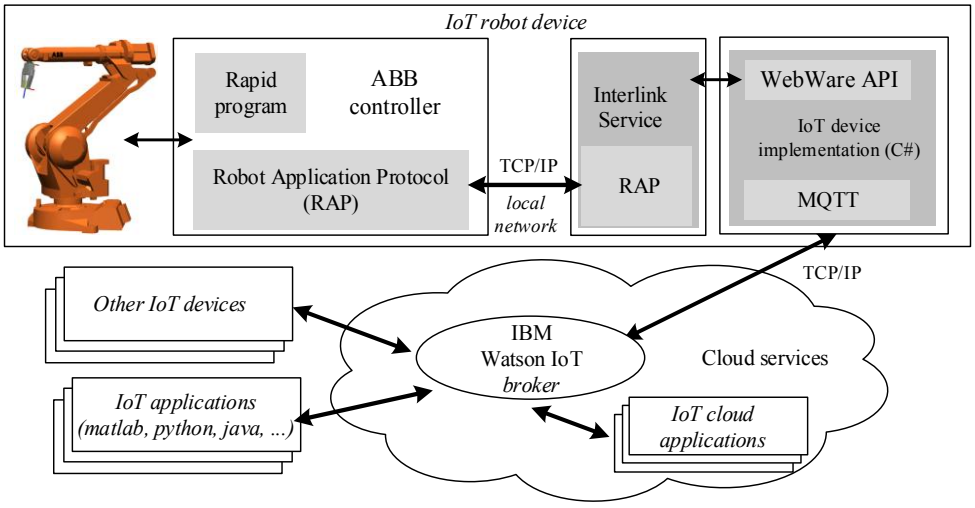

Fig. 2. IoT architecture including an industrial robot

commands: copy, load Rapid programs and run procedures, change the persistent variables and the output channels. Interlink service can be only used with a constrained WebWare SDK [16]; this is developed for Windows, supports Visual Basic and C#, needs window form application, and has no further supports.

The above issue is present to all industrial devices. For instance, in the case of our educational manufacturing cell [7], similar constraints appear for the machine tool, conveyor and computer vision system. Our solution involved the implementation of IoT device client to bridge the communication in both ways: publishing the robot's status/events and receiving and forwarding commands to the controller (see Fig. 2). This client has to be on the same computer with the interlink service and it must have access to the robot network and internet.

In order to gather robot information, we defined several topics for states: controller, program, motors, I/O, robot pose and customer program variables. These are persistent variables, being defined in the developed program loaded on the robot. The states are changed by the Interlink service when events are triggered, meaning that the IoT client is connected to that events. For example, when the motion of the robot is monitored, the IoT device is pulling the controller and publishing pose data under the JSON format (1) with the *current_pose* topic. The structure of data that store a pose of any ABB robot is robtarget (2), where [x, y, z] is the position of tool's reference frame, and [q0, q1, q2, q3] is the orientation given as quaternion.

{'x' = …, 'y' =…, 'z' =…,

'q0'=…,'q1'=…,'q2'=…,'q4'=….} (1)

robtarget p_1 = [[x,y,z], [q0,q1,q2,q3],…] (2)

Regarding the transmission of commands towards robot, the IoT client has to publish the command with a type and several parameters. We defined three command types: for controller (on/off motors), for analogical and digital output signals, and for programs (meaning the change of persistent variables and run of procedures). For example, in order to move the robot in a given position, the client application has to publish a *move_to* command with parameters for the final position (x, y, z). The command is received by the IoT robot device; first, persistent variables indicated in received parameters are updated on the robot controller and then the corresponding procedure is carried out. About this, it is important how persistent variables are chosen, so that to be appropriate for the procedure to be run on robot controller.

C. A simple IoT cloud application

Once the robot was integrated in IoT platform, it is accessible for monitoring and control from the cloud applications and also from other computers. In Fig. 3 a cloud-based application with the Node-RED tool is presented. The first goal was to monitor the robot motion and to plot the robot positions. Thus, it is used an input node for Watson IoT platform in order to receive events sent from robot with the current position, and three chart nodes to plot x, y and z axes. Figure 4 illustrates the data plotted in real-time when the robot made three repetitive tasks; the sampling period was set to 200 ms.

