A Deep Learning Approach for Automatic Counting of Bales and Product Boxes in Industrial Production Lines

Rafael J. Xavier[1], Charles F. O. Viegas[1], Bruno C. Costa[1], and Renato P. Ishii[2(✉)]

[1] See Working, Goias, 405, 79020-100 Campo Grande, MS, Brazil
{rafael,charles,bruno}@seeworking.com
[2] Federal University of Mato Grosso do Sul, Campo Grande, MS, Brazil
renato.ishii@ufms.br
https://seeworking.com, https://www.facom.ufms.br/

Abstract. Recent advances in machine learning and computer vision have led to widespread use of these technologies in the industrial sector. Quality control and production counting are the most important applications. This article describes a solution for counting products in an industrial production line. It consists of two main modules: i) hardware infrastructure and ii) software solution. In ii) there are modules for image capture and product recognition using the `Yolov5` algorithm and modules for tracking and counting products. The results show that our solution achieves 99.91% accuracy in product counting and classification. Furthermore, these results were compared to the current manual counting system used in the industry considered in this study. This demonstrated the feasibility of our solution in a real production environment.

Keywords: Detection · Classification · Tracking · Counting · Machine learning · Deep learning · Industry 4.0

1 Introduction

Nowadays, the automation of the productive chain has been a major factor in the industry business. Increasingly more people are being replaced by machinery that is, in many cases, more efficient and productive, especially in repetitive tasks [4]. Indeed, all the cycles of production and supply networks are being integrated and automated through emerging technologies that provide real-time data gathering, allowing better analysis and information for the organization. This trend is being recognized as the fourth industrial revolution or Industry 4.0. This concept puts the industry business in a new level of technology, integrating several areas of knowledge like: the Internet of Things (IoT), Computer Vision and Big Data, with the goal to make industries more flexible and adaptable, providing a productive chain that can automatically change itself to make customized products in large scale [14].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022, LNCS 13375, pp. 619–633, 2022.
https://doi.org/10.1007/978-3-031-10522-7_42

Recently, computer vision has been used to automate production lines and replace humans [12]. This technology is strongly supported by academics in counting and checking products. Neural networks are one of the most important of these technologies for classification and object recognition. The YOLO network in particular has recently gained a lot of insights [5,9–11] and is becoming the most important network for recognition. However, many of the solutions available on the market are expensive and no longer economically viable for small businesses. This paper discusses a solution that uses computer vision and artificial intelligence and aims to reduce the costs associated with current processing capacity in small and medium-sized enterprises.

Semalo Industry[1], a snack industry with plants in Campo Grande – MS and João Pessoa – PB in Brazil and Asunción in Paraguay, has advanced in automation and technology. Several recent investments have been made in automatic product packaging machines and production control software. Despite the progress, the factory has many difficulties in controlling its stocks, especially in reconciling the quantities of products entering and leaving the expedition area. Often the shop management system shows negative stocks, which is a serious anomaly. The production count is done visually by the employee responsible for taking products off a conveyor belt (finished and packaged products from the production area) and placing them on pallets. Problems occur when there is a difference between the production count and what actually arrives in the warehouse and when pallets are sent directly to delivery trucks due to contractual deadlines, making stock control difficult.

In this sense, the objective of this paper is to propose a solution that uses computer vision in the form of object recognition to automatically count bales and packages of storable products on pallets that provide satisfactory accuracy and performance for a real-time application. To perform this evaluation, we consider the following steps: Image capture, bounding box labelling, training the deep learning network, testing and collecting the results. The results show that our solution achieves 99.91% accuracy in product counting and classification. Furthermore, these results were compared with the current manual counting system used in the industry considered in this study. This demonstrated the feasibility of our solution in a real production environment.

The rest of this article is organized as follows: Sect. 2 describes related work on both tasks that uses similar methods to our approach. In Sect. 3, we describe the neural network architecture and its properties. In Sect. 4, we explain our solution approach in detail. The data set and the modeling problems are presented in Sect. 5. In Sect. 6, we present and discuss the experimental results. After discussing these results, we draw a conclusion and mention our thoughts on future work.

[1] https://semalo.com.br/.

2 Related Works

In this section, we review the recent detecting, tracking, and counting-based methods in the literature. Especially, we reveal the limitations in very few existing count classification-based methods to show the necessity of our research. We also discuss some works dealing with the imperfect targets to show the superiority of our method.

