Development of a Cloud- and Edge-Architecture for adaptive model weight optimization of a CNN exemplified by optical detection of hairpin welding

Johannes Vater[†][§], Maximilian Kirschning[†][‡], Dominik Scheurenberg[‡], Dirk Abel[‡] and Alois Knoll[§]
[†] Planning and Production of Electrified Powertrains, BMW Group, Munich, Germany
[§]Chair of Robotics, Artificial Intelligence and Real-time Systems, Technical University Munich, Munich, Germany
[‡] Institute of Automatic Control, RWTH Aachen University, Aachen, Germany
Email: Johannes.JV.Vater@bmw.de, Maximilian.Kirschning@rwth-aachen.de, D.Scheurenberg@irt.rwth-aachen.de, D.Abel@irt.rwth-aachen.de, knoll@in.tum.de

***Abstract*—The beginning of a global reorientation towards an increasingly conscientious approach to nature and the human habitat has been accompanied by changes in industry and society. The automotive industry, where a transition from combustion to electrically powered vehicles is currently underway, is also concerned with this change. In addition to increasing the capacity of the battery, improving the efficiency of the electric motor is essential. To achieve these goals, however, new technologies such as hairpins for the stator are needed. An important process step involves the welding of two pairs of hairpins, which often leads to welding defects. Nevertheless, expert knowledge in this field is limited. Optical monitoring of the welding process with the help of a convolutional neural network (CNN) is a good approach. This approach can compensate for the low level of expert knowledge and detects and classifies welding defects directly in the production line. However, the disadvantage of optical monitoring is that production conditions and the surrounding environment change over time. This has an impact on optical detection and can negatively affect the accuracy of a CNN. For example, the camera perspective can change, which has a negative effect on optical quality monitoring. Therefore, this paper presents an approach for monitoring and evaluating the quality of a CNN in a cloud instance online. If a deteriorating quality is detected, the CNN in the cloud is re-trained by continuously collected data and then automatically deployed to the production line. This allows the CNN to adapt to the changing environmental conditions. The present approach is demonstrated and validated with real data of the stator production process. Compared with the current state-of-the-art, this control loop is highly automated and requires a minimum of human intervention.**

***Index Terms*—cloud, edge, industry 4.0, image processing, adaptive optimization, hairpin**

I. Introduction

An electrical machine is one of the main components of an electrical powertrain. Compared with normal industrial electric motors, the electrical powertrain motor must satisfy special requirements including increasing the power density and efficiency while reducing the weight, costs, and production time. The implementation of these requirements demands the use of innovative technologies, such as the hairpin technology. A feature of this technology is that the classic copper windings of the stator are replaced by bent copper rods, which resemble hairpins. However, as described below, control of the current production process is difficult [1].

The lamination stack is fabricated at the beginning of the stator production process. Subsequently, the hairpins are inserted into the circular notches of the laminated core. The subsequent process step is the so-called twisting. During this process, the free hairpin ends are bent to form pairs of hairpins. In the subsequent step, the contacting and stripped hairpin pairs are connected electrically and mechanically using an automated laser welding process. A major disadvantage of the welding process is that copper absorbs less than 5% of the laser radiation at wavelengths of $\sim 1000nm$ and therefore has strong reflective properties. For this reason, laser beam welding requires a very intense power density [2]. However, this leads to the fault patterns shown in Fig. 1 [3]. Failures can be divided into four failure classes, namely correct welding (CW), insufficient welding (IW), welding spatter (WS), and welding craters (WC). In addition, these groups were distinguished based on the severity of the welding defects [4], [5].

The result of the laser welding process is currently monitored by an optical inspection system using a CNN. As shown in Fig. 2, the current system for monitoring and correcting welding defects is composed of the following three main components [4].

1) Production environment: This environment contains the basic system and its inputs and outputs. For the case addressed in this paper, the basic system is the hairpin welding station.