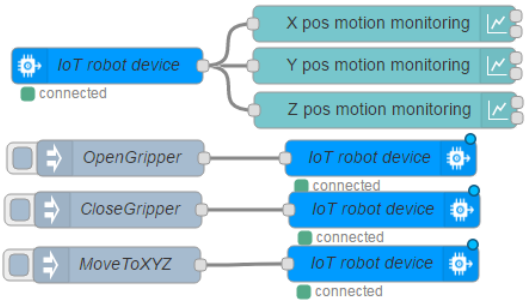

Fig. 3. Node-RED cloud application for monitoring and controlling an industrial robot

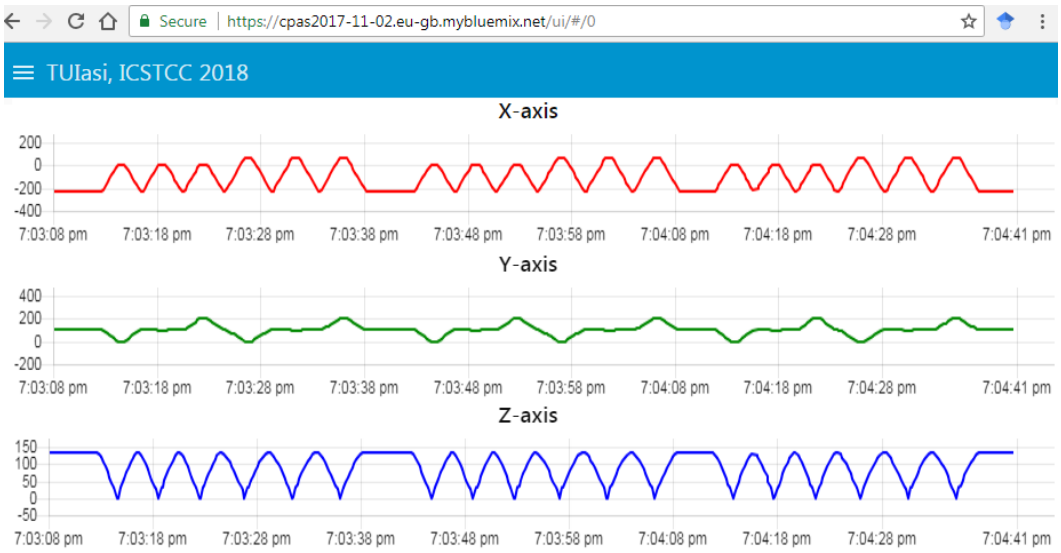

Fig. 4. Monitoring the robot motion: x, y, z axes

The second goal implied to send some commands, such as close/open the gripper and to move to a given position. For this, three output nodes are involved to publish commands toward IoT platform and three input nodes are acting as buttons. Furthermore, these three commands need to be previously registered by the IoT device and implemented consistently with the robot program. Codes 1 and 2 show the developed programs for the robot (using the Rapid language, the one devoted to ABB robots [18]) and the needed IoT device. When the MoveToXYZ command is received by IoT device, the persistent variables are updated and after that the corresponding procedure is executed.

Code 1. Example of robot program for customer commands

```
1.  PERS num x:=0, y:=0, z:=0; // external changeable variables
2.  PROC MoveToXYZ()
3.     ConfL\Off;
4.     MoveL Offs(P10_Ref, x, y, z),v200,fine,Efector;
5.     ConfL\On;
6.  ENDPROC
7.  PROC OpenGripper()
8.       WaitTime 0.2;
9.       Set DO11_7;
10.      Reset DO11_8;
11.      WaitTime 0.2;
12  ENDPROC
```

Code 2. Example of IoT robot device code for customer commands

```
1  public void processCmd(string cmd, string format, string data){
2    if (cmd.Equals("OpenGripper"))
3           AbbControler.RunRapidProcedure(robot, cmd);
4    if (cmd.Equals("MoveToXYZ")){
5         JObject msg = JObject.Parse(data);
6         AbbControler.SetRapidVar(robot, "x", msg.GetValue("x"));
7         AbbControler.SetRapidVar(robot, "y", msg.GetValue("y"));
5         AbbControler.SetRapidVar(robot, "z", msg.GetValue("z"));
6         AbbControler.RunRapidProcedure(robot, cmd);}}
```

III. Deep Learnig Based Object Recongnition

The considered robotized application applying Deep learning has as scenario the carrying out of certain tasks by an industrial robot in accordance with images received as input data. Thus, the operator writes a number from zero to nine on a paper, and the robot reacts by making one task from a set of ten jobs in our experiments, the robot's tasks were to write the presented number (see Fig. 1). The scenario can be extended to more complex tasks; one issue is about collecting the dataset to properly train a neural network.

