A Modular Edge-/Cloud-Solution for Automated Error Detection of Industrial Hairpin Weldings using Convolutional Neural Networks

Johannes Vater[†§], Pascal Schlaak[†] 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*
Email: Johannes.JV.Vater@bmw.de, Pascal.Schlaak@studmail.htw-aalen.de, knoll@in.tum.de

***Abstract*—The traction battery and the electric motor are the most important components of the electrified powertrain. To increase the energy efficiency of the electric motor, wound copper wires are being replaced by coated rectangular copper wires, so-called hairpins. Hence, to connect the hairpins conductively, they must be welded together. However, such new production processes are unknown compared with classic motor production. Therefore, this research aims to integrate Industry 4.0 techniques, such as cloud and edge computing, and advanced data analysis in the production process to better understand and optimize the manufacturing processes. Welding defects are classified with the help of a convolutional neural network (CNN) (predictive analysis) and, depending on the defect, a recommended course of action for reworking (prescriptive analysis) is given. However, the application of such complex algorithms as neural networks to large amounts of data requires huge computing resources. Therefore, a modular combination of an edge and cloud architecture is proposed in this paper. Furthermore, a pure cloud solution is compared with the edge solution.**

***Keywords*-cloud computing; edge computing; machine learning; convolutional neural networks; electric motor; hairpin; predictive analytics; prescriptive analytics**

I. Introduction

The electric drive train has special requirements for the drive system and also for the automotive industry, and its production sector. For this reason, it is necessary to develop innovative technologies to optimize production and increase the efficiency of the electric powertrain [1]. One novel technology is referred as hairpin technology. This involves replacing the traditional copper windings of the stator with thick copper bars. One of the main step is the welding of these copper bars. However, this process step is unstable and at the same time generates faults. Furthermore, this welding process cannot be monitored and the different types of possible defects cannot be classified and therefore not reworked directly in the production line.

This paper aims to detect and classify defective welds directly in the welding station to give an employee a recommendation for rework with the help of a CNN. Powerful hardware components and novel technologies of distributed computing offer good conditions for a real-time capable optimization of production. Edge computing enables functionality to be provided close to the process. A direct connection of components of the production line and processing of functionality in real-time becomes possible. The concepts such as quality prediction to avoid further processing of faulty components offer an opportunity to reduce scrap and rework times. Within the scope of this paper, an architecture combining both edge and cloud computing for automated error detection using a CNN will be developed and validated based on the production process described above. Real-time production data provides the basis for predicting defective welds and providing appropriate recommendations for action. A modular structure facilitates the integration of the solution into other use cases, while a validation of the designed architecture compared to a cloud solution clarifies the advantages of edge computing.

II. State-of-the-Art

In this section, the state-of-the-art in the use of edge computing in the manufacturing industry, and automotive production is discussed. To get an overview of approaches that deal with edge computing we reviewed scientific papers in this fields. Therefore, in the following the most relevant publication are listed.

In their work, Hou et al. focus on the use of edge computing and machine learning for the early detection of process problems within semiconductor technology. Therefore, they are using a single board computer, which in this case is the edge device, and a digital microscope [2]. Trinks and Felden describe in their work a real-time quality management system, to increase the efficiency of a 3D printing system. If an error is detected, the printing process is stopped and a message is sent to the operator in charge [3]. Lin et al. deal in their work with the planning of complex semiconductor manufacturing systems and the associated distribution of tasks. In the cloud data center, task scheduling predictions are generated and sent to the appropriate edge devices [4]. Luckow et al. evaluate in their research different application possibilities of DL in the logistic area of an automotive production. The edge device is a smartphone of the worker [5]. Borangiu et al. describe in their work an agent-based edge computing approach for the detection of defects in automotive production. In the absence of an insufficient amount of data, an anomaly detection system

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.0-202

using unsupervised learning algorithms was used to detect deviations and to initiate rework as early as possible [6]. Syafrudin et al. describe in their work, a cost-effective early warning system. To act in real time, the four edge devices were installed near different production processes of a door panel. Errors are distributed to the employees after detection via a push notification within the front-end application and via a smart-phone message [7].