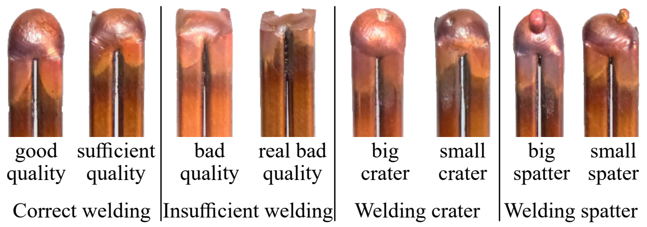

Fig. 1. Representation of the four error classes.

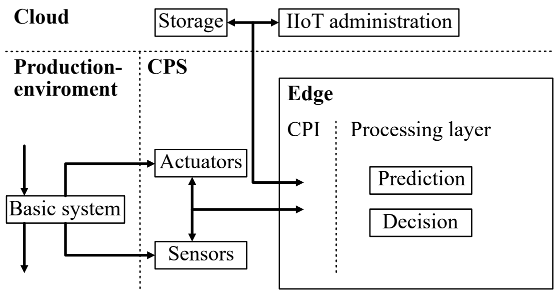

Fig. 2. Architecture of edge- and cloud-system.

2) Cyber Physical System (CPS): This includes all interfaces between the production area and the cloud. In addition, the sensors and actuators are also located in the CPS. The welding process is monitored with the help of a 3D scanner. As shown in Fig. 2, the edge-device, is located in the CPS. In parallel to the pre-processing of the raw data, the CNN is executed on the edge-device, which determines the welding class shown in Fig. 1. The actuators perform a reworking depending on the determined fault case. Owing to its local proximity to the production system, the rework can be performed in real time.
3) Cloud: Data storage and centralized management of the edge-devices both occur in the cloud.

The problem is that the optical inspection system and thus the CNN is highly dependent on the environmental conditions. Two example cases are the viewing angle of the camera system and the changing lighting conditions during the yearly season. If the optical inspection system fails to detect an existing fault, the stator passes through all further processing steps and is finally inspected at the end of the production line. In the case of a defective weld, the stator must be removed from the production line, the welding tool disassembled, the defect optically inspected, the stator manually aligned, the welding tool reassembled, the stator re-inserted, and the defective welds re-welded. This process is time-consuming and hence expensive. Thus, the present work focuses on the observation of the CNN accuracy, which allows automatic reaction to changing environmental conditions and improvement of the CNN via re-training. Self optimization of the system through the resulting data should guarantee that the best possible accuracy of the CNN is always realized.

II. State-of-the-Art

As explained in Section I, the welding process is monitored using an optical measuring principle, which is referred to as visual inspection [6]. According to DIN EN 1330, part 10, direct visual inspection with an uninterrupted beam path can be distinguished from indirect visual inspection with an interrupted beam path. The indirect inspection includes the use of photo and video technology [7]. Compared with other non-destructive testing methods, the advantages of an optical inspection method are the freedom of contact, speed, integrability, and low investment costs [8]. Following image acquisition, quality deviations in the production process can be detected and classified by means of the above-mentioned CNN. Studies have confirmed, that object recognition with the help of a CNN exceeds the performance of classic, manual methods such as HOG, SIFT or color histograms [9], [10]. An example quality control system for error detection with a CNN is described in Ref. [11], where the feasibility of the visual inspection system for welding defects in industrial production was evaluated.

However, automation systems are limited by their inability to capture, store, and analyze large amounts of data in real time in different environments [12]. A combination of cloud- and edge-computing provides excellent conditions for this purpose. A summary has been provided in Ref. [13], and hence, a detailed discussion focused on the state-of-the-art in the use of edge-computing in the manufacturing industry and in automotive production is excluded from this paper [13].

A. Motivation and objectives of this research

The disadvantage of the above described optical inspection system is the fact that it is negatively influenced by changing environmental conditions. The cloud-/edge-architectures described by Ref. [13] allow to re-train the CNN using computing resources located in the cloud.

