Methods

Christoph Wree and Rando Raßmann*

Methodology for optimizing convolutional neural networks for fast production processes

Methode zur Optimierung von Convolutional Neural Networks für schnelle Produktionsprozesse

https://doi.org/10.1515/auto-2024-0155
Received November 8, 2024; accepted May 7, 2025

Abstract: An individualized production is needed to manufacture products with a batch size of up to 1 at low costs. To ensure that the individual steps in the production process are executed for each individual product, the products have to be classified beforehand. For complex problems, machine vision can be used in conjunction with machine learning (ML) to classify the individual components. Here, models of Convolutional Neural Networks (CNNs) in particular achieve high classification accuracies. However, they require high computational cost, which makes it more challenging to execute CNNs in real-time within a PLC runtime environment and synchronize them with motion control tasks. In this paper, a methodology is presented using a production-oriented application example to minimize the inference time of a CNN models for image classification while maximizing the classification accuracy. The presented methodology demonstrates how computationally cost intensive CNNs can be optimized for real-time image recognition in coordination with machine and motion control tasks. The execution times as well as the achieved accuracies of the CNNs are measured. The results show that the CNNs trained on a synthetic CAD dataset are able to reliably classify individual products with an accuracy of 100 % in less than 185 µs (with an image size of $39 \times 26 \times 1$ pixel).

Keywords: individualized production; convolutional neural networks; hyper-parameter optimization

Zusammenfassung: Eine individualisierte Produktion wird benötigt, um Produkte mit einer Losgröße von bis zu 1 zu geringen Kosten herzustellen zu können. Damit die jeweiligen Schritte im Produktionsprozess ausgeführt werden, müssen die Produkte zuvor entsprechend klassifiziert werden. Bei komplexen Problemen kann maschinelle Bildverarbeitung in Verbindung mit maschinellem Lernen, insbesondere durch die Verwendung von Convolutional Neural Networks (CNNs), hohe Genauigkeiten erzielen. CNNs sind jedoch sehr rechenintensiv, was die Integration in eine Echtzeitumgebung innerhalb einer SPS erschwert. In diesem Beitrag wird eine Methode zur Minimierung der Inferenzzeit von CNN-Modellen für die Bildklassifizierung bei gleichzeitiger Maximierung der Genauigkeit vorgestellt. Die Ergebnisse zeigen, dass CNNs, die auf einem synthetischen CAD-Datensatz trainiert wurden, individualisierte Produkte mit 100 % Genauigkeit in weniger als 185 µs klassifizieren können (bei einer Bildgröße von $39 \times 26 \times 1$ Pixel).

Schlagwörter: Individualisierte Produktion; Convolutional Neural Networks; Hyperparameter-Optimierung

1 Introduction

To cope with the diverse customers' needs in a competitive market, manufactures have to increase the flexibility of their production lines. Therefore, currently, the trend in manufacturing systems is towards an individualized production. Individualized production refers to manufacturing systems that can produce products for a single customer up to a batch size of 1 at low cost [1]–[3]. Different sub-variants of the same product can be manufactured on the same production system. To ensure that each variant's individual manufacturing steps are executed, they have to be classified in advance. Figure 1 illustrates a schematic representation of such an individual production system, where the individual products must be classified according to the sequence of the customer's order.

Possible methods for classifying individual products or workpieces can be based on reading RFID tags [3] or barcodes [4]. However, it is difficult to attach RFID tags or

***Corresponding author: Rando Raßmann**, Department of Computer Science and Electrical Engineering, University of Applied Sciences Kiel, Grenzstr. 5, 24149 Kiel, Germany, E-mail: rando.rassmann@fh-kiel.de
Christoph Wree, Department of Computer Science and Electrical Engineering, University of Applied Sciences Kiel, Grenzstr. 5, 24149 Kiel, Germany, E-mail: christoph.wree@fh-kiel.de

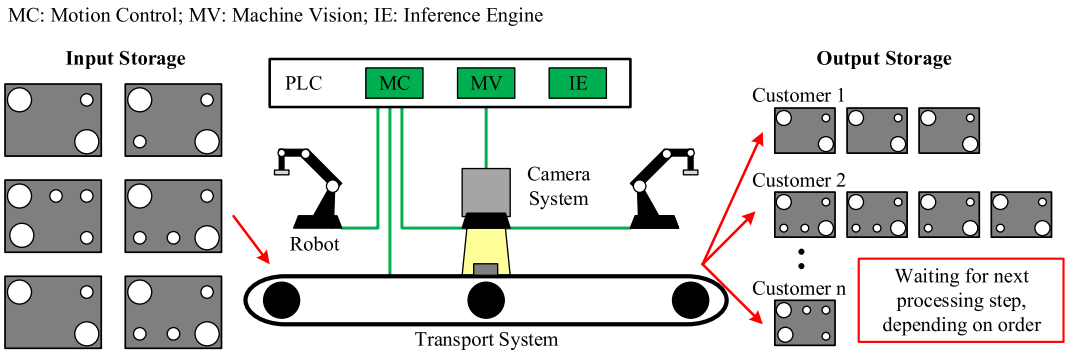

Figure 1: Manufactured products are collected in an input storage. From the input storage the individual products are placed onto a transport system. MV and the IE are used to classify the products. Depending on the classification, the PLC initiates the next production step, e.g. sorting the ordered products for packaging or routing the products to the next machining station.