In the study [8], the authors present a method for detecting and counting pistachios transported on conveyor belts. These pistachios are then classified into two types (opened and closed). The study recorded a precision rate of over 92% using the `RetinaNet` network [7] with a pre-trained `ResNet50` backbone. The study [6] uses smart cameras that run detection and counting algorithms. This technology allows for faster processing since no data needs to be transmitted, and is considered ideal for industrial inspections with large amounts of data. However, smart cameras are very expensive and not easily accessible. Using YOLO9000 [9], the authors [3] propose a method to integrate images from different cameras in a production line and process them as a single image. With this technique, the product can be tracked throughout the production line. However, according to the authors, this method requires a large data stream and high computational costs. Ulaszewski et al. [13] apply computer vision methods for real-time egg detection on a production line. They evaluate 3 neural network models: SSD-Mobilenetv2, FR-CNN, and `Yolov3`, and they identified that `Yolov3` was the best performing with 93% effectiveness of detection, tracking ad counting process. Bahaghighat et al. [1] propose a new model for blister counting in drug production lines. The total accuracy of more than 88% was obtained based on our best model using Haar cascade, Radon transform, and KNN. These previous works do not provide information about your performance related to the manual real counting results in the conveyor belt.

3 Neural Network Architecture

To solve the above problems, this study proposes to use `Yolov5` as the main tool to identify and classify products in industrial production lines. For this purpose, we need to re-train the network with the products we want to identify. Therefore, we had to create an extended data set with the object classes (boxes and bales) of the numerous products transported in Semalo's production lines.

3.1 YOLOv5

In this section, we discuss the implementation of `Yolov5` [5]. Notice that this version is implemented using the `Pytorch` framework, different from previous versions. Its architecture is robust, with easy implementation and integration with any other python software. Moreover, `Yolov5` brings 4 different architectures, each one with its own pre-trained models with different performances and assertiveness.

In this section we discuss the implementation of `Yolov5` [5]. Note that unlike previous versions, this version was implemented using the `Pytorch` framework. Its architecture is robust and easy to implement and integrate with any other Python software. Also, `Yolov5` provides 4 different architectures, each with its own pre-trained models with different performance and assertiveness.

Table 1 shows the differences between these networks and their mAP (mean Average Precision – the higher the better). `Yolov5x` has 2.2% higher accuracy than `Yolov5l`, although `Yolov5l` is 37.7% faster than `Yolov5x`.

Table 1. Comparative precision and speed table of different `Yolov5` versions.

Model	Size (pixels)	mAP val 0.5:0.95	mAP test 0.5:0.95	mAP val 0.5	Speed V100 (ms)
YOLOv5s	640	36.7	36.7	55.4	**2.0**
YOLOv5m	640	44.5	44.5	63.1	2.7
YOLOv5l	640	48.2	48.2	66.9	3.8
YOLOv5x	640	**50.4**	**50.4**	**68.8**	6.1

Therefore, we chose the `Yolov5l` network for our proposal because it requires less processing capacity and allows running multiple instances of the model on less expensive and processing-limited hardware without compromising the accuracy of the solution.

3.2 Metrics

`Yolov5` uses many different metrics to calculate the quality of the network. The most important of these is the mean mAP, which uses Intersection over Union (IoU). Intersection over Union is a scoring metric used to measure the accuracy of an object detector in a given data set. In the numerator of the Eq. 1, we calculate the area of overlap between the predicted bounding box and the ground truth bounding box. The denominator is the area of union, or more simply, the area enclosed by both the predicted bounding box and the ground truth bounding box. If you divide the area of overlap by the area of union, you get our final value – the intersection over union.

The IoU, which tests the similarities between two records is expressed in the following Eq. 1.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{1}$$

Here A and B are surfaces and $J(A,B)$ is the coefficient of 0 and 1. The higher the value, the greater the similarity between two sets. This coefficient is used in object recognition to test how close the model recognition is to a human recognition. Based on the tested coefficient, we can calculate whether the detections are positive, false-positive or false-negative.

Algorithm 1: Type of Detection Decision

```
if Valid detection then
    if IoU >= 50% then
        Positive Detection;
    else
        False-Positive Detection;
    end
else
    False-Negative Detection;
end
```

Considering $IoU \geq 50\%$ the Algorithm 1 therefore defines the type of detection.

Furthermore, the metrics of performance, precision and recall are defined in Eq. 2 and 3, respectively.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall(TPR) = \frac{TP}{TP + FN} \tag{3}$$

where, TP = *True-Positive*, FP = *False-Positive* and FN = *False-Negative*.