Nevertheless, the quality of the neural network used must be manually checked by a human. Therefore, an automated observation of the CNN and its quality would be an advantage, as this observation allows to automatically react to changing environmental conditions and improvement of the CNN via re-training. The system should be capable of re-training on the data generated in production to ensure the highest accuracy of the CNN is always realized. The results obtained by answering three additional research questions are presented in this paper:

- Does the metric allow automatic detection of pseudo errors?
- Can intelligent data selection algorithms support the accuracy of the CNN?
- How can the quality of the re-trained CNN be automatically assessed before this model is deployed to the production line?

III. Data Generation

The goal is the generation of a data set covering all necessary classes. However, this is a challenging task that has to be carried out in an industrial production line. To capture the 3D data of the welding seams, a 3D camera, XR-HT40M from Keyence, was used. The benefit of using a 3D camera versus a classic 2D camera is a higher inspection stability, as the height information allows to obtain important information and features for the inspection process. The disadvantage, of course, is the price, which is significantly more expensive than a conventional 2D camera for industrial applications.

As a result, 550 to 600 images of hairpin welds for each class with different degrees of defect severity were generated, as shown in Fig. 1. Table I provides a detailed listing of the data set. The datasets for 3D images were split into training and validation datasets with 80% training data and 20% validation data. This initial data set was verified by an expert.

IV. Optimization and Architecture

A. *Impact of disturbances*

A stable production environment is required for producing components of consistent quality. However, production environments are influenced by numerous factors, even if they are protected against environmental changes.

The quality control of the hairpin welding through CNN is subject to particular disturbances. The basis of the quality control is the pre-processed 3D scan, which corresponds to a 30×30 pixel grayscale image resulting from the pre-processing of the raw images described in Section III. The image size is relatively small, and therefore, even small disturbances have a significant impact on the detection accuracy. The factors exerting the greatest impact include changing light conditions, contamination of the camera lens or the component itself, and changes in the perspective.

One of the key challenges of a CNN is to deal with data variances in the real world. To achieve the most accurate results possible, CNNs require a large number of diverse training images. This problem is commonly handled via data augmentation. The augmentation process flips, rotates or adds noise to the images, which will improve the results of the CNN, but the problem of generalizing across various perspectives remains unsolved. A major strength of CNNs is, that they are invariant to moderate translations. Real-world images will very likely include new viewpoints, new lighting conditions, and occlusions that are only poorly represented in the current image data set. The change in perspective, as show exemplary in Fig. 6 in the bottom row, leads to a supposedly different geometry of the same Hairpin and would require new training. To counteract these influences, we propose updating the data set using self-optimization techniques. The effect of different policies for updating the data set is examined in further detail below.

B. *Self-optimization*

We use the paradigm of self-optimization to solve the explained problem. According to Gausemeier [14], a self-optimizing process consists of the three steps shown in Fig 3.

During the analysis of the current situation, user requirements, the system status and the system environment are identified and checked. Observations can also be obtained indirectly through communication with other systems, and monitoring the fulfillment degree of the given system goals is essential.

TABLE I
Division of the dataset for the failure classes

Class	Training set	Test set	Sum
IW	456	104	560
WS	455	102	557
WC	438	125	563
CW	478	126	604
Sum	1,827	457	2,284

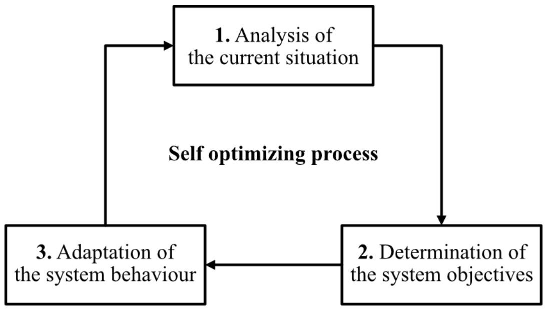
1. Analysis of the current situation
Self optimizing process
2. Determination of the system objectives
3. Adaptation of the system behaviour

Fig. 3. Self-optimization process.

Starting from previous system goals, new goals can be selected, adapted or generated. When goals are adapted, the existing ones are successively changed. The generation of goals means that new ones are created independently of existing ones.

The system behavior is adjusted in accordance with the system objectives. The adaptation of the system behavior can occur on the parameter, structural, and behavioral levels of the mechatronic system.