barcodes to the product during the manufacturing process. In addition, such identifiers could be damaged or occluded during a production step. Another approach is to use image analysis methods (machine vision) for classification [5]. In recent years, machine learning (ML) algorithms have been successfully applied in the field of computer vision to solve complex classification problems [2], [6], [7]. ML based vision systems can reduce the programming effort and improve the efficiency of a system [2]. For conventional machine vision the image processing system can be implemented on an external computer [5], on an intelligent camera [5] or within a PLC runtime environment hosted on an industrial PC (soft-PLC) [8].

One key question for using ML in an industrial environment is, where the inference engine is executed. The inference engine can be implemented in a cloud (option 1) with highest computational resources [9] or on an edge server (option 2) with high computational resources [2], [10], [11]. It is also possible to execute the ML models on a separate controller (option 3), which interacts with the PLC [12] or to execute the ML model directly within the PLC controller (option 4), within the PLC runtime [8]. For the latter two possibilities the computational resources are limited. The integration of CNN models for manufacturing systems and real-time applications has been discussed in various investigations [10], [13]–[15]. All four options have different advantages and disadvantages, which are summarized in Table 1.

In order to achieve higher accuracies, there is a common trend towards deeper and more complex model architectures based on CNNs for computer vision solutions [6], [16]. Modern CNNs models for image classification are able to classify complex data with hundreds of different objects from different perspectives and with varying backgrounds. This results in higher computational cost for the state-of-the-art models for both training and inference. In dynamic processes with high throughput rates and in conjunction with motion control, there is only a short time window between the acquisition of the sensor signal for classification and the execution of the following actuator command based on the classification result. It is crucial to obtain the classification results in the shortest possible time, so that there is sufficient time for the calculation of the subsequent positioning command for the actuators. In terms of using a cloud or edge computing, a long latency in the communication could be problematic. In addition, the unavailability of

Table 1: Comparison of different integration strategies for inference engines for individual production systems.

Requirement	Cloud (1)	Edge-computing (2)	External controller (3)	PLC (4)
Computational resources	++	+	−	−
Less/no additional interfaces	−−	−	+	++
Fast response time[a]	−−	−	+	++
Reliability	−	−	+	+
Less complexity for IT	−	−	+	+
Simple synchronization to other tasks	−	−	−	++
Reduced network traffic load	−−	−	+	++

[a]for applications with response times in a microsecond range. Requirements satisfied: ++, excellent; +, well; −, acceptable; −−, insufficient.

the service could also be a problem if the inference engine is located in a cloud or on an edge device [9].

Different studies have investigated how models of CNNs for image classification can improve industrial processes. In [2] a CNN-based sorting system for a flexible manufacturing system is proposed. The applications of automated systems for quality inspection with CNNs are presented in [10]. All these approaches are using a camera system which sends the image to an edge-server where the inference engine is located. Subsequently the results are transmitted to a PLC to set the following actuator command, based on the results. Both the inference of the model and inference pipeline must be optimized to meet the real-time constraints. In [17] an architecture is presented that allows the use of a GPU within the real-time environment of a soft-PLC.

However, a dedicated GPU is expensive. It is not always possible to add a GPU into an existing industrial PC (IPC). Moreover, within a production system the computational power of a GPU is not always needed to solve a problem applying ML. If the image processing and the inference of the ML models is implemented on the same controller as the machine control and the motion control, the different tasks can be efficiently synchronized with each other, and latency times as well as additional interfaces can be reduced. The integration of all tasks into one central system avoids also separate subsystems, guarantees full transparency overall processing steps and reduces the complexity to implement IT-security.

Due to the mentioned reasons, for dynamic processes in conjunction with motion control und short response times it is preferable to integrate the inference engine direct into a soft-PLC, instead off using an external controller, an edge-device or a cloud. Integrating ML based models into the real-time environment of a PLC or an external controller is challenging because the developed models need to have a deterministic runtime behavior and the computational resources are limited. Recent works [8], [12] have demonstrated that it is possible to integrate Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) into the real-time environment of a (soft-)PLC for simple classification problems. However the computational cost for CNN models are too high with inference times of 350 ms [12] and 41 ms [18], in order to use them in dynamic applications in conjunction with motion control with typical PLC task cycle times of 1–2 ms [13].