The metric *average precision* required to calculate the mAP is given by the area *under the curve* of each class, represented by the Eq. 4.

$$AP = \int_0^1 p(r)\, dx \tag{4}$$

The mAP 0.5 is the computation of the average accuracy, given by Eq. 5, of all classes computed by AP with the condition IoU > 0.5. The mAP 0.5 : 0.95 is the average of AP with IoU > 0.5 to Iou > 0.95, where the different instances of IoUs are added with 0.5. Thus, mAP 0.5 : 0.95 is the most important metric for measuring the quality of the model, because it measures not only the quality of the network classification in a test data set, but also the quality of the object localization in an image using different IoUs.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} APi \tag{5}$$

4 The Solution Architecture

The developed solution enables the capture, decoding, recognition, tracking and counting of all products in the Semalo industry. In this section we describe in detail how we developed the physical and technological part of our solution.

4.1 Physical

In Semalo's production line, the camera is located on the side of the conveyor belt that transports products between the production and storage areas. When a product is detected by a camera, its image is captured and sent to a server. To perform the counting in real time and avoid data loss, we send the images over a high-speed wired network. Figure 1 illustrates this process.

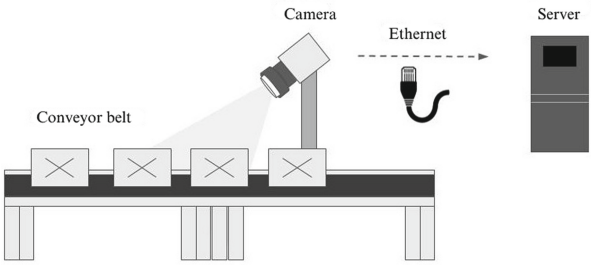

Fig. 1. The diagram shows the process of capturing and uploading images through a communication network.

By positioning the camera on the side of the conveyor belt and perpendicular to the products, we were able to capture more details of the bales and boxes, such as brand, color, logo, shape and texture. These features are important because many products are very similar and often differ only in color and size. Figure 2 shows some examples of products.

In addition, the production is divided into 5 different conveyor belts according to characteristics. On conveyor belts 1 and 2 the corn chips are transported, they go through the extrusion process, then they are roasted, flavored and packed in bales or boxes made of plastic. Conveyor belt 3 transports the conventional wheat or corn chips. They are roasted, shaped and also packed in bales and boxes.

Finally, conveyor belts 4 and 5 carry chips made mainly by potatoes, these are only packed in boxes. Table 2 overviews the output of products on the conveyor belts.

Finally, conveyors 4 and 5 carry French fries, which consist mainly of potatoes and are packed only in boxes. The Table 2 gives an overview of the product outputs on the conveyors.

Table 2. Table of the groups of conveyor belts by type of products.

Group of conveyor belts	Types of products	Packing
Conveyor belts 1 and 2	Extruded	Bales and boxes
Conveyor belt 3	Conventional chips	Bales and boxes
Conveyor belts 4 and 5	Potatoes	Boxes



Fig. 2. Example of packed products **Kró** detected by our solution in the moment of count.

4.2 Technological

The software that does the acquisition, detection, and counting was created in Python. We divided this software into 5 modules (initialization, video, detection, tracking, and counting). The following diagram Fig. 3 shows the basic interactions between these modules.

The first module is the initialization. It contains a `Docker` configuration file that initializes the application without requiring you to manually check dependencies. It also installs the application in a container that is separate from the main computing environment. Also, this module initializes the cameras and prepares them for the next module. The second module is the video module. It receives the images captured by the cameras via the RTSP (Real Time Streaming Protocol) protocol and converts these images to a 640x640 resolution compatible with the `Yolov5` detector.

The third module takes care of recognition, fetches the converted frames, and, using Yolov5, returns all the bounding boxes with their respective product classifications. So, this information is sent to the next module.

