Deep Learning based Parts Classification in a Cognitive Robotic Cell System

Sabin Rosioru
Asti Automation
Bucharest, Romania
sabin.rosioru@astiautomation.com

Grigore Stamatescu
Automation and Industrial Informatics
University Politehnica of Bucharest
Bucharest, Romania
grigore.stamatescu@upb.ro

Iulia Stamatescu
Automation and Industrial Informatics
University Politehnica of Bucharest
Bucharest, Romania
iulia.stamatescu@upb.ro

Ioana Fagarasan
Automation and Industrial Informatics
University Politehnica of Bucharest
Bucharest, Romania
ioana.fagarasan@upb.ro

Dan Popescu
Automation and Industrial Informatics
University Politehnica of Bucharest
Bucharest, Romania
dan.popescu@upb.ro

***Abstract*—Advanced manufacturing systems increasingly rely on intelligent algorithms to discriminate, model and predict system behaviours that lead to increased productivity. Edge intelligence allows the industrial systems to collect, compute and act based on process data while reducing the latency and cost associated to an hierarchical control system in which complex decisions are generated in the upper layers of the automation hierarchy. Greater local computing capabilities allow the online operation of such algorithms while accounting for increased performance requirements and lower sampling periods of the control loops. In this work we present the concept of a cognitive robotic cell that collects, stores and processes data *in situ* for enabling the control of a robotic arm in a production setting. The main features that characterise the robotic cell are embedded computing, open interfaces, and standards-based industrial communication with hardware peripherals and digital twin models for validation. An application of part classification is presented that uses the YOLOv4 image processing algorithm for real-time and online assessment that guides the control of an ABB IRB120C robotic arm. Results illustrate the feasibility and robustness of the approach in a real application. Quantitative evaluation underlines the performance of the implemented system.**

***Index Terms*—robot control, deep learning, part classification, YOLOv4, image processing**

I. Introduction

The Industry 4.0 and Cyberphysical Systems (CPS) paradigms support the ongoing migration from rigid automation hierarchies towards flat data acquisition, control and decision structures that emphasize an intelligent distributed periphery in industrial scenarios. In practice, local computing capabilities are exploited in order to deploy complex data processing and learning algorithms, such as multi-layer neural networks, that can continuously extract information in real time from multi-variate data streams, both scalar sensor readings and rich multimedia information in the form of images and/or video captures. The main advantages of such an approach lay in the reduced latencies for control, increased productivity and a more resource efficient design for sustainable manufacturing.

A 4C approach [1] that first includes computing, communication and control, places cognition at the top of the classification for intelligent systems. It refers to the processing and developing of knowledge from sensor data and past experiences and applying this knowledge to predict and anticipate future behaviour. At a technical level, knowledge can be encoded in neural network weights and is periodically updated with changes incurred by online collected data streams and partial retraining of the network. Integrating human-like features in a technological solution assumes limitations for the extent to which the evaluation can be carried out. In our case, the evaluation consists of quantitative metrics that reflect to which extent the performance of the robotic cell is improved after deploying and tuning the intelligent algorithm.

The use of deep learning in control applications [2] is being increasingly adopted as a model-free and data-driven technique that can process large quantities of diverse input data which can be collected in ever increasing scenarios by leveraging modern, connected, automation technology. In comparison to fully connected neural network architectures, deep learning networks include a built-in feature extraction stage that gradually propagates higher level semantics through the network towards the final output. In the case of image processing this refers to the encoding of object characteristics in increasing levels of complexity, from edges, simple shapes, complex shapes and full objects. Challenges remain for the exactness and replicability of the provided control outputs in comparison to conventional model-based techniques, optimal and robust control. Hybrid methods such as active learning [3] allow the improvement of the network performance with human-guided knowledge and domain expertise in the training phase. Alternatively, transfer learning [4] allows reuse of

Financial support from the Competitiveness Operational Program 2014-2020, Action 1.1.3: Creating synergies with RDI actions of the EU's HORIZON 2020 framework program and other international RDI programs, MySMIS Code 108792, project acronym "UPB4H", financed by contract: 250/11.05.2020 is gratefully acknowledged.

existing networks to new applications with reduced adaptation.

