Enabling robot selective trained Deep Neural Networks for Object Detection through intelligent Infrastructure

Christian Poss, Thomas Irrenhauser, Marco Prueglmeier
Logistics Robotics, BMW Group, Munich
E-mail: christian.poss, thomas.irrenhauser, marco.prueglmeier@bmw.de

Daniel Goehring
Dahlem Center for Robotics and Machine Learning, Freie Universitaet Berlin
E-mail: daniel.goehring@fu-berlin.de

Firas Zoghlami, Vahid Salehi
Department of Applied Sciences and Mechatronic, University of Applied Sciences Munich
E-mail: firas.zoghlami, vahid.salehi_douzloo@hm.edu

Olimjon Ibragimov
School of International Business and Entrepreneurship, Steinbeis University Berlin
E-mail: olimjon.ibragimov@student.steinbeis-sibe.de

Abstract. To save costs in logistics, handling steps are going to be automated by robots in the future. Due to the complex industrial conditions prevailing there, this is only possible with a sufficient degree of intelligence in the respective systems. Despite advances in artificially intelligent algorithms, the latest neural networks reveal significant weaknesses in performance and transferability to other applications. In order to enable a holistic autonomous material flow, the paper presents an infrastructure concept, which makes it possible to identify and train the most suitable networks robot-selectively with very limited effort. Using two practical examples, the functionality of the designed algorithms for the industrial implementation of a new use case as well as the updating and improvement of an existing system is finally outlined. It will be shown that with measures such as the automated collection of training data, the AI-supported labeling process, the intuitive validation of the trained networks via a mobile application and the automated retraining of robots already integrated, a further step can be taken towards holistically automated logistics process chains.

1. Introduction

Due to the increasing complexity, caused for example by the advancing globalization as well as the trend of the increasing product individualization, the production and logistics costs increased disproportionately [1]. One way of countering this trend is to automate the entire logistics material flow [2]. The main characteristics of logistics processes in highly variant assembly are inherent adaptability and flexibility, which lead to a continuous and dynamic change in the variety of objects [3]. Classic automation such as the systems found in car body construction is therefore not possible. Much more, the robots must have a certain degree of intelligence in order to recognize different containers in different positions. Despite the progress achieved in the field of artificial intelligence, which makes the automation of such application appear realistic in principle for the first time, such algorithms still pose fundamental problems [4] [5]. Thus it can be observed that the performance of the networks increases if they are designed and trained for a single application. These two disadvantages make the holistic introduction of intelligent systems in industrial applications considerably more difficult. This publication meets this challenge. By embedding the robots in an infrastructural system, it will be possible to provide the most suitable networks for each individual robot. This supports the introduction of new systems and the necessary reactions to process changes, such as new types of logistics containers.

2. Problem Description

In order to create a common understanding of the problem to be solved, the intralogistics material flow of a highly variant assembly plant is first examined in detail in the following chapter. Subsequently, the robot solutions used for the respective processes to be carried out are presented and their current integration into the plant environment is described. Finally, an outlook is given on the situation in the near future in order to illustrate the importance and aggravation of the problem mentioned in the introduction.

2.1. Material Flow

In the intralogistical material flow, a distinction is made between production-synchronous and production-asynchronous deliveries. While in particular large components such as car seats or pre-assembled components such as the car instrument panel are delivered to the assembly line at precisely the right time, the majority of the components continue to be provided asynchronously over several process steps. This material flow is shown in Figure 1. The components are transported in containers stacked on pallets from the suppliers to the incoming goods department of the plant by truck. There the containers are separated and temporarily stored in automated storage systems. From there, they are brought to the assembly line either directly or via resequencing. As it is not possible to provide all components on the assembly line for a certain period of time due to the high number of variants and the limited available space, the latter are necessary in order to provide only the required components for the next vehicles. The containers are then transported to the assembly line by tugger trains. Arrived at the assembly line, the containers are removed from the tugger trailers and transferred to the staging racks. In return, the empties are reloaded from the staging racks onto the route train and taken to the outgoing goods department. There the containers are sorted, palletized and returned to the respective suppliers. [3]

2.2. Robots

In order to achieve a holistic automation of the material flow as described in the previous subchapter, it is necessary to automate both transport and handling steps. The statements in this publication refer exclusively to the latter, i.e. the

- depalletizing of full containers,
- the sequencing of individual parts,
- the loading and unloading of shelves (placing),
- the palletizing of empties.