To execute the CNNs models in less than 1 ms directly on a soft-PLC, the computational cost of the model architectures has to be reduced, while maintaining a high accuracy. The field of application of ML models for image classification for manufacturing applications differs substantially from the field of application of modern ML image classification models such as YOLO. In manufacturing systems all processing steps are well known and controlled. There are no or less interfering effects or changes in the image acquisition process. In addition, the models do not need to be able to identify so many different and complex objects. Thus, the conditions of the industrial environment simplify the problem to be solved. This can be utilized to reduced computational cost of the models while maintaining a high accuracy.

In this paper an approach for integrating CNN models for image classification into a soft-PLC in conjunction with motion control for an individualized production is presented. The applied methodology for minimizing the execution time and maximizing the accuracy, through hyper-parameter optimization is explained. The CNN optimization is based on the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [19]. In order to train the models without the availability of a dataset from the production system, this work investigates whether a synthetic dataset can be used for the training. The final model selection is built on the required floating-point operations (FLOPs) and the achieved accuracy.

The rest of the paper is organized as follows. In Section 2, the production system of the applied use case and the used methodology for optimizing the model architectures are explained. The results of the optimization process, as well as the required execution times and the achieved accuracies on the production system are presented in Section 3. Section 4 discusses the results, followed by Section 5, which concludes the study.

2 Methodology

2.1 Methodology for determining an appropriate model architecture

The computational cost as well as the accuracy of the models to classify the products are substantially influenced by the hyper-parameters of the model architectures (e.g. the input image size as well as the numbers, dimensions, and types of the hidden layers). The hyper-parameters of the training itself, such as learning rate or batch size, also affect the final model accuracy. A direct measurement of the required execution time of the CNNs during the training is not possible in the user mode of a Windows operating system. In addition, in the later production system, the models are executed in a single task on a CPU of a soft-SPS with less computer resources, while during the training process the models are executed on an GPU on a workstation. In order to estimate

the required execution time, the number of floating-point operations can be used as an indirect metric, to compare the different model architectures among each other [16]. However, it should be noted that there is no linear relationship between the number of FLOPs and the required execution time.

To identify every product correctly, an accuracy of 100 % is required. Thus, the question arises whether the computational cost of the network architecture can be reduced while maintaining an accuracy of 100 %. Figure 2 plots the theoretical accuracy vs. the computational cost for a classification problem which is be optimized by maximizing the accuracy and minimizing the execution time.

Model 1 represents the simplest architecture with only one input neuron that is directly connected to the output vector without any hidden layers. Due to the simple architecture, the computational cost for the model are low but the model is not able to learn, so it is not able to achieve a high accuracy. By e.g. increasing the size of the input layer or by adding hidden layers, more trainable parameters are available in the model. The computational cost of the model increases and thus, the accuracy of the model may also be raised. So, with increasing the computational cost, models with higher accuracies can be achieved. Model 2 has the lowest computational cost for a certain accuracy. Model 3 has the lowest computational cost to achieve an accuracy of 100 %. This model is the model with the optimal model architecture and has to be identified. Models 4, 5, 7 and 8 have higher computational cost but do not achieve high accuracies due to the selected hyper-parameters. Models 6 and 9 achieve an accuracy of 100 % but have higher computational cost, which leads to a higher execution time.

2.2 Description of the production system demonstrator used for the experiments

In this practical application example, cuboids representing hydraulic blocks shall be classified by using CNNs instead of machine vision. The hydraulic blocks differ in number, position, depth, diameter and type of their holes (see input storage in Figure 3). It is evident that solving this kind of problem with conventional machine vision techniques would be a time-consuming process and requires expert knowledge.

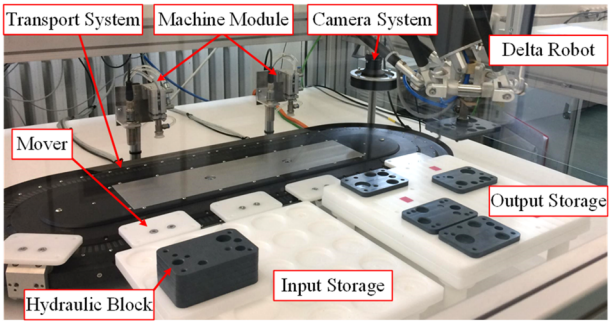

Figure 3: Demonstrating an individualized production using ML for classifying products.