Several related works have been identified that described alternative approaches to robotic cell design with deep learning image processing for control. A system that includes an UR5 industrial robot with Faster-RCNN image processing is presented by [5]. Two types of time-of-flight distance cameras are used, Kinect One and ZED, in order to collect additional features. The solution is integrated with the Robotic Operating System (ROS) for increased flexibility and relaying the image detection outputs for robot control. Accuracy is compared while differentiating with the number of discriminating classes. Image-based seam tracking welding robot control using an improved edge detection algorithm in described in [6]. The authors present a pre- and post- processing approach to extract features from imgaes captured with a CCD camera of the industrial process. The welding seam is detected through an improved edge detection method and periodic image calibration. The method is considered robust to light changes and other disturbances. Deep learning for object classification for robotic disassembly and servicing is described by [7]. The Tiny-YOLOv2 architecture is used to detect cross-recessed screws while allowing for limited resource use in an embedded deployment. Using a manually annotated dataset of 900 images of 12MP resolution the authors obtain average precision (AP) metrics of up to 98%. A mobile robot platform with object detection and recognition is showcased in [8]. A combination between conventional machine learning and deep learning architectures is proposed to achieve high level representations of both visual and textual content. The deep learning component covers various typical neural network architectures: VGG16, AlexNet, CaffeNet, GoogLeNet and ResNet with various layers, that are evaluated for the presented task and achieves an accuracy of up to 87% with a significant reduction of retail surveying time. In previous work, enabling contributions for integrated industrial communication of open-source software frameworks with industrial PLC technology for motion control applications have been described in [9] and [10]. Fog computing architectures enabling distributed sensing [11] and industrial cloud sensor integration [12] also provide support for ubiquitous computing and learning.

Main contributions of our current work include:

- Conceptual description and implementation of a cognitive robotic cell system for advanced manufacturing;
- An application and evaluation of the YOLOv4 deep learning neural network architecture to parts classification.

The rest of the papers is structured as follows. Section II presents the concept of a cognitive robotic cell with embedded learning support for intelligent data processing and digital twin model that extends the application space through scalable experimentation. Section III details the YOLOv4 deep learning image processing algorithm and provides the requires justification for its use in our work. In depth results are discussed in Section IV together with technical details that showcase the performance of the implemented solution. Section V concludes the paper with outlook on future work.

II. System Description

This section presents the conceptual design associated to the robotic cell and its extensions toward cognitive features. Figure 1 illustrates the components and interactions that support higher level functions for the robotic cell systems. The main components of the physical robotic cell are the following:

- Robot: the robot system includes the 6 DoF robotic arm together with the robot controller and gripper;
- PLC: the programmable logic controller coordinates the functionality of the robot arm with the other components of the system for achieving high level tasks; the PLC includes the central processing unit, input-output modules for analog and digital signals and suitable industrial communication modules;
- Drive system: the drive system includes a servodrive for accurate positioning of a conveyor belt module; it carries out the synchronization required between the pick-and-place operation and the part transport;
- HMI: a touch screen human machine interface is provided for local visualisation and operation of the process state; it can also display alarms and error codes suitable for on site debugging and fault remediation;
- Industrial communication: PROFINET is used as standards-based industrial communication network that connects all the main components of the robotic cell; it allows real-time features that guarantee deterministic communication for control; for local identification and part tracking, an RFID module with associated tags is also employed;
- Energy Meter: energy metering at the robotic cell level allows the profiling of individual operations and the construction of a data-driven predictive model to flag anomalous functioning through spikes and other types of consistent deviations in the energy traces.