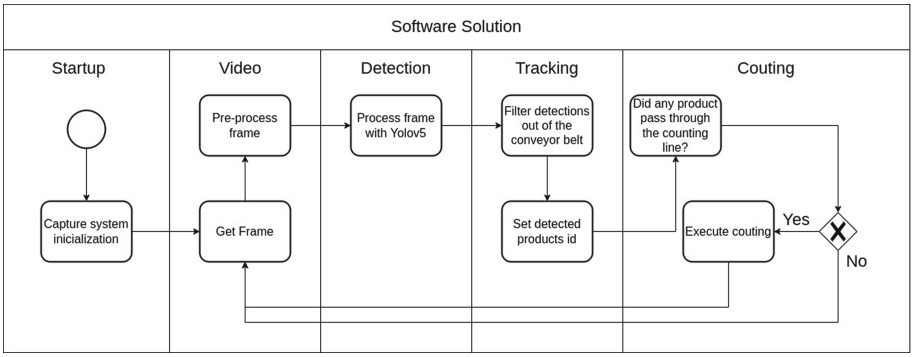

Fig. 3. Diagram with the basic interactions between our solution modules.

The fourth module is tracking, which first filters what is not on the conveyors. This module, which is one of the most complex parts of the solution, has gone through several iterations. The first was based on tracking products by their centroids. In this method, we tracked a product through 3 virtual lines, 2 vertical, the first of which is responsible for marking a product as ready to count and the second for marking a product as ready to count in our database. The third line is horizontal and filters products from the conveyor belt. Figure 4 shows these lines in Semalo's industry. These triggers were activated whenever the centroid of a product passed through the lines. This method works well for tracking people and objects, but in this case, where products were transported along a conveyor belt, it proved to be faulty at moments when products accumulated and got stuck in the belt. When this happened, IDs were swapped or duplicated, leading to errors in the count.



Fig. 4. Frame captured in the conveyor belt with the virtual lines for counting and filtering

Therefore, we reconstructed the tracking and counting software using an open-source technology called SORT [2]. This algorithm also uses centroids and Euclidean distances, but counts with the prediction of an object in the future. This helps with conjugation between a product in a previous frame and the same product in the current frame, making it a good solution for our tracking.

We made several modifications to help with the counting, loss of frames, detection failures and classification by the neural network. For this, we implemented one algorithm, to operate with SORT, that saves the history of classifications and its probabilities when a product passes through the conveyor belt. This way, when a product passed through the counting lines it was possible to verify the most common classification for such a product and define a normalized product, lowering the chances of a wrong count because of factors like illumination, temporary loss of connection and other external influences. Algorithm 2 is used to update a unique products position when they are being carried by a belt.

We made several changes to facilitate counting, frame loss, detection errors, and classification by the neural network. For this purpose, we implemented an algorithm that works with SORT and stores the history of classifications and their probabilities when a product passes through the conveyor belt. In this way, when a product passes through the counting lines, it was possible to check the most frequent classification for such a product and define a normalized product, reducing the probability of a wrong count due to factors such as lighting, temporary interruption, and other external influences. The Algorithm 2 is used to update the position of a single product when it is worn by a belt.

Algorithm 2: *SORT* update per frame

```
foreach Registered product do
    if Correspondent detected in frame then
        Returns updated product position;
        Returns updated normalized product;
    else
        Returns predicted product position;
    end
end
```

5 Data Set Construction and Modeling

To enable the training of the neural network, we first had to create the data set.

5.1 Data Set Construction Methodology

Due to the variability and availability of the 68 products produced by Semalo, we built the data set iteratively. In the beginning, only a few examples of each

product were included. As the solution improved, we took new images to try to improve the low accuracy in classifying and identifying the products. One challenge we faced was the change in brightness over the course of a day in the industry. To solve this problem, we had to take images throughout the production cycle (morning, day, evening and night). In this way, we were able to improve the generalization of our model.

However, during the measurement tests in production, it became clear that it would not be possible to eliminate all the variations that could occur in a product and affect its classification. Therefore, to improve generalization, we took and annotated the product images in a laboratory environment, a neutral and completely different environment from the production environment. Figure 5 shows an example of image acquisition in the lab environment and others in production.



Fig. 5. Image captured in the laboratory and in production, respectively.

5.2 Data Set Features

In total, we annotated and merged 199,356 images, of which 79,083 were taken in production and 120,458 in the lab. There are 68 types of unique products manufactured by Semalo. We took these images in different environments that

vary in terms of lighting, camera, assembly line, and background. The main camera we used was the `HikvisionDS-2CD3025G0-I (B)`, with the RTSP recording protocol. We labeled the bounding boxes to identify the products in the images, which is essential for training the algorithm. This task was done using `LabelImg`. This tool exports the labels with their coordinates in a format compatible with `Yolov5`.

We had to perform many tests to properly define the basic parameterization of the network. We considered accuracy, training time and overfitting.