The cognitive control loop proposed by Schmidt (see Fig. 4) [15] represents a possible implementation structure of the described process. The control loop is composed of a three-layered system:

1) In the **perception layer**, all incoming data streams are acquired and forwarded to the central layer for information processing and decision making.
2) In the **layer for information processing and decision making**, the self-optimization of the process chain is realized by five system modules.
    a) The **communication module** distributes and coordinates the incoming data streams from the perception layer to the analysis, modeling, and optimization module. This module also links the information processing and decision making layer

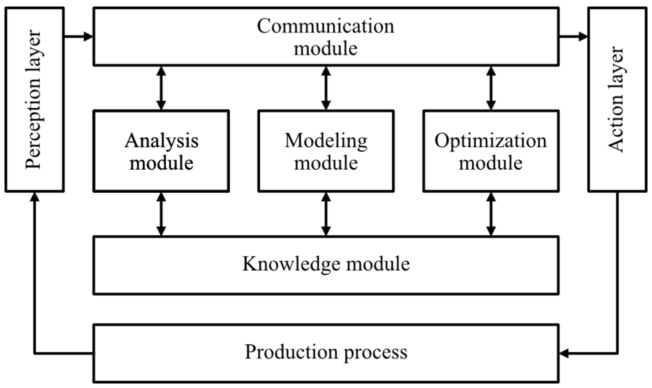

Fig. 4. Effect pattern of the cognitive control loop.

to the two outer layers.

b) In the **analysis Module**, the existing amount of data on a relevant subset is reduced
c) Based on the reduced amount of data, the **optimization module** is now able to optimize the process chain.
d) The decision is then evaluated in the **modeling module** by approximating and simulating the result using a system model. If the result matches the desired system goals, the decision is transferred to the communication module.
e) The **knowledge module**, which contains past data, decisions, and target configurations, acts as a database for each of the aforementioned modules.

3) The **action layer** serves to implement the decisions made and evaluated by the simulation.

C. Optimization goals

According to Gausemeier, optimization goals are essential for the implementation of a self-optimizing system [14]. As explained in Section III-B, these goals can either be variable or static. The required goals are application-specific, and hence, only the goals that are significant for the given application will be discussed in the subsequent sections. Within the given production context, a low failure rate of the product is a major criterion. The classifications as false positive are particularly critical here, as these lead to faulty parts being classified as fault-free. Furthermore, the confusion of a weld crater or weld spatter as an incorrect weld is critical, as these require different reworking strategies. The goals for the optimization are therefore a low misinterpretation of the prediction and a high level of safety. Owing to the multiple goals, a prioritization is needed. The exclusion of false positives plays a key role in producing goods, and therefore, we prioritize this goal, which is further addressed in Section IV-H.

D. Cloud-/Edge-architecture for the optimization loop

To implement the solution shown in Section IV-B, the given effect pattern is modified and a streamlined cloud-/edge-architecture is used. Within this architecture the communications module is substituted with a modern pipeline approach that follows the actual data flow. The edge-device performs the tasks involved in error detection close to the manufacturing line. These include the pre-processing of images, prediction of the error case by the CNN, decision on rework measures, and the communication with the production facilities and the cloud. Each edge-device is implemented as an industrial internet of things device [16]. The central organization module, which organizes the individual devices and serves as a data storage and computing resource, is located in the cloud.

The optimization of the CNN and its adaptation to environmental changes described above are implemented in the cloud. The necessary functions are implemented in four modules. Fig. 5 shows the actual architecture of the given optimization loop.