A. Need for Action and Objectives of this Research

The current applications presented above are not in the area of production with high cycle times. Another strong deficit of the publications is that there is no modular structure of the systems. Therefore, these systems cannot be adapted without great effort. Furthermore, there is no central management of the software. This means that, for example, adjustments to the software must be carried out physically on site and not centrally. Additionally, there is currently only one application where cloud computing is used in the production of electric motors. There are no edge computing applications in this area [8]. Thus, this paper extends the application range of edge computing and combines two innovative technologies: the production of the electric powertrain of a car with cloud and edge computing.

III. Design and Implementation of the Architecture

A. Business Understanding

The main steps in the production of the stator are the deformation of the copper rod into a shape which resembles a hairpin. These hairpins are then inserted into the stator lamination stack followed by twisting the hairpins and joining them together using laser welding [9]. A problematic aspect of this process, however, is the high reflection of copper, which hardly absorbs any radiation. Consequently, a higher laser power must be selected unlike when welding steel or aluminium [10]. At the moment this welding process cannot be monitored and the different types of possible defects cannot be classified. As a result, the stator goes through all further processing steps until it is finally tested at the end of the production line. If a faulty weld occurs the stator must be removed from the production line, the welding tool disassembled, the defect inspected visually, the stator aligned manually, the welding tool reassembled, the stator reinstalled and the defective welds rewelded. This requires a lot of time and expensive as well. Since a certain error-proneness of the laser welding process cannot be excluded, an approach is needed that enables the detection of the errors. One stator consists of 54 pairs of hairpin. Furthermore, the defined cycle time of the production line is 30 seconds. Due to the high number of welds, a real-time capable system is needed.

For this reason, we first analyzed the welding process in detail with a technologist and then identified four important types of weld seams in series production. These

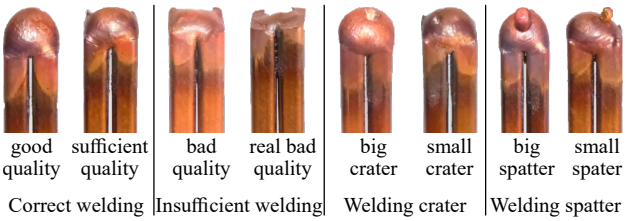

Fig. 1: Representation of the four error classes

classes are divided into four groups, namely correct welding (CW), insufficient welding (IW), welding spatter (WS), and welding craters (WC) like illustrated in Fig. 1. In addition, a distinction was made according to the severity of the welding defects.

B. Requirements

A definition of the requirements forms the basis for a specification of the system. These requirements were established through semi-structured interviews with twelve experts in the fields of data analytics, control engineering, Internet of Things (IoT), and production:

- Real-time prescriptive analytics: The most important requirement is the proactive delivery of recommendations for action in the event of quality deviations. It must be guaranteed under real-time requirements to avoid loss of cycle time. As already mentioned, a stator has 54 hairpin pairs, which have to be welded. The solution to be developed here can therefore not exceed a cycle time of 30 seconds.
- Modularity: A modular design of the system is essential so that the system can be easily adapted or extended to other applications.
- Data-driven modeling: The detection of quality deviations with the help of data-driven algorithms such as a CNN has the advantage that no expert knowledge is required for programming. Therefore, such a model should be preferred.
- Iterative improvement of the model: Production conditions usually change over time. As described in the previous requirement, a CNN should be used. This means that it must be retrained with new data and adapted to the changed conditions. A suitable processing resource is required for this.
- Relabeling: Machine-based learning algorithms can mis-classify the quality of a product. The data and its labels are used to retrain the algorithm in case of changing conditions. It is essential that the labels are correct. Therefore, an employee must have the possibility to change already generated labels.
- Central management: For self-optimization of production, a newly trained model as described above must also be deployed in production. Also, it is important that an employee can change this remotely. In addition,