5.3 Network Training

As explained in Sect. 5.1, when training the model, we divided the products into the same groups used in Semalo's industry. This subdivision improved the accuracy and reduced the number of models. The performance of the model is shown later in Sect. 6.

5.4 Network Configuration Tests

In order to configure the parameterization of the network, two tests were performed. In the first, epochs were defined, i.e., the number of iterations a network performs through a data set. The number of epochs affects accuracy and had to be defined to reduce training time while maintaining the accuracy of the network. With 200 epochs and the amount of data collected, the training would take 8 days.

For the epoch test, we assumed that the data set consisted of images taken in different environments. Then we randomly ordered it and used 80% of them for training and 20% for testing.

We tested the following number of epochs: $10, 25, 50, 100$ and 200. With a number of 924 products divided into 4 different types, we chose 50 epochs. This resulted in an accuracy of 99.7%, a good processing time without affecting the quality of the network, and solved the problem of overfitting that occurred with longer training times. Figure 6 shows the training hours required for each epoch and their accuracy in counting and classifying.

We then tested how the images are distributed between tests and training. We tested the following distributions: 1 – completely random distribution, 80% for training and 20% for tests; 2 – completely random distribution, 50% for training and 50% for tests; 3 – distribution with images from the lab and production, 10% of each data set was randomly transferred to each other. For illustration, using 4 and 5 conveyors as examples, we used $59,423$ images from the lab and $55,059$ from production for testing. After separating these 2 groups, we took 10% from each of these groups and mixed in the others.

The results showed that the best distributions were 1 and 3, both had a hit rate of 100% in counting and classifying, the 3 distribution had a small advantage over the 1 in the mAP@.5 test, reaching 91.2%, while the 1 distribution reached 90.4%, a difference of 0.8%.

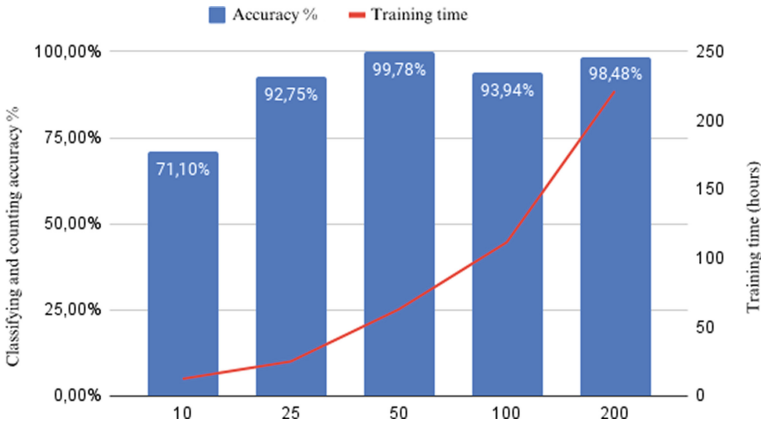

Fig. 6. Epoch tests with counting and classification accuracy and their training time.

In summary, the final baseline network configuration includes 50 epochs and a distribution with images from the lab and from production, with 10% of each data set randomly applied to each other.

5.5 Network's Hyper-parameters

The hyper-parameters are an important factor in the accuracy of a model. The parameters used in data augmentation are the most important because they directly affect how an image is used in training. We have changed the following elements:

- *hsv_h (hue), hsv_s (saturation) and hsv_v (value)* have values 0.005, 0.05, and 0.05, respectively. The parameterization of these elements was used to minimize drastic changes to the color of the product during training. These could change important properties of the products, making them more difficult for the network to recognize;
- *flipud*: Vertical flipping was used with a ratio of 1/10 because flipped products can be transported on the conveyors;
- *mosaic*: Mosaic was used for data augmentation, the ratio 5/10. It cuts many training images into one. Figure 7 shows an example of a mosaic.

The parameters not associated with *data augmentantion* are:

- *batch* size: 4;
- Initial learning rate: 0.01;
- Number of epochs: 50;
- Optimizer: *stochastic gradient descent* (SGD) with *momentum* 0.937 and *weighted descent*: 0.0005;
- Image size limit: 640 pixel height and 640 pixel width.



Fig. 7. Example of *data-augmentation* of type *mosaic*.

6 Results

The results of the trained models, grouped by conveyor belt, are shown in Table 3, and their metrics were explained in Sect. 3.