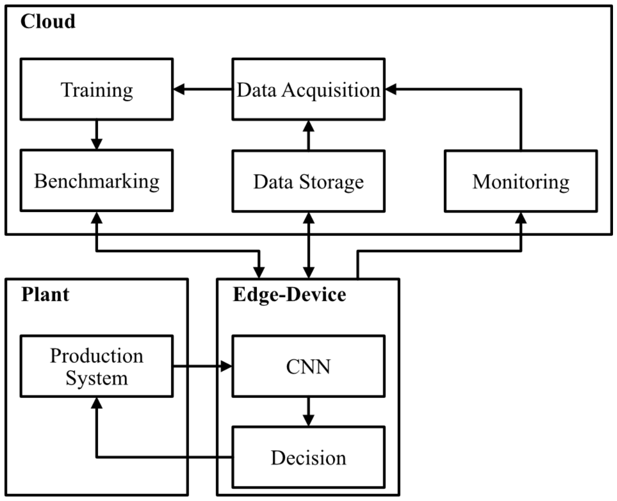

Fig. 5. Architecture of the optimization loop.

The **monitoring** module observes the current state of the prediction parameters of the CNN. This module resembles the perception layer proposed by Schmidt [15]. This includes the certainty of the CNN, which is generated by the Softmax-function, for the given prediction and the number of relabels. If the module finds any inconsistencies that can be fixed with re-training, the **training** module will be notified. Furthermore, the module serves as the central training resource and communicates with the data acquisition and benchmarking modules, and therefore, resembles the modeling module. When the training module receives a new training assignment, this module queries the **data acquisition module** (analysis module) for data sets that are expected to guarantee the best training results. Several versions of the same CNN are trained and compared via the **benchmarking module** (optimization module), based on these data sets. If a better CNN than the current one is found, the better CNN is deployed to the edge-device. The individual modules are explained in the following sections.

E. Data structure

Our data sets for training and validation of the CNN are divided into three categories. The general data pool contains all images and predictions generated by the production process, the training data set is formed from this pool. The pool contains all the data that was used for the last successful training process, and serves as a starting point for a new training process. The data acquisition module explained in Section IV-G is used to extend this data set, with new images and predictions are continuously generated during production. The environmental parameters of these images and predictions can change over time (see Section IV-A), and hence, expansion of the test data set over time is warranted. Therefore, the algorithm for matching two images (see Section IV-G) compares each new image with the test data set. If the image differs to a given percentage from all other images, then this image is designated as an unknown image and is included in the test data set. A common ratio of 80/20 of the training data to the test data has been established in the literature. This ratio is targeted for filling the test data set. If the test data set exceeds this ratio by a defined percentage, the oldest images are removed from the set. Through this method, the actuality of the test data set is ensured, as the set is always built up from the last relevant images.

F. Monitoring

The monitoring module observes the accuracy, precision and confidence of the CNN, which is implemented on the edge-device based on its process values. Furthermore, prediction-dependent values are distinguished from general values.

Prediction-dependent values represent the certainty of the CNN based on its prediction or whether the prediction is correct. They are compared for each prediction and thus offer the possibility of rapidly detecting deviations.

The general values are tested with only some predictions, but are queried at specific points in time. These include the precision and the F_1-score. They can be used to detect long-term errors which, due to their rare occurrence, are non-detectable during continuous testing.

For continuous analysis of the production-dependent values, the monitoring module observes each prediction and interprets the resulting values. In the given case, the number of incorrect predictions and the certainty about a prediction have been found to be robust values for detecting a change in the environment. To estimate the incidence of an incorrect prediction, the final predictions are evaluated. If the number of false predictions exceeds a given threshold value for the process, a new training process is started by the monitoring module. This value depends on the current production and varies in each application. The certainty of the network is checked by comparing the mean value of the final predictions with a reference mean value. If this value lies outside the first standard deviation that was determined by a data scientist, then the probability is high, that changes have occurred in the environment and the module initiates a new training process.

In order to detect the general values such as the precision, the model operating during production is tested against a test data set at a given time interval. This test data set is continuously expanded as described in Section IV-E and thus provides a good representation of the current environment. If the precision calculated in this manner drops below a threshold, then this is interpreted as an indication of environmental changes and a new training process is initiated.

G. Data acquisition

To obtain accurate classification results from the CNN, highly diverse training data is required. To achieve this diverse data, we propose updating of data set with new images, that represent information, which is not or only poorly represented in the current data set. An evaluation procedure for determining the similarity of the images is therefore needed. The procedure proposed in this paper is divided into two parts, finding similar images in the training data set and determining the difference between the new image and the similar images found in the data set.