all modules shall be automatedly deployed from a central position via a continuous integration and continuous deployment process (CI/CD) to the desired edge device.
- Independence of control hardware: The independence and interoperability of the edge devices is important. This ensures that the model for detecting quality deviations and generating recommendations for action is not dependent on specific hardware.
- Operational data storage: The edge device unit must be able to efficiently store and manage 'active' data of the operative production via real time access.
- Long-term data storage: Consistent long-term documentation and provision of historical data is required to generate knowledge about interrelationships and problem causes for model retraining.

C. High-level design

The design of the overall concept describes the general system structure. The high-level design is illustrated in Fig. 2. The components are described in the following Sections. Subsystems of the solution to be developed here can be assigned to the three levels: production environment, edge, and cloud.

The production environment contains all components that are physically in the production process. Products are processed and transformed. In this application, the production line is the hairpinwelding station, which is equipped with a 3D scanner. This 3D scanner is located directly in the laser welding process and takes 3D scans of the welds of the hairpin pairs. For each pair, the camera generates a three-dimensional image of the dome.

The components of the edge level are in direct communication with components of the production process and are in physical proximity to it. The functionalities on the edge device, which we call modules in the following, include the steps: acquisition, preprocessing, prediction, storage, and interaction. These are explained in more detail in Section III-E.

The cloud is used as a data storage and organization platform. It coordinates the data collected by the multiple edge devices and stores them in a central location.

As a basis for this work we use Microsoft Azure. Azure IoT Edge is one service, which is provided via the Azure Cloud. The IoT Edge runtime provides the basis for the function modules to be developed which consists of two components: IoT Edge Hub and IoT Edge Agent. The IoT Edge Hub serves as a communication interface between the individual function modules. The communication between the modules, which are decoupled from each other, is made possible by a publish-subscribe mechanism using MQTT, with an IoT Edge Hub acting as a message broker. Furthermore, data can be exchanged between the edge device and the cloud via defined interfaces.

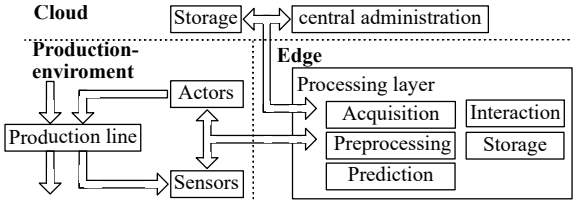

Fig. 2: High-level design of edge- and cloud-system

To enable a modular structure of functionality, independent of the respective hardware, an additional technology is required. For this reason we use container technology.

In the following a detailed description about the components of each level is listed.

D. Components of the Production Environment

As described, the sensor used in the production process has an optical measuring principle. The advantages of an optical test method are non-contact, speed, and integrability to other use cases. Due to a higher accuracy of the CNN, a 3D scanner was used instead of an ordinary camera. The accuracy of the CNN with 3D data instead of ordinary images as input is about 5% higher. For the acquisition of the images we developed an interface between the scanner and edge device. The Keyence XR-HT40M 3D scanner used in this application provides a file transfer protocol (FTP) client function, which allows images to be transferred to an FTP server automatically. Therefore, the acquisition module enables an FTP server on the edge device to receive data from the camera using FTP.

E. Components of the Edge level

The edge device used in this research is an industrial personal computer (IPC) of the type Siemens SIMATIC IPC427D. In the following the modules of the edge level are explained.