To measure the execution times of the trained models within an industrial controller, the models are integrated into a soft-PLC of a production system demonstrator. The production system is shown in Figure 3 and is structured as follows: Individual products are located in an input storage. In this application example, the hydraulic blocks can be stacked in random order and can be rotated 180° around the *z*-axis. A delta robot picks the individual products (hydraulic blocks) from the input storage and places them on the transport system. The transport system moves the individual products to different stations and allows to determine the position of the products at any time. The production system consists of three stations in total, two machine modules and a camera system. The camera system is operated with an LED ring light to reduce the influence of changing light conditions. The image is captured while the product is travelling at 4 m/s on the transport system. The product only requires 25 ms to reach the robot's pick position. From there, depending on the classification result, the robot

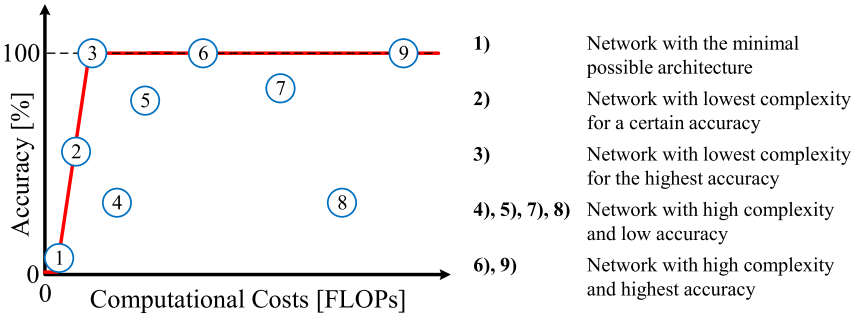

Figure 2: Accuracy (%) vs. computational cost (FLOPs) for different model architectures for solving an image classification problem.

places the product at the individual storage location.The control of the machine, the control of the transport system, the control of the delta robot, the image acquisition and preprocessing, and the inference of the CNNs are executed on different tasks, but in the same runtime environment. The production system is operated with a TwinCAT runtime running on a conventional Beckhoff Industrial PC (IPC) C6920 with an Intel® i7-4700EQ 2.4 GHz processor with four cores. A detailed description of the production system's structure can be found in [8].

2.3 Dataset generation to classify the hydraulic blocks

For solving classification problems based on ML, a dataset is needed to develop such a model, and the data quantity has to match the problem complexity. For image classification problems, the dataset has to contain data tuples of input images and corresponding class labels. Among other things, images already captured from the production system may be used for this purpose. However, this method could not be applied to new products since this data is not yet available. In this case, the development of the models can only start after the production system has been put into operation. In addition, recording and labeling the data is a time-consuming and resource-intensive process.

To solve this problem, a synthetic dataset is generated using the CAD models of the individual products. The process for generating the data is carried out as follows: A total of 10 different 3D models of the hydraulic blocks are generated. Next, the top view of each hydraulic block is created, so that the initial dataset consists of 10 images, one image per class (see Figure 4a). By applying data augmentation methods, the dataset size is increased. The generation of the synthetic images is divided into two steps. First, 20 images per class are generated by randomly rotating the original image by $\pm 3°$ and randomly shifting it by ± 50 pixels on the x- and *y*-axes. Second, 1000 images per class are generated by randomly changing the 20 images in brightness, contrast, and elastic deformation (see Figure 4b). The final dataset for developing the CNNs consists of 10,000 grayscale images in total, each with a shape of 300×200 pixels (1000 images per class). This dataset is split up into two subsets, one for training and one for validation, with 7000 and 3000 samples, respectively. A synthetic dataset is beneficial, because the model development can start once the CAD drafts of the individual products are available. There is no need to wait for hundreds of prototypes of each product to start respective development process.

In addition, a holdout dataset consisting of 200 images in total is generated to assess whether the model can generalize well on unseen data (see Figure 4c). For this purpose, a prototype for each of the 10 individual products is generated. For example, this can be done by using rapid prototyping technologies such as 3D printing. Subsequently, 20 images with varying lighting conditions are taken of each individual product with a smart phone camera. The images for the test dataset (Figure 4d) used for the final evaluation of the models is generated with the camera system of the production system.

2.4 Model architecture optimization process

For developing models with high performance, the selection of appropriate hyper-parameters is one of the major challenges. Common search strategies for hyper-parameter optimization are random search, Bayesian optimization, evolutionary methods e.g. genetic algorithms [20]. Traditional optimization techniques such as gradient based methods are not suitable to solve this problem because the hyper-parameters in the search space are discrete, and the objective function is not differentiable.

For this practical example the optimization objective of the hyper-parameter optimization is to find an architecture $\vec{s} \in \boldsymbol{S}$ that minimizes the model's FLOPs while maximizing the accuracy, where the vector $\vec{s}$ represent the decision variables for the hyper-parameters within a defined search space $\boldsymbol{S}$.