In addition to the physical realisation a digital twin model is operating in parallel to the real system. It allows extended testing and simulation of the physical behaviour and the generation of synthetic data traces to improve performance of the control logic. This limits wear and tear of the real system and can prevent certain dangerous and unsafe operation conditions in unknown system states. The digital twin model is linked with the physical system via a Model-based Design (MBD) environment and supporting tools, such as Festo CIROS. It allows accurate, part level, modelling of the various mechanical and electrical components. Functionality is implemented by running the same PLC code and robot arm routines as in the existing equipment. At the PLC level, a dedicated embedded machine learning library can be used to pre-process and model the sensor data traces for supervised and unsupervised learning close to the manufacturing process. The digital twin model can also be deployed in a cloud environment for continuous operation where it can be periodically updated according to observed behaviour in the physical system. The solution inclues both proprietary software modules from the manufacturers of the automation equipment, as well as open-

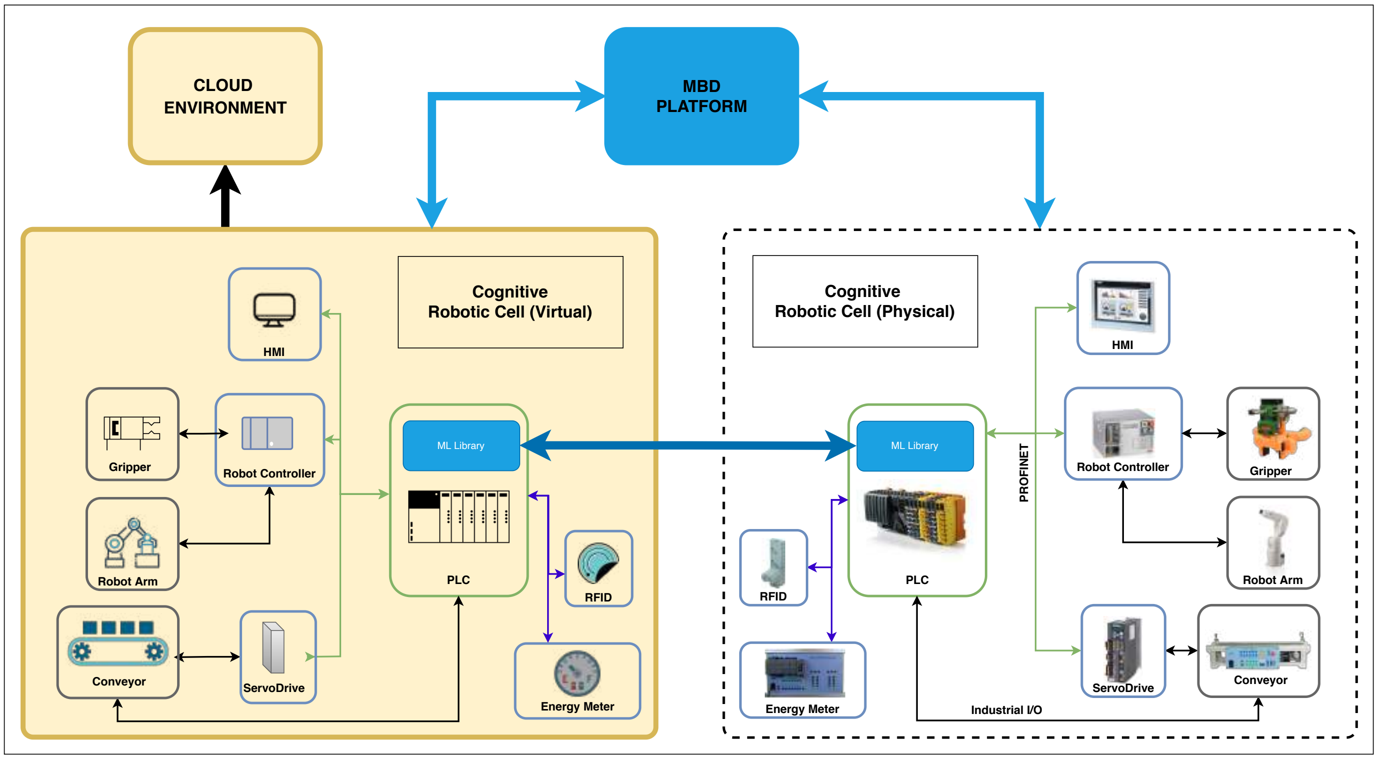

Fig. 1: Conceptual Diagram of the Cognitive Robotic Cell

source software components that enable the local integration and connection to third-party modules.