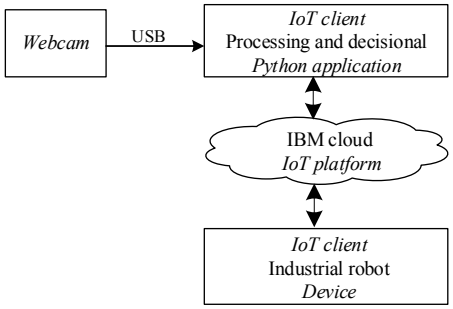

Fig. 5. IoT based architecture

The IoT based architecture is presented in Fig. 5. This is an instance of Fig. 2, where the processing and decisional part is an IoT application and the IoT robot device achieves the connection with the industrial robot. Data acquisition is performed using a webcam, images being transmitted via USB. The decision-making system (IoT application) is developed in Python; it consists of the image processing with the OpenCV library and a recognition algorithm that uses the TensorFlow library.

A. The considered learning process

To be able to recognize the transmitted images, first, the system must be trained with the learning dataset. Three activation functions were considered to train the deep neural network by TensorFlow: *Softmax*, *Sigmoid,* and *ReLU*. Each of these brought different performance. The *Softmax* function calculates the probability of each target class over all possible target classes; in our case there are ten possible classes. The *Sigmoid* function is used in binary classification and is nonlinear. The ReLU (*Rectified Linear Unit*) is inspired by biology and it is defined as the positive part of its argument (3):

$$f(z) = \max(0,z) \tag{3}$$

where z is the input variable of neuron. This function provided a better learning of deep neural network than *Softmax* and *Sigmoid* functions, as shown in the next section.

As a first phase, the learning process used an MNIST (Modified National Institute of Standards and Technology) [19] digit dataset divided into three categories: 55.000 items of learning data, 10.000 test data and 5.000 validation data. Each image is binary, with a size of 28x28 and has associated a tag with the number from the picture (see Fig. 6). The MNIST dataset was useful, allowing us to determine the best recognition percentage for each activation function applied by the developed program. One must take into account that the quality of images from the dataset greatly influences the accuracy of the learning process. This determines the need for real captured image to be appropriately processed, considering light, shadows, camera pose, and camera specifications.

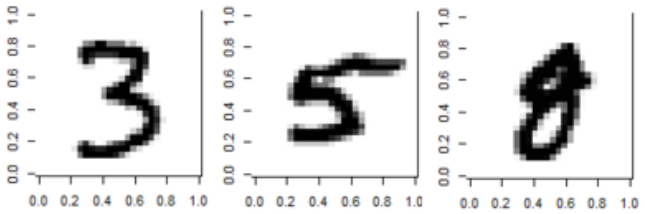

Fig. 6. Digits representation in the MNIST dataset [19]

About this, image acquisition and processing are made with OpenCV. Figure 7 illustrates the processing steps for a captured image: gray scaling, thresholding, dilatation, centralize and resize. The reasons of these steps are highlighted in the next section.

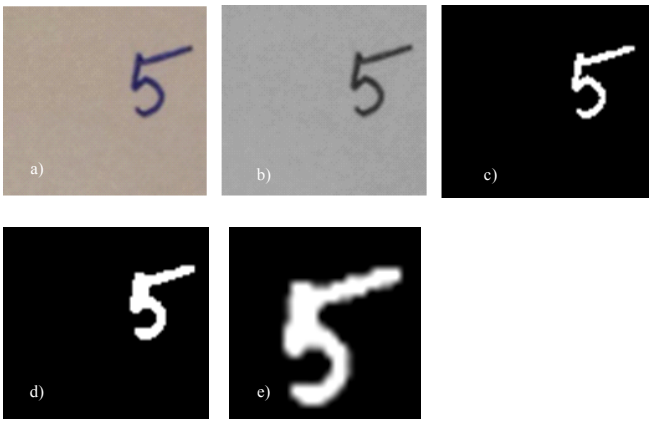

Fig. 7. a) original image b) conversion to grayscale c) thresholding d) dilatation e) centralize and resize

B. *Performance analysis*

The considered activation functions and processing steps influence the quality of recognition. In this investigation, the validation data are captured from the web camera and MNIST validation data are no longer used. For each digit, three different images are used (thus, our set includes 30 images) considering the influence of writing instrument, light, angle and quality. Figure 8 shows the considered samples for the digit 8. Moreover, the image processing includes only resizing and thresholding (see Fig. 9).