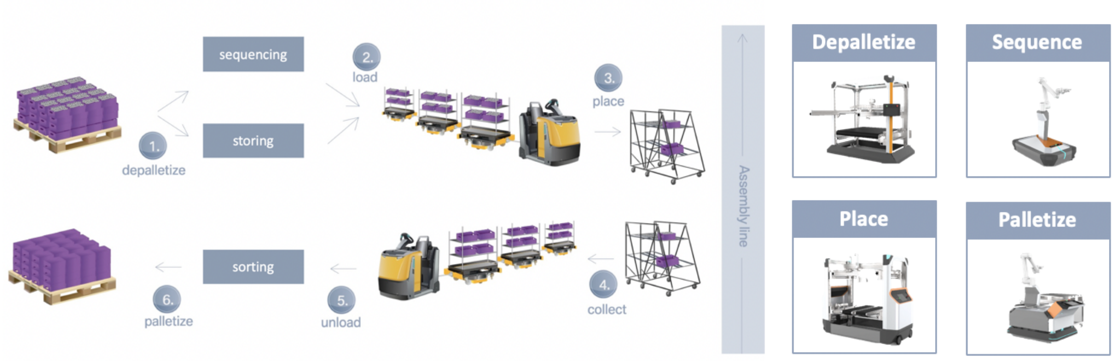

Figure 1. Logistics material flow in high variant assemblies and robot portfolio

Since these four handling steps differ significantly in terms of their requirements for the algorithms and the hardware of the robots, four specialized systems were developed. These are also shown in Figure 1.

2.3. Method Today

Due to the different general conditions for the use of robots described in the previous subchapter, even robots with the same basic task (e.g. depalletizing) must be individually adapted to their application. The main influencing factors here are various lighting situations (e.g. caused by windows, gates or infrastructural reflections), varying objects as well as soiling caused by dust or abrasion of the rubber wheels of forklifts. But the operating environment can also differ considerably between different halls and, in particular, between different plants. A comparative test has shown, for example, that when using a neural network for object detection, which was trained with images from a German plant, the use of this network in the same application case, but in an American plant, is expected to have a reduced performance by 20 % [6]. Various tests have also shown that it is possible to operate robots in this challenging process with artificial intelligence. However, the neural networks of these robots were specifically adapted to each application.

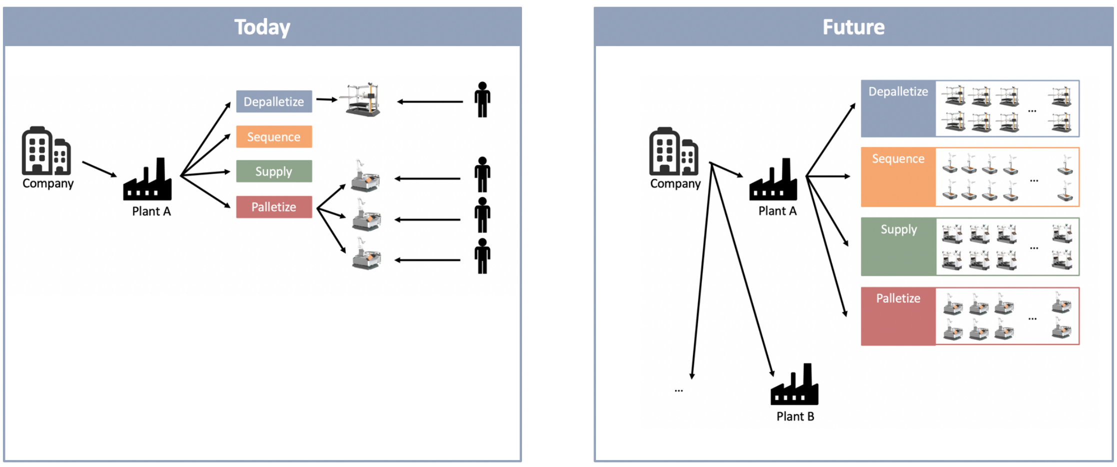

Figure 2. Todays secnario and futre vision of robot usage in the plants

The described robots were all designed following the guideline to work through the required processes as autonomously as possible. And can perform their tasks at the moment without any connection to other systems or infrastructure.