We use image descriptors in the first part of the procedure to determine the most similar images. Therefore, we calculate the descriptors of the new image and the distances to the previously calculated descriptors of the images from the data set.

To calculate the image descriptors, we use the KAZE algorithm. Computing KAZE features is more computationally demanding than computing SIFT or SURF features, but they promise greater accuracy in both feature detection and description [17], [18]. The KAZE features are calculated by using nonlinear diffusion filtering. This method results in nonlinear partial differential equations (PDEs) that describe the change in the luminance of an image through increasing scale levels. An analytical solution for PDEs is mostly lacking, and therefore, additive operator splitting schemes are used for numerical approximations of the differential equations [17].

The best match of the image descriptors is found by determining the nearest neighbor of image descriptors from the images in the data set. The nearest neighbor is identified as the match with the smallest Euclidean distance. To ensure the uniqueness of the features, we use the method proposed in [18], where the distance between the closest neighbor and the second-closest neighbor is calculated. The second-closest match can be considered the likelihood of a false match. In this implementation, we discard matches where the difference in distance is greater than 0.75 to the second-closest match, thereby reducing the chance of false matches.

In the second part of the procedure, we evaluate the variation in the actual image compared with the most similar images identified in the first step. We use two characteristic values for this evaluation, the mean squared error (MSE) and the structural similarity index (SSIM). The MSE evaluates the absolute differences of the images per pixel [19]. The SSIM evaluates the spatial information and therefore the inter-dependencies ofthe pixels of both images. Compared with other methods, this renders the method less sensitive to translation and rotation [20], [21]. The combination of these characteristic values allows for an estimation even if the actual image contains new features that are not represented in the images of the current data set (see Fig. 6). Moreover, a MSE score of 0 and a SSIM score of 1 represent identical images.

The algorithm described in this Section is used for filling up the test data set as shown in Section IV-E, as well as for creating the training data set. Each image created and predicted in production is tested against the test data set using the algorithm, and if found to be new, is added to the test data set. If the monitoring has determined a deviation of the ML model, and a new training order is created, a training data set is created using the above algorithm. A search for suitable images from the general data pool is therefore performed starting from the old data record. Preferably, the system searches for images that are most similar to the images causing the error. If these are identified, the data set is forwarded to the training module.

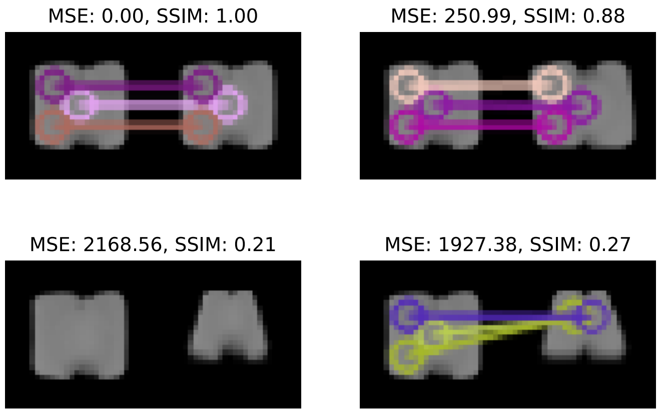

Fig. 6. Comparison of hairpin images with corresponding MSE and SSIM score and drawn in matching image descriptors (colored circles).

H. Training and benchmarking of the CNN

The training module contains the actual processes for training the CNN. This module receives a selection of data sets from the data acquisition module, which should provide a reliable model. Training is quite time consuming leading to downtime in the production, therefore the training process is performed on cloud clusters designed for this task. This design ensures that the training runs parallel to the production without influencing it.

The created models are evaluated by the benchmarking module. Each model receives a score, which can consist of different scoring mechanisms. For our purpose, the analysis using a confusion matrix has proven to be particularly useful. In this matrix, the frequency of false predictions can be directly determined. The production process makes certain demands regarding the prediction, and these can be directly tested. The demands are that no faulty product is predicted as a correct product and no wrong rework measure is executed as described in Section IV-C. By comparing the matrices associated with each model, the suitability of each model is determined. In Section I, the given application case is discussed in further detail. The scores used for benchmarking are individually tailored to the problem, and hence, they can vary greatly. Further possibilities would be, for instance, the precision.