Genetic algorithms are well-suited for addressing the formulated multi-objective optimization problem, where the objectives are in conflict with each other [20]. They iteratively generate new populations using genetic operators

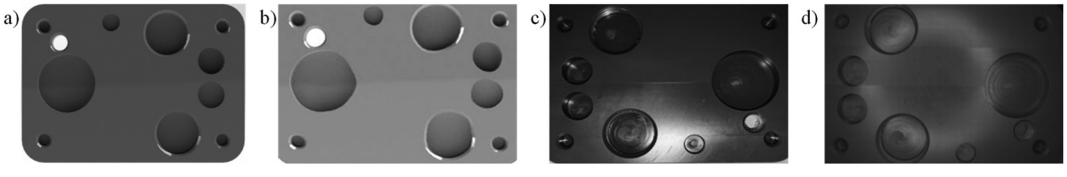

Figure 4: Comparison of a) an image of a hydraulic block CAD model to generate a synthetic development dataset, b) a synthetically generated image of the hydraulic block in the development dataset, c) image of the manufactured hydraulic block taken with a smartphone camera for the holdout dataset and d) an image of the hydraulic block to be classified taken by the camera system of the production system (test dataset).

Table 2: Search space $\boldsymbol{S}$ for the hyper-parameter optimisation based on the NSGA-II.

Hyper-parameter	Search space $\boldsymbol{S}$
Input-image size	30×20 – 300×200
Batch normalization	True/False
Convolutional-layer:	
Number of layers	1–3
Kernel size	1–5
Number of filters	1–64
Activation function	Relu, sigmoid, tanh
Pooling-layer:	
Pooling	True/False
Pooling size	2–10
Dense-layer:	
Number of layers	0–2
Activation function	Relu, sigmoid, tanh
Number of neurons	16–512
Dropout	True/False
Dropout rate	0–0.8

such as selection, crossover, and mutation until a predefined termination criterion is met. Since there is no single best solution, instead trade-offs between accuracy and inference time must be identified. This trade-off can be achieved by identifying non-dominated solutions within the objective space $\boldsymbol{Y}$, known as the Pareto-optimal set. A solution vector $\vec{y}_1$ dominates a solution vector $\vec{y}_2$ in $\boldsymbol{Y}$ if there is no other solution that can improve at least one of the objectives without degradation any other objective. To find a model architecture the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [19] is particularly suitable. One of its key features is its ability to return a set of Pareto-optimal solutions, facilitating informed decision-making for the model selection process.

The total hyper-parameter optimization process is implemented in a Python script. For the training of the models the framework Keras and for the implementation of the NSGA-II the Python module pymoo [21] are used. The defined search space $\boldsymbol{S}$ for the hyper-parameter optimization to determine an appropriate model architecture is shown in Table 2.

For scaling the images to the required input data size, a bilinear interpolation is applied. Example images for the minimum input data for the three datasets are shown in Figure 5.

Figure 6 illustrates the applied optimization process for identifying model architectures with a high accuracy and less required FLOPs. Since the size of the input images can be changed, the 300×200 pixels 8 bit grayscale images are reduced to the new pixel size before each training process. For the training of the models the synthetic development dataset is split into a training dataset and a validation dataset. After the training of each model architecture, the models are evaluated by determining the achieved accuracy on the holdout dataset and the number of required FLOPs.

2.5 Integration of the CNNs into the production system

To integrate the models into the runtime environment of the soft-PLC of the aforementioned production system, a compiler is required that converts the trained models into machine-readable code. If a TwinCAT runtime is used, this can be achieved by using the MATLAB Coder[1] in conjunction with the TE1401 TwinCAT Target for MATLAB.[2] The CNNs trained are imported into MATLAB (R2023a) and called by MATLAB functions using the “Predict” method. Next, the MATLAB Coder is used together with the TE1401 TwinCAT Target for MATLAB to convert the MATLAB functions into a PLC library. Finally, the PLC library can be used in TwinCAT Engineering to integrate the generated function blocks into the PLC project.

3 Optimization results

3.1 Results of the optimization process

In total 2400 models over 60 generations are examined with the help of the NSGA-II. Each model is trained on an Nvidia GeForce RTX 4090 GPU for 100 epochs using Adam as a training optimizer. The entire optimization process has taken 23 h. Figure 7a illustrates the required number of FLOPs versus the accuracy for all investigated CNNs, obtained during the NSGA-II optimization process on the holdout dataset.

In Figure 7a the required FLOPs and the achieved accuracy of all investigated models on the holdout dataset are shown. The majority of suitable model architectures are in a range between 0.01 MFLOPs and 100 MFLOPs. It is

1 Release Notes for MATLAB Coder–MATLAB & Simulink–MathWorks Deutschland. [Online]. Available: https://de.mathworks.com/help/coder/release-notes.html (accessed: Oct. 13 2024).

2 Beckhoff Automation GmbH &amp and Co. KG, Hülshorstweg 20 33,415 Verl, TwinCAT 3 Target for MATLAB®. [Online]. Available: https://www.beckhoff.com/de-de/produkte/automation/twincat/texxxx-twincat-3-engineering/te1401.html (accessed: Oct. 13 2024).