2.4. Future Vision

The successful implementation of the first robots, based on artificial intelligence in the series processes of logistics in the real plant environment, is advantageous, but involves today still a very high manual effort. A complete rollout of a medium four-digit number of robots across all plants is not possible with this approach. A compariosn between todays approach and the future scenario is shown in Figure 2.

3. Approach

In order to be able to benefit from the savings of a holistically automated process chain despite the challenges described above, an infrastructural framework is presented in the following. This enables the robot-selective individualization of neural networks through partial automation and intelligent algorithms. First, building on explanations to the current state of the art the infrastructure and the overall interaction of the individual globally distributed robots will be discussed. Subsequently, the workflow for the implementation of new applications as well as the workflow for the update of already integrated attachments is explained.

3.1. Related Work

AI infrastructure for Machine Learning Operations (AIOps) represents a technology stack required to get machine learning algorithms into production in a stable, scalable and reliable way [14]. Similar to DevOps, AIOps is a set of concepts that serve for a smooth integration of development and operations [15]. It comprises every stage of the machine-learning workflow - starting from data science tools, libraries of machine learning algorithms down to the hardware for processing and training, thus enabling data scientists, data engineers, software engineers and robotics teams to access and manage the computing resources to test, train and deploy AI algorithms.

The workflow consists of three stages:

- Stage 1: exploratory data analysis and data preparation
- Stage 2: training of machine learning algorithms on a cluster of distributed GPUs
- Stage 3: scalable deployment of trained models for inference
- Stage 4: Monitoring, management and updates

Following these steps an AI infrastructure maintains a continuous training loop based on the feedback from the monitoring stage of the workflow.

Various modern frameworks allow both data preprocessing and training of models (Stages 1 and 2). Three most popular libraries for building and training deep learning algorithms are TensorFlow, Keras and Caffe [16]. TensorFlow's biggest advantage is its use of dataflow graphs and their visualization through TensorBoard [17]. Moreover, trainings can be conducted on big clusters of GPUs.

Scalability of both training and deployment tasks is crucial, since in order to achieve an optimal result those tasks must be executed repeatedly and simultaneously. Containerization proves to be a robust solution which allows multiple instances of the same application to be run on the same machine [18]. Additionally, it resolves one of the most frequent issues that developers face: incompatibility of development environments. Encapsulated into a container, an application can easily be inserted and removed from the host system, keeping the environment clean.

Finally, creating and running containers can be automated, memories and resources can be allocated and tasks can be scheduled through orchestration systems such as Kubernetes. Its main purpose is the management of complex container-oriented distributed systems [19].

Each of these open source technologies have reached the level of maturity, and currently, are widely used as a stand-alone solution in the industry. However, there is no such system which incorporates the benefits of all above listed software for continuous integration of machine learning algorithms in the network of not only servers but also robots. Solving this deficit is the target of the following subchapters.

3.2. Infrastructure

The performance of neuronal networks can be most strongly influenced by the following levers: amount of training data, compilation of the training data, architecture of the neural network used, hyper parameters of the network and hyper parameters for the training of networks [7].