1) Preprocessing module: A dissemination of data between the acquisition module and the preprocessing module cannot be realized via MQTT, since messages to be sent exceed the maximum message size, which arises from the file size of the raw data images (12.29 megabytes). Therefore, we are using an exchange directory on the edge device, which several modules can access. The acquisition module stores data in a directory on the IPC, whereas the preprocessing module waits for new data in this directory and then starts a preprocessing of the data. Preprocessing decodes the height information from the 3D scan, reduces the image size, converts it into a grayscale image and normalizes the data. The exact procedure is described in detail by the authors in Ref. [11]. Subsequently, for a prediction of quality, all information is published. Then, a message object is created for each preprocessed hairpin. Each message is published via MQTT on a topic through the IoT Edge Hub. By preprocessing the data, the size of an image can be reduced to 1.99 kilobytes.

2) Prediction module: To predict the quality of the hairpin, as shown in Fig. 1, we developed a CNN, which is based on the VGG network. The basic structure of our CNN is a series of convolution blocks. These convolution blocks consist of two consecutive filters including ReLU activation layer, followed by a pooling layer. The number of filters is doubled from one to the other convolution block. The first layer uses 8 filters of the size $3x3$. The second block has 16 filters, the third 32 filters and so on. This model consists of four convolution blocks including batch normalization (BN) and ReLU activation. At the end of each block, except the last one, a max pooling is applied. For the last block, global average pooling is used instead of max pooling. This is followed by a fully interconnected layer with 32 neurons, again a BN layer, a ReLU activation function, and a drop-out layer with a drop-out rate of 0.5. Finally, in this architecture, an output layer with one neuron for each class and a softmax activation function follows, so that a probability for each class is generated as output. A total of 1827 training data sets were used to optimize the network architecture and multiplied by an augmentation algorithm to produce $91,350$ samples. A validation of the results was carried out using 457 test data. We achieved an average accuracy of 99.21% using this network architecture, and therefore used as a prediction algorithm in the prediction module. More information that clarifies and justifies our choice of methods for classification of welding defects in copper hairpins and a more detailed explanation of the model and its construction is provided by the authors in previous publications [11].

When the prediction module is started, the CNN model is loaded and the messages of the preprocessing module are subscribed to. As a result, a list with the probabilities of the four classes of the weld is expected. Furthermore, the class with the highest probability is identified and checked if it exceeds a threshold value of 80% otherwise, the weld is marked as uncertain decision (UD). An UD is intended to allow for later improvement of the CNN by manual reclassification using the web app. Finally, a message of the predicted data is published.

3) Data storage module: This module has the task to store the preprocessed images and information like the class of the welded hairpin on the edge device and in the cloud. We are using a local database to store process information such as pin numbers in an SQL server. Therefore, we implemented this server in a separate docker module to ensure consistent modularity. Furthermore, we implemented an additional directory of the IPC for storing the preprocessed 3D scans to allow access to the data by other modules, such as the web app. Moreover, the local data is moved to the cloud and deleted at certain intervals so that the storage resources of the edge device are not overloaded in productive operation.

4) Interaction module: This module provides two important functionalities. Firstly, it provides an actual status of the process and the welded haripins and offers the operator the strategy for the rework. Secondly, it offers the possibility, to relabel already classified welding defects which are saved in the SQL database.

The application can be used on different devices, such as laptop and smartphone, so that an employee is always informed about the current process.

A traffic light system (green = OK, yellow = UD, red = Not OK) shows the current status of the production process of a stator, as shown in Fig. 3, in the top element of the start page (see 1). Three elements of equal size (see 2) below this notification element visualize important quality key figures of a production day, and this creates transparency for managers. If new hairpin pairs are welded together, the prediction of quality is displayed by coloring the elements in the stator view (see 3). A listing of UD (see 4) is intended to provide a user with an interface for direct reclassification. This is essential to continuously improve the accuracy of the CNN as requested. All not OK predictions of the current welds are displayed in the lower right element of the start page (see 5) together with a rework strategy. The rework strategy of each welding fault is explained by the authors in previous publications [11]. If the user now wishes to change a UD classification, he will be redirected to a new page where an overview of all relevant information is listed. There, a selection of the class has to be made first. Important information and an example are provided to the employee. This simplifies the selection, even if an employee is an inexperience user.