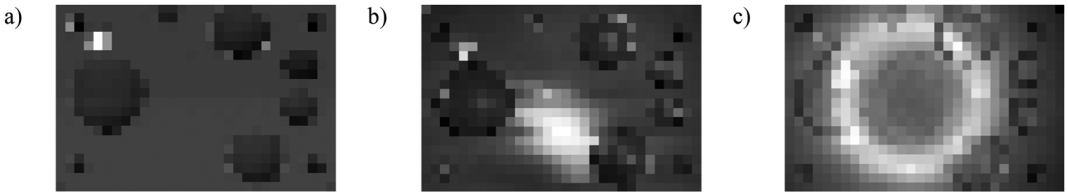

Figure 5: Example images reduced to the minimum size of 30 × 20 pixel of: a) Development dataset, b) holdout dataset, c) test dataset.

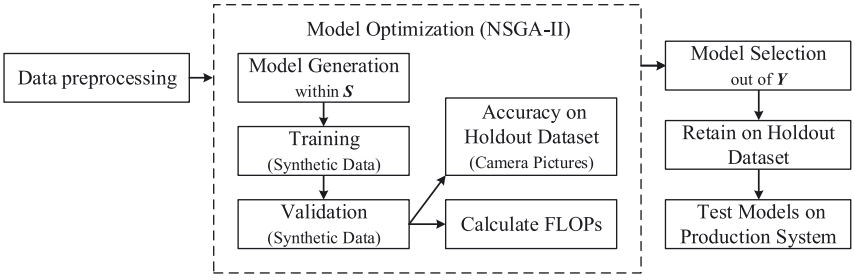

Figure 6: Optimization process for identifying a model architecture with a high accuracy and less FLOPs.

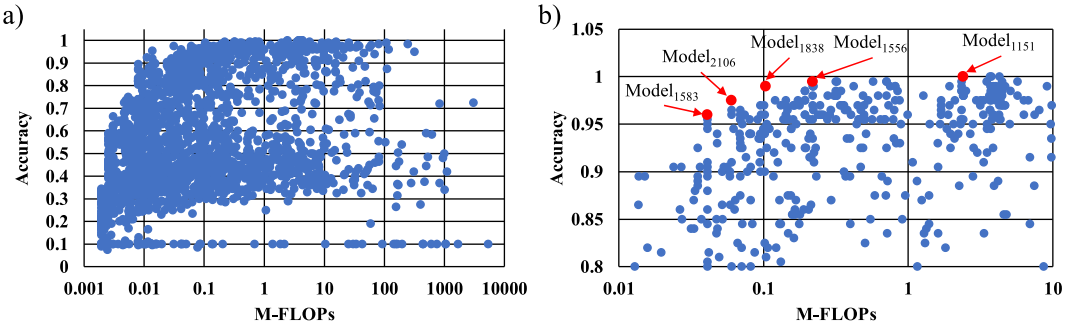

Figure 7: Accuracy vs. FLOPs on the holdout dataset a) all investigated CNNs, b) CNNs with 100 % accuracy on the CAD validation dataset.

also evident that models with more than 0.1 MFLOPs are able to achieve accuracies greater than 99 %. Below 1 MFLOPs the accuracy of the models starts to decrease slightly. With further reduction of the computational cost (<0.1 MFLOPs), the loss in performance becomes even more visible.

3.2 Methodology for the model selection

After the optimization process, five CNNs are selected to implement them on the production system to investigate the execution time and the accuracy on the production system. To select the models, the first step is to reduce the number of potential models by filtering out all models that do not achieve 100 % accuracy in the CAD validation data set (see Figure 7b). Second, the first five models that dominate in terms of accuracy and number of FLOPs are selected (red points in Figure 7b). The architectures of the selected models are presented in Figure 8. Table 3 summarizes the achieved accuracies.

3.3 Investigation of model performances

To investigate the execution times and the accuracies on the production system, the five selected models are trained using the CAD-Dataset. Subsequently, the pretrained models are retrained for a few epochs (<50) on the holdout dataset (compare Figure 6). Finally, the models are integrated into the soft PLC with the described methodology in Section 2.5. After the integration of the selected CNNs, the accuracy and execution time of the CNNs are investigated (compare Table 3). The results demonstrate that the CNNs achieve high accuracies on both the holdout and test datasets.

$Model_{1583}$ exhibits the fastest execution time of about 185 µs The slowest model is $Model_{1151}$ with an average execution time of 10.6 ms.