A dedicated camera subsystem is included as add-on accessory. It is used for realising object detection and identification and the vision-based control of the robotic arm.

III. Computational Algorithms for Parts Classification

There are currently two categories of algorithms that identify objects, namely what is called ”one-shot” or ”multi-shot” detection. A one-shot model is capable of identifying an object without applying a preliminary step. On the other hand, multi-shot models use a preliminary step in which regions of importance are detected and then objects in these regions are identified. Although multi-shot models have higher accuracy, one-shot models are much faster, allowing real-time detection. A well-known algorithm often used in machine vision applications is YOLO [13]. This architecture does not look at the whole image. Instead, is scans parts of the frame which have high chance of containing the object. In YOLO or You Only Look Once a single convolutional network predicts the bounding boxes and the class probabilities for these boxes. YOLO is orders of magnitude faster than other object detection algorithms, like CNN, R-CNN. The limitation of YOLO algorithm is that it struggles with small objects within the image.The network architecture is based on a multi layer structure made up of convolutional layers, 53 in total, which have the main purpose of extracting the essential features from a picture. The activation function used is ”Mish” and provides a way to improve the training process by increasing the network accuracy.

$$f(x) = x * tanh(ln(1 + e^x)) \tag{1}$$

In order to evaluate several algorithms, there needs to be some type of metric which can be used to rank them. In the case of object detection networks, the mean average precision (mAP [%]) is used. It compares the detected box with the ground-truth bounding box and returns a score. The accuracy of the model is directly proportional to this performance metric. The loss function consists of three parts: regression loss, confidence loss and classification loss. The last one, is based on the Euclidian distance between the predicted box and the ground truth. Both of the remaining two metrics are based on cross entropy error. [14].

$$f = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha\upsilon - \tag{2}$$

$$\sum_{i=0}^{s^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left[\hat{C}_i log(C_i) + (1-\hat{C}_i)log(1-C_i)\right] -$$

$$\lambda_{noobj}\sum_{i=0}^{s^2}\sum_{j=0}^{B} 1_{ij}^{noobj}\left[\hat{C}_i log(C_i) + (1-\hat{C}_i)log(1-C_i)\right] -$$

$$\sum_{i=0}^{s^2} 1_i^{obj} \sum_{c\in classes}\left[\hat{p}_i(c)log(p_i(c)) + (1-\hat{p}_i(c))log(1-p_i(c))\right]$$

IV. Results

Before implementing the physical cell, a preliminary simulation has been developed. Using Siemens S7-PLCSIM, we were able to develop the software capable of controlling the installation. Besides the Siemens solution, there was also created a simulation counterpart using the B&R Automation Studio. The signals for the communication of different entities were transmitted into a Node-Red instance using OPC-UA and S7 Communication. Thus, other simulated processes could be integrated into the virtual cell. The robot movement has been tested using the Festo solution, CIROS. The drawback is that the Festo simulation software, doesn't offer support for ABB robots. Instead, a Mitsubishi arm has been selected. After the construction and validation of the simulation, the physical implementation was started.

The results for this application were obtained by overlapping a layer of artificial intelligence onto an industrial process. Below the physical equipment: Siemens 1200 PLC, display and ABB robotic arm, we have implemented a dedicated intelligent software layer, capable of identifying and returning the spatial coordinates of the desired object. The framework used for the classification method is YOLO (You Only Look Once) [15]. A picture of the physical realisation of the robotic cell used in the experiments is presented in Figure 2.