Subsequently, all subclasses belonging to the selected class are visualized and described. The subclasses of a not OK weld are IW, WC, WS as described in Section III-A. Finally, all information of the relabeling is visualized. To overwrite the old information in the cloud a confirmation must be made. The user is redirected to the homepage of the web app.

F. Components of the Cloud level

This Section lists the implementation of the modules in the cloud level.

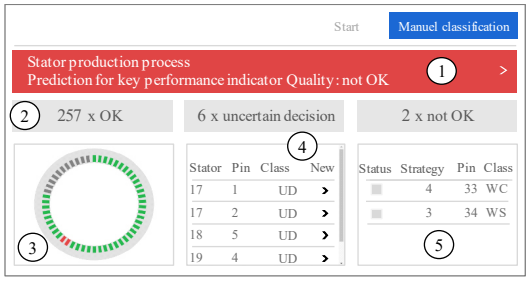

Fig. 3: Homepage

1) Data storage module: Beside saving the data on the edge device, we continuously replicate the SQL database and the object storage of the edge device. Process information and preprocessed images are also stored in the cloud. Therefore, we are using the Azure SQL database service. Based on Transact-SQL, tables with high availability can be used on server instances. For managing the preprocessed images, we are using a blob storage in the cloud.

2) Central management: In addition to managing modules, the cloud is to act as a central control unit. Software code is hosted via services of the cloud and automatically provided on the edge device. The code is automatically built into a container image and loaded into a storage instance, container registration, of the cloud. To provide the container images, an IoT Hub instance is required in the cloud, which communicates with the IoT Edge runtime of the edge device.

3) Continuous improvement of CNN: In order to continuously improve the CNN and adapt it to possible environmental conditions, the CNN is trained in the cloud on the basis of new data. Thus, a virtual machine (VM) is started in the cloud.

G. Implementation of the Architecture

For data transfer between the modules, messages for specific topics can be published and subscribed too. We defined the topics in such a way that communication takes place as shown in TABLE I, which illustrates how modules receive and publish data. The acquisition module receives images from the camera via FTP and stores them in a shared directory on the IPC. The preprocessing module waits for new data in this directory, processes the image data, and publishes results as a message about the topic preprocessing via MQTT. The prediction module subscribes to this topic, performs necessary functionalites to predict the class and creates another message about the topic prediction, which is also published via MQTT. The storage module subscribes to the messages of the prediction module via MQTT and stores them in local instances and instances of the cloud. The web app for generating and visualizing a recommended action for the rework queries data from the local administration instances and changes database entries if necessary. The final architecture of the edge-computing solution is illustrated in Fig. 4.

TABLE I: DATA EXCHANGE BETWEEN THE MODULES

Module	Input	Output
Acquisition	FTP Camera	Directory IPC
Preprocessing	Directory IPC	MQTT: preprocessing
Prediction	MQTT: preprocessing	MQTT: prediction
Storage	MQTT: prediction	data and object storage
Interaction	data and object storage	Data storage

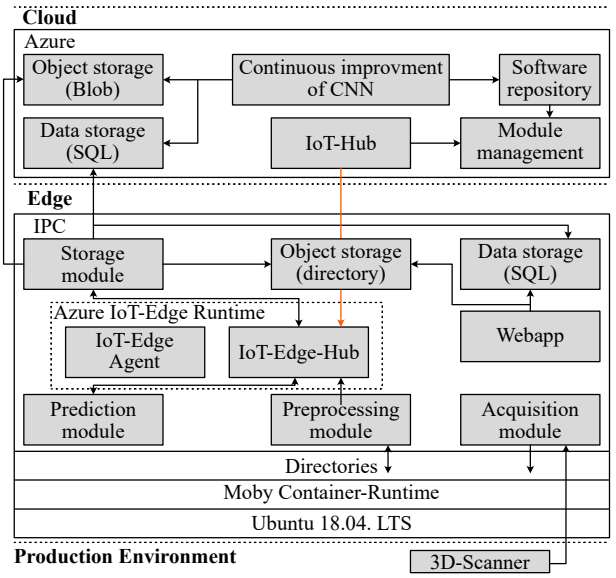

Fig. 4: Final architecture of the edge and cloud computing solution

IV. VALIDATION

To validate the system, we compared the above described edge computing solution with a cloud computing solution.