Table 3. Counting table

Group of conveyor belts	Precision	Recall	mAP@.5	mAP@.5:.95
Conveyor belts 1 e 2	98.07%	98.33%	99.27%	95.35%
Conveyor belt 3	97.27%	98.82%	99.02%	95.69%
Conveyor belts 4 e 5	97.70%	96.54%	98.50%	96.70%

To validate our solution, we recorded videos of the neural network counting 10 products at different moments and did the manual counting as well, so we could compare the results. A total of 4.442 bales and 1384 boxes were counted, each type representing 5 different products. Figure 4 shows the results.

To validate our solution, we recorded videos in which the neural network counted 10 products at different time points, and we also performed the counting manually to compare the results. A total of 4, 442 bales and 1, 384 boxes were counted, and each type represented 5 different products. Figure 4 shows the results.

Table 4. Counting table

Type	Manual counting	Solution counting	Accuracy rate
Bales	4,442	4,437	99.88%
Boxes	1,384	1,384	100%

The results prove that our solution is suitable for counting products in industrial lines, with an overall accuracy rate of 99.91%. Even though the accuracy rate is very high, changes in the environment can drastically reduce the accuracy rate, resulting in fewer products being counted than should be.

7 Conclusion and Future Works

In this paper, we have presented an approach to recognition, classification and counting using the `Yolov5` Deep Learning technique. Our main goal is to maximize quality control and production counting in industrial environments. The results show that our solution achieves 99.91% accuracy in product counting and classification. Moreover, these results were compared to the current manual counting system used in the industry considered in this study. This proves the feasibility of our solution in a real production environment, i.e. Semalo's industry.

We foresee the solution evolution in order to make it possible to count products with a low-power embedded system containing both software and hardware artifacts, without the need for a dedicated server to process and count the products in an even cheaper way. As another future work, we envisage the use of the current solution in other industries with characteristics different from those exposed by this paper.

Acknowledgments. This paper was only possible thanks to the help of the Semalo Indústria e Comércio de Alimentos and its workers. We thank the support of the UFMS (Universidade Federal de Mato Grosso do Sul), PET (Programa de Educação Tutorial – FNDE), FUNDECT, Finep, and Ministério da Ciência, Tecnologia, Inovações e Comunicações, funded by FNDCT. We also thank the support of the INCT of the Future Internet for Smart Cities funded by CNPq, proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and FAPESP, proc. 2014/50937-1 and 2015/24485-9.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of FUNDECT, Finep, FAPESP, CAPES and CNPq.

References

1. Bahaghighat, M., Akbari, L., Xin, Q.: A machine learning-based approach for counting blister cards within drug packages. IEEE Access **7**, 83785–83796 (2019). https://doi.org/10.1109/ACCESS.2019.2924445

2. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing (ICIP), pp. 3464–3468 (2016). https://doi.org/10.1109/ICIP.2016.7533003
3. Deac, C., Popa, C.L., Ghinea, M., Cotet, C.: Machine vision in manufacturing processes and the digital twin of manufacturing architectures, pp. 0733–0736 (2017). https://doi.org/10.2507/28th.daaam.proceedings.103
4. Frank, A.G., Dalenogare, L.S., Ayala, N.F.: Industry 4.0 technologies: implementation patterns in manufacturing companies. Int. J. Prod. Econ. **210**, 15–26 (2019). https://doi.org/10.1016/j.ijpe.2019.01.004, http://www.sciencedirect.com/science/article/pii/S0925527319300040
5. Jocher, G., et al.: ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations (2021). https://doi.org/10.5281/zenodo.4679653
6. Lee, S., Yang, C.: A real time object recognition and counting system for smart industrial camera sensor. IEEE Sens. J. **17**(8), 2516–2523 (2017)
7. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)
8. Rahimzadeh, M., Attar, A.: Introduction of a new dataset and method for detecting and counting the pistachios based on deep learning (2020)
9. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection (2015)
10. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger (2016)
11. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement (2018)
12. Shrestha, A., Mahmood, A.: Review of deep learning algorithms and architectures. IEEE Access **7**, 53040–53065 (2019)
13. Ulaszewski, M., Janowski, R., Janowski, A.: Application of computer vision to egg detection on a production line in real time. Electron. Lett. Comput. Vision Image Anal. **20**, 113–143 (2021). https://doi.org/10.5565/rev/elcvia.1390
14. Wang, S., Wan, J., Zhang, D., Li, D., Zhang, C.: Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. Comput. Netw. **101**, 158–168 (2016). https://doi.org/10.1016/j.comnet.2015.12.017, http://www.sciencedirect.com/science/article/pii/S1389128615005046