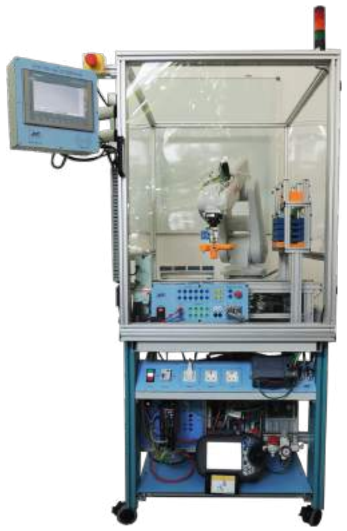
Fig. 2: Cognitive robotic cell system

The main objective is to identify a solution for the industrial process which could be solved by using artificial intelligent structure in order to optimize the process by reducing the down-time and improving the production time. Multiple versions of the YOLO algorithm were tested, but finally the fourth iteration was chosen. A brief comparison of the existing architectures can be observed in Table I, which presents the performance of the YOLOv3_tiny, YOLOv3, YOLOv4_tiny and YOLOv4 networks on realistic datasets created by our experiments. The mAP, loss function and identification time, in miliseconds, are reported.

Network	mAP(%)	Loss	Identification
YOLOv3_tiny	19.68	1.9065	21.9 ms
YOLOv3	46.41	0.5065	42.2 ms
YOLOv4_tiny	49.78	1.485	19.6 ms
YOLOv4	94.45	0.3123	43.8 ms

TABLE I: Performance of various YOLO generation on the parts classification problem

The application runs in the cloud environment using the Google Colab Platform. Due to the restriction created by this approach, the communication between the cognitive application and the physical hardware is done using a gateway created via Node-Red. When the network identifies an object it will generated a 4 point coordinates system for each one. Thus, the information can the be sent to the control level PLC and the system will integrate the new information into its software. Communication between the cloud and the gateway is done by reading/writing CSV files, and the connection of the gateway to the PLC is done using S7-Communication (Figure 3).

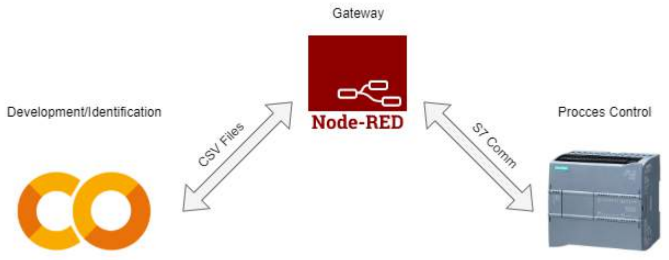

Fig. 3: Diagram of the control solution

The first step in building such an application is creating a set of images that will be used for training and validation. The database contains 90 pictures for each of the object that will be identified. In this specific application, we must identify and classify between three types of small pieces used in the assembly process. Also, the resolution of these images has been taken into account, as higher resolutions will improve the network identification ability. Based on the fact that the position of the camera is in the relative vicinity of the identification space, we concluded that almost any type of camera can be used. In order to reduce the cost of the system, we settled on a 720p, corresponding to 1280 pixels width and 720 pixels height, camera. The YOLO architecture was chosen because the system requires a fast and precise identification procedure. Also, unlike the industrial counterpart, this system is not placed in perfect conditions regarding the identification. Even the smallest change of light can drastically impact the process, so the neural network must be robust when faced with such perturbations. The solution has been tested using both natural light and artificial light. Under cold light conditions, over 5000 Kelvin, the detection has the best precision. The environment in which the cell has been tested could not take into account the fine particles of dust that are commonly found in an industrial setting.

The next step is training the YOLOv4 network to detect three types of objects. For this, we used the Google Colab Platform, which offers free computational resources. The scope of this application is to develop a real time system. Thus, using a low performance laptop or computer, hardware speaking, would not be enough. The software is written using the Python programming language. The framework on which the architecture is working, is based on the C++ language. Neural networks can be viewed as an optimization problem where the scope is to minimize a loss function. This describes how well the network is able to identify the objects based on the training and validation sets on which it was trained. The ratio of training to validation data used was: 5 to 1. The neural network has been trained until the model could not reduce its loss any further. Thus, after training for 5400 epochs the model could not be improved anymore as it is shown in Figure 4. Figure 5 presents the classification confusion matrix with the types of classification errors encountered during testing.