If a network with greater accuracy network, than the current one is found, the network is transferred to the deployment module and saved as a valid model in the cloud. This process is described in more detail in the following Section.

I. Deployment

To deploy the CNN on the edge-device, a fixed pipeline needs to be provided. First, the model is embedded in a container, which is enclosed with a REST-API that can be used to make requests for prediction. This container is then deployed in the cloud and validated parallel to production. If the new model proves to be more accurate than the old one, the container is loaded onto the edge-device. Both are being executed on the edge-device simultaneously, while the classification of the old network is given out. The transition to the output of the new model is then made between individual cycle times.

J. Validation of the given Architecture

In addition to the real data described in Section III we created a series of manipulated images (by changing the camera angle) to validate the architecture presented above. To manipulate the real data, described in Section III, we used the function *warpPerspective()* from OpenCV. Only insufficient welds are used here, as these have proven to be particularly critical. For example, Fig. 6 shows the original images in the upper row and the modified images in the lower row. Using this method, 546 images, which are divided into a test data set of 100 images and a training data set of 446 images, are created. Fig. 7 shows a confusion matrix of the old model tested against the new test data set (changed environmental conditions). Here, a misinterpretation of the IW can be easily seen, as two are classified as CW and 31 as WS. This will be detected by the monitoring module, since the number of relabels will be unusually high.

Fig. 8 shows the CM after re-training. The IW has improved significantly, since all images are now correctly classified. The slight decrease in the prediction of WC has no influence on the subsequent production process, because the reworking procedure is the same. However, further analysis and optimization can also improve the performance of the model.

Compared with the old data set, the new data set contains more diverse images. This shows, that the data acquisition algorithm can balance the data set. The average MSE score of a fake image compared with the new data set is significantly lower than that of the old data set, whereas the SSIM score is higher, as seen in TABLE II. This indicates that the data acquisition algorithm is able to expand the data set with the new images (changed environmental conditions) that was inadequately represented in the data set.

Overall, the results demonstrate that an automated system can adapt an existing machine learning model to its envi-

	CW'	IW'	WS'	WC'
CW	40	0	0	0
IW	2	107	31	0
WS	0	0	40	0
WC	0	0	0	40

Fig. 7. Confusion matrix a priori.

	CW'	IW'	WS'	WC'
CW	40	0	0	0
IW	0	140	0	0
WS	0	0	40	0
WC	0	0	4	36

Fig. 8. Confusion matrix a posteriori.

TABLE II
MEAN SCORES FOR MSE AND SSIM

Scores	Old training data	New training data
MSE	1554.31	376.01
SSIM	0.39	0.79
Similarity	1.88%	9.71%

ronment. However, a comparison of the parameters in the correct production technology context is always particularly important.

V. Conclusion and Outlook

A system that detects faulty predictions in a machine learning algorithm deployed in a production environment and automatically optimizes the model against the given errors is proposed in this work. An architecture capable of observing a given model, analyzing this observation, and generating a new model from the observation, is developed. Algorithms, which allow targeted data selection, are employed for this development. Subsequently, the architecture is validated on a real-world case study and evaluated against the specified objectives. The results revealed that the system optimizes itself automatically when environmental changes occur.

In future studies the proposed architecture could be tested on other production units. Here, other machine learning models could be tested and other parameters could be used to compare the models.

The machine learning models could also form the basis of an ensemble model, which combines the predictions from multiple models to improve the overall performance. Combinations of multiple models (in general) lead to lower variance in the predictions, and hence may provide more reliable predictions than a single model.

References

[1] A. Riedl, M. Manusch, M. Weigelt, T. Gläßel, A. Kühl, S. Reinstein, and J. Franke, "Challenges of the Hairpin Technology for Production Techniques," in *2018 21st International Conference on Electrical Machines and Systems (ICEMS)*, 2018, pp. 2471–2476.