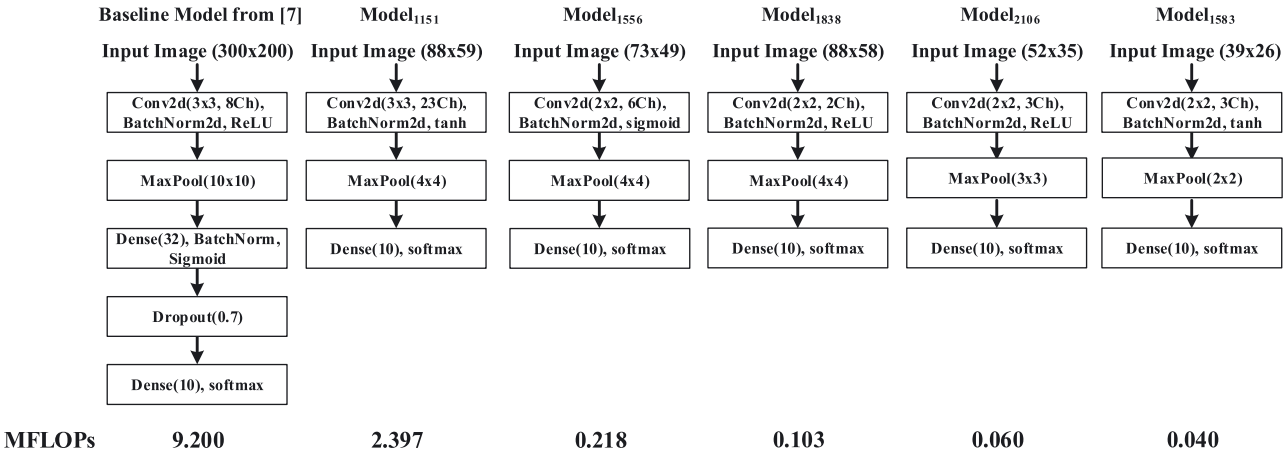

Figure 8: Model architecture and the number of FLOPs of the selected models.

Table 3: Accuracies (%) on the used training, validation, holdout and test datasets, that has been used by the NSGA-II optimization algorithm, and the final test dataset as well as execution times (µs) of the selected models.

Model	Accuracy (%)				Execution time (µs)
	Training	Validation	Holdout	Test	
$Model_{1151}$	100	100	100	100	10,640
$Model_{1556}$	100	100	99.5	100	1278
$Model_{1838}$	100	100	99.0	100	433
$Model_{2106}$	100	100	97.5	100	237
$Model_{1583}$	99.98	100	96.0	100	185

4 Discussion

The results in Table 3 show that the execution times can be substantially reduced while maintaining a classification performance of 100 % accuracy on the test dataset. These results can be explained, because the industrial environment is well controlled which means that the classification process is less affected by changes. Based on the evaluation of the results from Table 3, the following statements can be made:

- Synthetically generated data from the CAD design can be used to train models of neural networks that achieves high accuracies on real data when embedded into a production system. This is advantageous because the models can be developed before commissioning the production system.
- By additionally considering the required computational cost in the hyper-parameter optimization, model architectures with minimal sizes can be identified that solve the classification problem.
- The knowledge and control of the entire process can be exploited to simplify both the problem to be solved and the model architecture.
- The optimization process results in a model architecture with a sub-millisecond execution time of a soft PLC. The fastest model is 200 times faster than in [18]. Thus, the execution time is substantially lower than the PLC task cycle time of fast motion control applications (1–2 ms). This enables the deployment of CNNs for image classification in conjunction with motion control applications with real-time requirements.

5 Conclusions

This paper investigates the optimization of CNN architectures for image classification in an industrial context. The hyper-parameters of the CNN models are optimized for their accuracy and the FLOPs by using the evolutionary algorithm NSGA-II. In order to train the models without having access to a dataset of the real products, a synthetic dataset is created and used for training and optimization. The results of

the presented methodology demonstrate that the proposed optimization significantly improves both the accuracy and execution time of the CNNs on the demonstrator production system. The execution times of the investigated models can be reduced by a factor 200 from 41 ms to 185 µs, while maintaining an accuracy of 100 %. The presented approach can be used to integrate machine learning models directly on a soft-PLC. This is beneficial because, firstly, no additional hardware (external controller) is required and, secondly, there are no latency times or additional interfaces (cloud and edge computing).

Research ethics: Not applicable.
Informed consent: Not applicable.
Author contribution: All authors have accepted responsibility for the entire content of this manuscript and approved its submission.
Use of Large Language Models, AI and Machine Learning Tools: None declared.
Conflict of interest: The author states no conflict of interest.
Research funding: None declared.
Data availability: Not applicable.