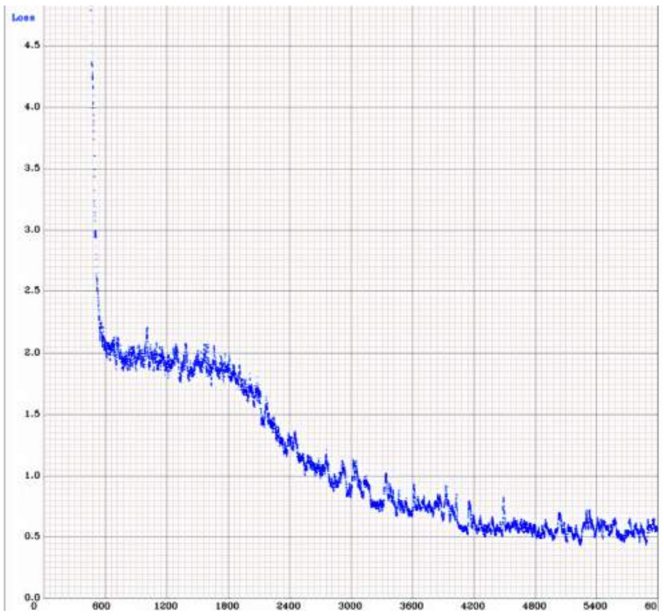

Fig. 4: Evolution of the loss function

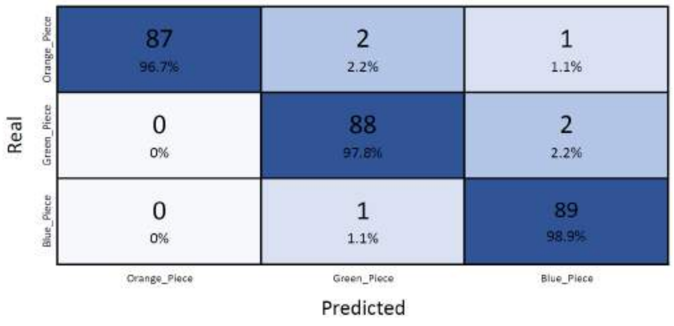

Fig. 5: Confusion matrix

We continued by creating a script capable of continuously running the identification software that also generates a set of coordinates for each detection, as in Figure 6 and Table II. This piece of software also runs in the cloud. Due to this approach, the information regarding the detection is sent to the control unit using a gateway capable of interpreting comma separated value (CSV) files and converts the data to be sent via S7-Communication. Each detection is able to give the spatial coordinates of the object in pixels reported to the top left corner of the frame. The position of the camera is fixed related to the identification plan, and so we are able the compute the transformation from pixels to millimeters using an known object as a reference.

Px_Left	Px_Top	Px_Right	Px_Bottom	Type
283	166	361	221	Bl_Piece
304	241	378	309	Bl_Piece
440	159	515	224	Or_Piece
235	243	273	305	Gr_Piece
436	243	475	306	Gr_Piece

TABLE II: Parts coordinates

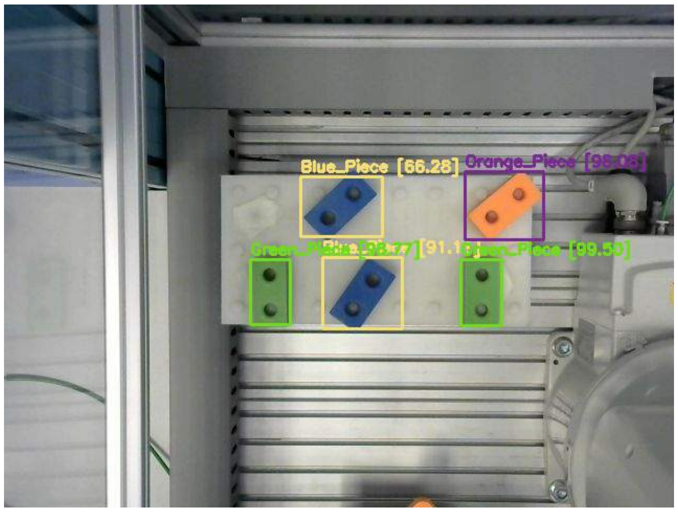

Fig. 6: Example of identification

A drawback of the architecture is that the software is capable of generating the coordinates of the object based on a rectangle shape, not on the object itself. Meaning that the values detected are not related to the corner of the object. This can be mathematically corrected. We used equation 3 to recalculate the true positions and also to transfer the measurement of the system from pixels to millimeters, because this information is used by the robot which needs to know the two offsets on x and y axis and the rotation of the object. The calculation is based on the real dimension of the object. In this application, the small rectangular piece has four measurements. (X_1, Y_1) represent the real measurements, and (x, y) represents the measurements calculated from the detection.