The infrastructure to be designed should therefore offer possibilities to select these influencing factors in such a way as to achieve the optimum performance for the respective robot. The final result is shown in Figure 3

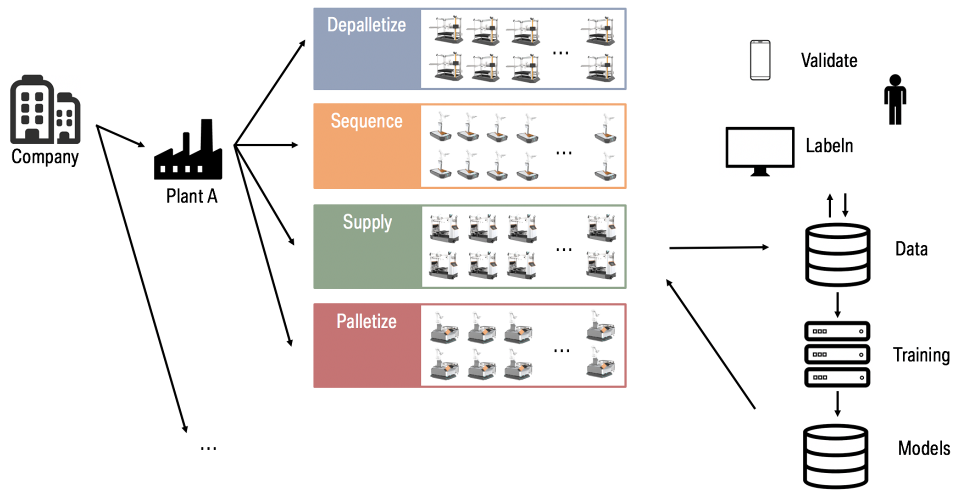

Figure 3. Suggested infrastructure for holistic robot usage in the plants

3.2.1. Training Data The manual recording of training data is associated with a high manual effort [8]. Furthermore, the training data recorded in this way do not fully reflect the real process conditions. The already integrated robots, on the other hand, make it possible to generate a large amount of new training data without manual effort. However, this is only feasible if the images can be stored centrally, enriched with metadata without long delays. The resulting pool of training data can then be used in its entirety for pre-training convolutional layer, for example. By enriching the continuously collected training images with metadata, it is also possible to evaluate different training setups. Since the robot has already recognized the objects with its existing network, the manual effort for labeling the data is no longer necessary. If, on the other hand, the object was not correctly detected by the robot, the corresponding image has to be labelled manually, but then offers large added value in order to make the network itself more robust.

3.2.2. Training Configuration and Network Selection Due to the ever more complex network architectures, a holistic optimization of the parameterization of the respective training iterations is possible almost exclusively through continuous testing [9]. To do this, the new infrastructure must be equipped with the appropriate graphics performance to be able to validate as many configuration variants as possible in as short time as possible. In order to reduce the number of necessary experiments, a database for the systematic documentation of the training history is to be integrated. This allows patterns to be recognized and the hyper parameters for new training sessions to be initialized with best practice values.

3.3. Implementation of New Use Cases

When implementing new applications, two cases can be distinguished: The implementation of a process with completely independent objects, such as the sequencing of new components as well as the integration of robots in plants with already seen objects, however, in slightly modified variants or under different general conditions.