When the *Softmax* function was used, the TensorFlow built a neural network with a single layer of neurons and the percentage of recognition was up to 60% (when MNIST data validation are used, recognition percentage is up to 92%). As Fig. 9 shows, the quality of image processing is low. We can conclude that the difference in recognition of 30% appearing between the real data and that from MNIST (see Table I) is closely related to how image is processed. The Table I shows in the last row which digits are partial recognized (a bolded and underlined value is one being incorrectly recognized). For instance, the digits 7, 8 and 9 are not recognized at all.

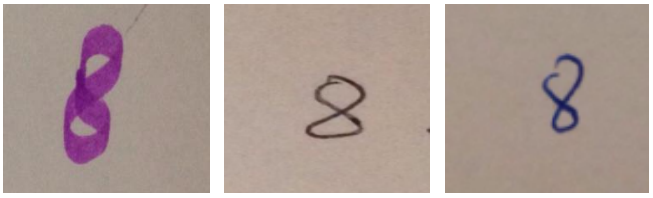

Fig. 8. Examples of images with digit 8

By considering the *Sigmoid* function, a neural network with five layers of neurons was built. The first layer has 10 neurons and this number is progressively increased to the fifth layer with 200 neurons. On the same sample of 30 images, the percentage of recognized images reaches 70% for those being captured by the camera and up to 95% for the MNIST validation set. Table I shows that more digits are recognized in each input set (at least one digit from each set). When ReLU function was used, recognition percentage increased to 83% for our images and to 96% for MNIST dataset.

To improve the recognition percentage for all three activation functions, one had to increase the quality of image processing. Images of digits from Fig. 9 differ a lot from those of Fig. 7, due to the weak contrast between the sheet and the drawn number; only simple binarization does not bring the image to the needed quality. Furthermore, image dilatation is introduced to enlarge the bright regions; it also shrinks the dark regions to join the broken areas of the number. There is a morphological closure that removes the remaining dark colored spots and connects small carpets so that dark gaps between the bright ones "close". Another change in processing lies in the fact that image resizing at 28x28 takes place at the end of the process and not at the beginning; this was done because it was seen from the tests that resizing before binarization, dilatation, and centering reduce image quality. After these improvements, better results were obtained as presented in Fig. 10 (to be compared with those of Fig.9).

With these changes in image processing, we ran the model again using the ReLU function, and we could see that the percentage of recognition exceeds 90% (see Table I, the ReLU 2 column).

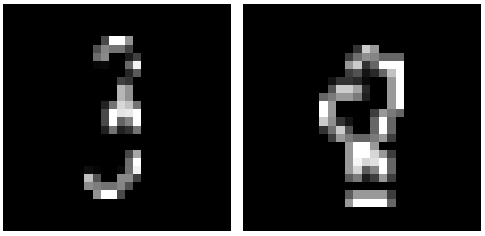

Fig. 9. Results of resizing and thresholding the image from Fig. 8

TABLE I. LEARNING PERFORMANCE USING THREE ACTIVATION FUNCTIONS

Performance	Softmax	Sigmoid	ReLU	ReLU 2
MNIST image recognition	92%	95%	98%	98%
Real image recognition	53% - 60%	70%	83%	90%
[img1 img2 img3] with/ **without** recognized values img_i – input image	[0 0 0] [**2** 1 1] [2 2 2] [3 **6** **1**] [4 **6** 4] [5 5 5] [**8** 6 6] [**2** **2** **2**] [**1** **2** **9**] [**3** **3** **3**]	[0 0 0] [**3** **5** 1] [2 2 2] [3 3 **1**] [4 4 **9**] [5 5 5] [**8** 6 6] [**2** 7 7] [8 8 **9**] [**3** **2** 9]	[0 0 0] [1 1 1] [2 2 2] [3 3 3] [4 4 4] [5 5 5] [**8** 6 6] [**2** 7 **2**] [8 8 8] [**3** **3** 9]	[0 0 0] [1 1 1] [2 2 2] [3 3 3] [4 4 4] [5 5 5] [6 6 6] [7 7 7] [8 8 **9**] [**3** **2** 9]

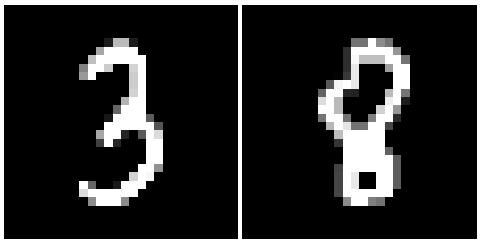

Fig. 10. Examples obtained after improving the image processing

IV. Overview Results and Conclusions

The first conclusion is about the way the whole system was integrated. Due to several platforms being used, we expected to encounter some effort to unify the robotized application. On the contrary, by seeing the system as an IoT one, the integration process got simple. The drawback is on the industrial equipment when this must be made as an IoT device. We can hope to appear a community involvement for developing IoT devices needed for the manufacturing environment, similar to the ROS community.