References

[1] Y. Koren, "The local factory of the future for producing individualized products," *The Bridge*, vol. 2021, no. 51, 2021.
[2] Y. Wang, K. Hong, J. Zou, T. Peng, and H. Yang, "A CNN-based visual sorting system with cloud-edge computing for flexible manufacturing systems," *IEEE Trans. Ind. Inf.*, vol. 16, no. 7, pp. 4726—4735, 2020.
[3] R. Y. Zhong, Q. Dai, T. Qu, G. Hu, and G. Q. Huang, "RFID-enabled real-time manufacturing execution system for mass-customization production," *Robot. Comput.-Integr. Manuf.*, vol. 29, no. 2, pp. 283—292, 2013.
[4] R.-S. Chen, K. Y. Lu, S. C. W. Yu, H. W. Tzeng, and C. Chang, "A case study in the design of BTO/CTO shop floor control system," *Inf. Management*, vol. 41, no. 1, pp. 25—37, 2003.
[5] A. Hornberg, *Handbook of Machine and Computer Vision*, 2nd ed., Berlin, Wiley-VCH, 2017.
[6] S. T. Krishna, *et al*., "Deep learning and transfer learning approaches for image classification," *Int. J. Recent Technol. Eng.*, vol. 7, no. 5S4, pp. 427—432, 2019.
[7] N. O'Mahony, *et al*., "Deep learning vs. Traditional computer vision," in *Advances in Intelligent Systems and Computing, vol. 943, Advances in computer vision: Proceedings of the 2019 Computer Vision Conference (CVC)*, K. Arai, and S. Kapoor, Eds., Cham, Springer, 2020, pp. 128—144.
[8] R. Raßmann, *et al*., "Investigations on real-time image recognition with convolutional neural networks on industrial controllers," in *Studies in Computational Intelligence, Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future: Proceedings of SOHOMA 2022*, 1st ed., T. Borangiu, D. Trentesaux, and P. Leitão, Eds., Cham, Springer, 2023, pp. 380—391.
[9] I. Rodriguez-Conde, C. Campos, and F. Fdez-Riverola, "Optimized convolutional neural network architectures for efficient on-device vision-based object detection," (in En;en), *Neural Comput. & Applic.*, vol. 34, no. 13, pp. 10469—10501, 2022, https://doi.org/10.1007/s00521-021-06830-w.
[10] H. Ha and J. Jeong, "CNN-based defect inspection for injection molding using edge computing and industrial IoT systems," *Appl. Sci.*, vol. 11, no. 14, p. 6378, 2021.
[11] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: a review," *Proc. IEEE*, vol. 111, no. 1, pp. 42—91, 2023.
[12] E. Solowjow, *et al*., "Industrial robot grasping with deep learning using a programmable logic controller (PLC)," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Hong Kong, IEEE, 2020, pp. 97—103.
[13] F. Schellroth, *et al*., "Latency optimized architectures for a real-time inference pipeline for control tasks," in *2021 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, Bahir Dar, Ethiopia, IEEE, 2021, pp. 166—171.
[14] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: a comprehensive survey," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 2, pp. 869—904, 2020.
[15] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial internet of things: a cyber-physical systems perspective," *IEEE Access.: Pract. Innovat., Open Solut.*, vol. 6, pp. 78238—78259, 2018.
[16] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Trends in AI inference energy consumption: beyond the performance-vs-parameter laws of deep learning," *Sustain. Comput.: Inform. Syst.*, vol. 38, p. 100857, 2023.
[17] A. Schmidt, F. Schellroth, M. Fischer, L. Allimant, and O. Riedel, "Reinforcement learning methods based on GPU accelerated industrial control hardware," *Neural. Comput. Applic.*, vol. 33, no. 18, pp. 12191—12207, 2021.
[18] R. Raßmann, *et al*., "Implementierung von Convolutional Neural Networks zur echtzeitfähigen Bildklassifizierung auf konventionellen Industriesteuerungen," in *Tagungsband AALE 2023: Mit Automatisierung gegen den Klimawandel*, Leipzig, Hochschule für Technik, Wirtschaft und Kultur Leipzig, 2023.
[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Computat.*, vol. 6, no. 2, pp. 182—197, 2002.
[20] F. Hutter, *et al*., *Automated Machine Learning: Methods, Systems, Challenges*, Cham, Springer, 2019.
[21] J. Blank and K. Deb, "Pymoo: multi-objective optimization in Python," *IEEE Access.: Pract. Innovat., Open Solut.*, vol. 8, pp. 89497—89509, 2020.

Bionotes

Christoph Wree
Department of Computer Science and Electrical Engineering, University of Applied Sciences Kiel, Grenzstr. 5, 24149 Kiel, Germany
christoph.wree@fh-kiel.de

Christoph Wree is a Professor of Automation Technology at the Kiel University of Applied Sciences. His research focuses on automation technology in the context of Industry 4.0 and the energy transition. Additionally, he investigates the application of machine learning in automation technology.

Rando Raßmann
Department of Computer Science and Electrical Engineering, University of Applied Sciences Kiel, Grenzstr. 5, 24149 Kiel, Germany
rando.rassmann@fh-kiel.de

Rando Raßmann obtained a Master of Engineering degree in electrical engineering from the University of Applied Sciences Kiel in 2022. He is currently working as a research associate at the University of Applied Sciences Kiel. His research interests include design methodologies and artificial intelligence.