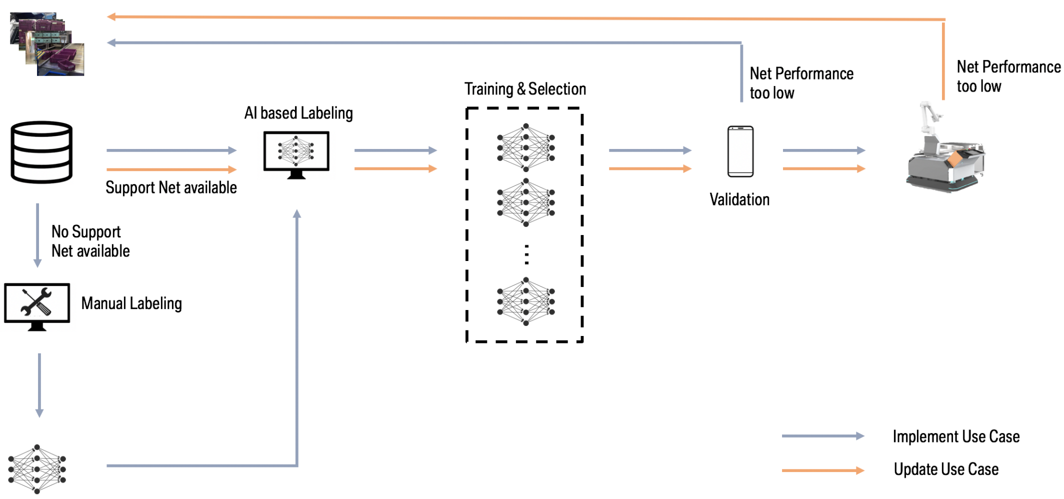

Figure 4. Methods to create and update neural networks

In a completely new application, it is first necessary to record a certain number of different training images. This initial data set has to be labelled manually. A first net is then trained with these rudimentary labelled data sets (approx. 50 images). This can then be used for labeling support. As the responsible persons only has to improve the naturally not ideal net during this procedure, the manual effort for the creation of the complete training set can be noticeably reduced.

Once the critical amount of training data has been reached, the training order can be added to the training pipeline. It trains different network architectures with different training data and parameter sets. If the best performing training configuration reaches the critical limit value for the application, the trained net can be made available in documented form. Since it would be too risky to test the network directly on the robot in series production, this is initially done using a specially designed mobile application. The trained networks can be visualized and evaluated in real time. If the required performance is not met, the network must be improved by adding new training data. By visualizing the results, the weak points and improvement potentials of the trained network can also be intuitively identified. The newly recorded training data is then uploaded and re-inserted into the training workflow.

3.4. Automated Retraining

The application case of automated retraining can occur if the process conditions of an already productive robot have changed significantly. Such a drop in performance can be quickly determined on the basis of the continuously monitored robot parameters. If this scenario occurs, further training data is recorded from the failed object detection processes. Although this leads to a temporary process-related deterioration, it allows the recording of relevant training data for a new training of the network without manual intervention. The images of erroneous predictions are, as already mentioned, specially marked and manually labelled. As soon as this is completed, the network is re-trained with the new training data and the old training parameters and made available for the robot.

Another chance is the non-error driven retraining. As the literature shows, more training data can lead to better results [9] [10]. So it is also conceivable here to train each net anew as soon as a critical amount of new training data has been collected. It should be noted, however, that the classes to be recognized are not equally distributed. For example, there are strong differences particularly in the containers with regard to their percentage frequencies in the material flow. These inequalities would be exacerbated by a continuous increase in all training data. Therefore, a further analysis is necessary in order to achieve an overall optimum across all classes either through augmented training data or individually selected training data.

4. Results

This chapter shows the exact data of the workflows described in the previous section and validates them using two implementation examples. For this purpose, the training of a new network for the assembly line feeding robot for a certain assembly line section in the BMW plant in Leipzig is first considered. A second example is the depalletizing robot. This was tested with an initially manually trained network in series production. The potential improvements are explained in the second part on the basis of the stored images and the experience gained from the initial training.

Overall, it should be noted in this chapter that the infrastructure designed and explained has not yet been fully implemented to this extent. One reason for this is, for example, the lack of WLAN availability in some factories. The algorithms integrated in the pipeline were therefore tested and validated in manually generated scenarios. If they function with manually provided data, they can also be transferred to the automated workflow. The here presented results are therefore completly adaptable to the serial solution.

4.1. New Training: PlaceBot New

Initially, 300 training pictures were taken to integrate the conveyor robot into the plant. We took care that these pictures reflect the pictures to be expected later as good as possible. Of the 300 training images, 50 were initially labelled manually. These 50 raw images were then used to train a first neural network for object detection, which could then be used for ai-supported labelling. Yolo was used for this, as this was shown in the tests to be best suited for low training data [11]. Thanks to data augmentation, the performance of the network is sufficiently high to significantly reduce manual effort by improving predictions instead of drawing in all class objects. It is not suitable for productive use. After all training pictures were labelled, further neuronal nets were trained. Both Faster RCNNs [12] and SSDs [13] were trained with different parameterizations (training parameters, network parameters) and a maximum mAP of 59 % was achieved. During the analysis of the network with the mobile application in the real process environment, a certain container class was identified as a weak point. With the training data extended by 100 pictures of this class, the already mentioned training rounds were repeated. By this an mAP of 73 % was achieved. This accuracy could be increased to 89 % by applying the maximum possible data augmentation. Through the use of cropping, spelling, noise, varying brightness and varying saturation, the neural networks could be re-trained with 300,000 training images. The 89 % achieved are sufficiently accurate, since the object detection in the robot is supplemented by a validation module which, by using environmental knowledge, prevents false detections and, in turn, can deduce unrecognized objects.