The cloud solution for controlling a manufacturing system has the latency issue. Our experiments showed a delay between 0.4s – 2s for commands and the direct interaction between the IoT device and robot is around 0.3s. The remained delay can be from the IoT application and broker. This issue is to be further studied.

The deep learning application developed to recognize digits from zero to nine in a real environment allowed us to understand several issues. Image processing and the activation function have a great importance, as well as the light, shadows and camera specifications. The learning process using the TensorFlow needed 32 seconds, meaning that for complex object recognition tasks it should be carried out on a proper cloud environment. Our future intention regards the application of reinforcement learning for a robotized system.

References

[1] P. Benardos, and G-C. Vosniakos, "Internet of Things and Industrial Applications for Precision Machining", Solid State Phenomena, vol. 261, pp. 440–447, 2017.

[2] S. Mumtaz, A Alsohaily, Z. Pang, A. Rayes, K.F. Tsang, and J. Rodriguez, "Massive Internet of Things for industrial applications: Addressing wireless IIoT connectivity challenges and ecosystem fragmentation", IEEE Industrial Electronics Magazine, vol. 11(1), pp.28-33, 2017.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", Nature, vol. 521(7553), p. 436, 2015.

[4] D. Wu, D.W. Rosen, L. Wang and D. Schaefer, "Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation", Computer-Aided Design, vol. 59, pp.1-14. 2015.

[5] M. Luvisotto, F. Tramarin, L. Vangelista, and S. Vitturi, "On the use of LoRaWAN for Indoor Industrial IoT Applications", Wireless Communication and Mobile Computing, vol. 2018, 2018.

[6] S. Raileanu, F. Aton, T. Borangiu, O. Morariu, I. Iabob, "An experimental study on the integration of embedded devices into private manufacturing cloud infrastructures", 8th Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing, June 11-12, Bergamo, Italy, 2018 (in press).

[7] D. Panescu, C. Pascal, and R.M. Olaeru, "A rule-based approach for a multi-robot application", 19th International Conference on System Theory, Control and Computing (ICSTCC), pp. 75-80, IEEE, 2015.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition. Proceedings of the IEEE, vol. 86(11), pp.2278-2324, 1998.

[9] OpenCV ,"Open Source Computer Vision", https://opencv.org/

[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean., M. Devin, S. Ghemawat, G. Irving, M. Isard and M. Kudlur, "TensorFlow: A System for Large-Scale Machine Learning", in OSDI, vol. 16, pp. 265-283, 2016.

[11] L. Rampasek, and A. Goldenberg, "Tensorflow: Biology's gateway to deep learning?", Cell systems, vol. 2(1), pp.12-14, 2016.

[12] D. Panescu, and C. Pascal, "Holonic coordination obtained by joining the contract net protocol with constraint satisfaction", Computers in Industry, vol. 81, pp.36-46, 2016.

[13] A. Burlacu, C. Copot, A. Panainte, C. Pascal and C. Lazar, "Real-time Image based Visual Servoing Architecture for Manipulator Robots", in VISAPP, pp. 502-510, 2011.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng, "ROS: an open-source Robot Operating System". In ICRA workshop on open source software, vol. 3 (2), p. 5, 2009.

[15] IBM IoT platform,"Architecture of the Watson IoT Platform" https://console.bluemix.net/docs/services/IoT/iotplatform_overview.htm, (accessed in 2018)

[16] ***, WebWare SDK User's Guide, ABB Flexible Automation, Sweden, 2000.

[17] ***, RAP. Service Protocol Definition, ABB Flexible Automation, Sweden, 1996.

[18] ***, ABB RAPID, Technical Reference Manual - RAPID Instructions, Functions and Data types, 3HAC050917-001, 2014.

[19] Y. LeCun, C. Cortes, and C.J.C. Burges, "The MNIST database of handwritten digits", http://yann.lecun.com/exdb/mnist/, (accessed in 2018).