[2] A. Blom, P. Dunias, P. van Engen, W. Hoving, and J. de Kramer, "Process spread reduction of laser microspot welding of thin copper parts using real-time control," in *Photon Processing in Microelectronics and Photonics II*, 2003.

[3] P. Schmidt, "Laser beam welding of electrical contacts of Lithium-ion batteries for electric- and hybrid-electric vehicles," Dissertation, Technical University of Munich, Munich, 2015.

[4] J. Vater, P. Schamberger, A. Knoll, and D. Winkle, "Fault Classification and Correction based on Convolutional Neural Networks exemplified by laser welding of hairpin windings," in *2019 9th International Electric Drives Production Conference (EDPC)*. IEEE, 2019.

[5] A. Mayr, B. Lutz, M. Weigelt, T. Gläßel, D. Kißkalt, M. Masuch, A. Riedel, and J. Franke, "Evaluation of Machine Learning for Quality Monitoring of Laser Welding Using the Example of the Contacting of Hairpin Windings," in *2018 8th International Electric Drives Production Conference (EDPC)*. IEEE, 2018, pp. 1–7.

[6] K. Schiebold, "Zerstörungsfreie Werkstoffprüfung - Sichtprüfung, Non-destructive testing of materials - Visual inspection (english)." Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[7] D. D. I. für Normung e.V., "Zerstörungsfreie Prüfung – Terminologie – Begriffe der Sichtprüfung, Non-destructive testing - Terminology - Visual inspection terms (english)," 2002.

[8] J. Huber, C. Tammer, D. Schneider, C. Seidel, and G. Reinhart, "Non-destructive Quality Testing of Battery Separators," *Procedia CIRP*, vol. 62, pp. 423–428, 2017.

[9] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks." ACM Press, 2015, pp. 161–170.

[10] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 10, pp. 6232–6251, 2016.

[11] P. Sassi, P. Tripicchio, and C. A. Avizzano, "A Smart Monitoring System for Automatic Welding Defect Detection," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 12, pp. 9641–9650, 2019.

[12] D. Wu, S. Liu, L. Zhang, J. Terpenny, R. Gao, T. Kurfess, and J. Guzzo, "A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing," *Journal of Manufacturing Systems*, vol. 43, pp. 25–34, 2017.

[13] J. Vater, P. Schlaak, and A. Knoll, "A Modular Edge-/Cloud-Architecture in automotive production for Automated Error Detection and Correction using Machine Learning," in *IEEE Computer Society Signature Conference on Computers, Software and Applications (COMPSAC)*, 2020, in press.

[14] J. Gausemeier, F. Rammig, and W. Schäfer, "Design methodology for intelligent technical systems: develop intelligent technical systems of the future." Springer, 2014.

[15] A. Schmidt, "Wirkmuster zur Selbstoptimierung - Konstrukte für den Entwurf selbstoptimierender Systeme, Effect patterns for self-optimization - Constructions for the design of self-optimizing systems (english)," ser. HNI-Verlagsschriftenreihe, 2006.

[16] J. Vater, M. Kirschning, and A. Knoll, "Closing the loop: Real-time Error Detection and Correction in automotive production using Edge-/Cloud-Architecture and a CNN," in *IEEE International Conference on Omni-layer Intelligent systems*. IEEE, 2020, in press.

[17] P. Alcantarilla, A. Bartoli, and A. Davison, "KAZE Features," *Computer Vision – ECCV 2012 Lecture Notes in Computer Science*, p. 214–227, 2012.

[18] S. Tareen and Z. Saleem, "A comparative analysis of sift, surf, kaze, akaze, orb, and brisk," in *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*, 2018, pp. 1–10.

[19] R. Gonzalez and R. Woods, *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[20] W. Zhou, A. Bovik, S. H., and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[21] Y. Gao, A. Rehman, and Z. Wang, "CW-SSIM based image classification," in *2011 18th IEEE International Conference on Image Processing*, 2011, pp. 1249–1252.