$$\alpha = \arcsin\left(\frac{y - x * (\frac{y_1}{x_1})}{1 - (\frac{y_1}{x_1})^2}\right)/0.01745329252 \tag{3}$$

Having the actual rotation of the object and its spatial coordinates means that now the system can precisely calculate the movement of the robot and the best position for the gripping procedure. The programming of the ABB robotic arm is made using RAPID language. Also, the cinematic calculation of a trajectory is done by the integrated controller. As a final acceptance test, the cell has been subjected to series of tasks designed to test the robustness of the process and the precision of the neural network.

V. Conclusion

We have presented a system architecture used for object detection into a cognitive robotic cell. The solution includes the implementation of a neural network based on YOLO architecture, the API which allows the communication of the software with the psychical hardware and also the programming of the robotic cell in order to fulfill its desired behaviour. Future work will focus on validating the real-time performance of this solution in a real production environment.

References

[1] L. Liu, S. Liu, and W. Shi, "4c: A computation, communication, and control co-design framework for cavs," *IEEE Wireless Communications*, vol. 28, no. 4, pp. 42–48, 2021.

[2] I. Matei, R. Minhas, M. Zhenirovskyy, J. d. Kleer, and R. Rai, "Deep learning for control: a non-reinforcement learning view," in *2020 American Control Conference (ACC)*, 2020, pp. 2942–2948.

[3] C. Wang, K. V. Hindriks, and R. Babuska, "Active learning of affordances for robot use of household objects," in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 566–572.

[4] M. K. Helwa and A. P. Schoellig, "Multi-robot transfer learning: A dynamical system perspective," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4702–4708.

[5] X. Chen and J. Guhl, "Industrial robot control with object recognition based on deep learning," *Procedia CIRP*, vol. 76, pp. 149–154, 2018, 7th CIRP Conference on Assembly Technologies and Systems.

[6] N. Banafian, R. Fesharakifard, and M. B. Menhaj, "Precise seam tracking in robotic welding by an improved image processing approach," *The International Journal of Advanced Manufacturing Technology*, vol. 114, no. 1, pp. 251–270, 2021.

[7] D. P. Brogan, N. M. DiFilippo, and M. K. Jouaneh, "Deep learning computer vision for robotic disassembly and servicing applications," *Array*, vol. 12, p. 100094, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2590005621000394

[8] M. Paolanti, L. Romeo, M. Martini, A. Mancini, E. Frontoni, and P. Zingaretti, "Robotic retail surveying by deep learning visual and textual data," *Robotics and Autonomous Sys.*, pp. 179–188, 2019.

[9] R.-A. Luchian, G. Stamatescu, I. Stamatescu, I. Fagarasan, and D. Popescu, "Iiot decentralized system monitoring for smart industry applications," in *2021 29th Mediterranean Conference on Control and Automation (MED)*, 2021, pp. 1161–1166.

[10] R.-A. Luchian, S. Rosioru, I. Stamatescu, I. Fagarasan, and G. Stamatescu, "Enabling industrial motion control through iiot multi-agent communication," in *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, 2021, pp. 1–6.

[11] V. Mihai, C. Dragana, G. Stamatescu, D. Popescu, and L. Ichim, "Wireless sensor network architecture based on fog computing," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2018, pp. 743–747.

[12] O. Chenaru, G. Stamatescu, I. Stamatescu, and D. Popescu, "Towards cloud integration for industrial wireless sensor network systems," in *2015 9th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, 2015, pp. 917–922.

[13] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[14] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 12 993–13 000, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6999

[15] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.