4.2. Retraining: SplitBot with New Boxes

A generic network was initially used for the first attempted application of the depalletizing robot. This means that the training data did not completely reflect the later process conditions, but contained general use cases from logistics. A net trained in this way achieved an accuracy of 64 % during the first factory tests in Leipzig. Based on this experience, the training data were reassembled and the network was trained with a data set consisting of 50 generic and 250 application-specific images, also using the maximum data augmentation already discussed. This specialisation enabled the mAP to be increased to 86 % at the Leipzig plant. After the tests in Leipzig, this robot was transported to the Dingolfing plant for further tests. In addition to the changed environmental conditions, a new container type was used there. After the first detected errors, the training data was expanded accordingly and a new training session was initiated. This means that the mAP was again raised to 91 %. The robot could thus continue to be used under the intended process conditions without the great expense of a completely re-trained network architecture.

5. Conclusion

This paper presents an infrastructure concept that partially automates the individual training of neural networks and increases the overall robot-selective performance. Furthermore, the functionality of the designed algorithms was tested and confirmed by means of two real examples. With measures such as the automated collection of training data, the ai-supported labeling process, the intuitive validation of the trained networks via a mobile application and the automated retraining of already integrated robots, a further step can be taken in the direction of holistically automated logistics process chains, as shown by the example of two applications for depalletizing and assembly line provision.

References

[1] K. Permenter, "Cost and Complexity: Top Challenges of Today's Chief Supply Chain Officer", Aberdeen Group, January 2012.

[2] A. Stringflow, "How to Reduce Logistics Costs: 19 Experts Reveal Ways Organizations Can Cut Their Logistics Transportation and Carry Costs", New Technology, June 2018.

[3] D. Arnold et al., "Handbuch Logistik", Springer-Verlag Berlin Heidelberg, 978-3-540-72929-7, 2008.

[4] LeCun, Y., Bengio, Y., Hinton, G. Deep learning. nature, 521(7553), 436. 2015.

[5] Schmidhuber, Juergen. "Deep learning in neural networks: An overview." Neural networks 61, 85-117. 2015

[6] C. Poss et al., "Application of Open-Source Deep Neural Networks for Object Detection in Industrial Environments", 17th IEEE International Conference on Machine Learning and Applications, 2018.

[7] Goodfellow, Ian, et al. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

[8] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[9] Rosebrock. "Deep Learning for Computer Vision - Image Net Bundle", 2018.

[10] Mayer, Nikolaus, et al. "What makes good synthetic training data for learning disparity and optical flow estimation?." International Journal of Computer Vision (2018): 1-19.

[11] Jeong, Hyeok-June, Kyeong-Sik Park, and Young-Guk Ha. "Image Preprocessing for Efficient Training of YOLO Deep Learning Networks." Big Data and Smart Computing (BigComp), 2018 IEEE International Conference on. IEEE, 2018.

[12] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[13] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[14] C. Nicholson, https://blog.skymind.ai/ai-infrastructure-machine-learning-operations-mlops

[15] M. de Bayser, L. G. Azevedo, R. Cerqueira. "ResearchOps: The case for DevOps in scientific applications", IFIP/IEEE International Symposium on Integrated Network Management, 2015
[16] J. Zacharias, M. Barz, D. Sonntag. "A Survey on Deep Learning Toolkits and Libraries for Intelligent User Interfaces", arXiv.org, 2018.
[17] M. Abadi, et al. "TensorFlow: A System for Large-Scale Machine Learning", Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, 2016.
[18] J. Turnbull. "The Docker Book: Containerization is the New Virtualization", 2014.
[19] B. Burns, et al. "Borg, Omega, and Kubernetes", Queue - Containers, 2016.