A. Experimental setup

To ensure a similar setup for the cloud and edge computing solution, both systems were deployed on identical VMs in the cloud. Both VMs were located in the same region, contained identical hardware specifications and used identical versions of required software. Each VM contained two vCPUs, 8 gigabytes of RAM and disk storage. The transformation of the edge computing solution to the cloud contradicts the purpose of the edge technology, but was necessary for the comparability of the systems.

An FTP client is used to simulate the camera in the production line by reading raw data images from the production environment and communicating them to the VM via FTP. An FTP server serves as an acquisition module and provides the counterpart within the VM to receive the data from a client. If data from a client is received via the server, the data can be written to a previously defined directory of the host system, in this case a VM. The preprocessing module waits for new events and the above described processing pipeline starts.

In contrast to the cloud system, the client for reading and sending data via FTP is located in the same virtual layer in the edge computing solution. An FTP server is also used to capture new raw data images. The logic for processing, prediction, and management of the images is also identical. Processed data, however, is managed in local storage instances, not via services outside the VM.

In this experiment we primarily investigated processing times of the duration of relevant processing steps, as well as the total processing time of the systems, was recorded using defined measurement points, see Fig. 5.

B. Results

In the course of the experiment, 54 welds of a stator were processed by each system, which allowed an average value of the different processing times to be calculated. All times of a processing step were summed and divided by the number of welds. TABLES II shows all average values of the processing times for the cloud and edge systems. It contains the average processing time per individual weld and the average duration of the processing steps per stator. Due to the parallelization of computing processes, the total time is not the sum of the individual time stamps. For example, a second pair of hairpins can already be processed even though processing of the first pair has not yet been completed.

As shown in TABLES II, an edge computing solution is clearly advantageous. Compared to the cloud computing solution, the edge computing solution reduced the processing time of a stator enormous. This corresponds to a reduction of the runtime by $1 - \frac{15{,}774s}{107{,}621s} \approx 85\%$. Saving the data in the object store and the SQL database in the cloud is comparatively faster than storing the data in local instances. However, this is due to very powerful computing resources of the cloud centers such as high read and write rates of the hardware. To summarize, all results satisfy the requirements. A total processing time of the edge computing solution of 15.774 seconds fulfills the time requirements of the process, providing a 30-second time window to capture the image data. The container technology and the publish-subscribe mechanism also allow a modular structure. Depending on the application, containers can be adapted, added or exchanged. TABLE III summarizes the requirements, listed in Section III-B and compare them with our provided solution.

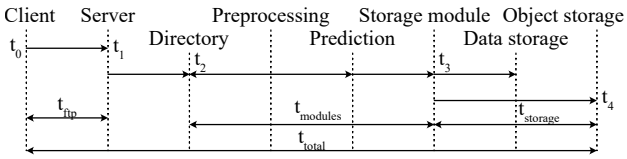

Fig. 5: Measuring points for determining the processing times

TABLE II: AVERAGE DURATION OF PROCESSING STEPS PER WELD AND STATOR

System	t_{total}	t_{ftp}	$t_{modules}$	$t_{storage}$
Average duration of processing steps per weld				
Cloud	3.986s	3.946s	0.716s	0.135s
Edge	0.584s	0.115s	0.692s	0.227s
Average duration of processing steps per stator				
Cloud	107.621s	106.541s	19.342s	3.640s
Edge	15.774s	3.093s	18.680s	6.117s

TABLE III: COMPARISON OF REQUIREMENTS AND SOLUTION

Requirements (Section III-B)	**Solution Component**
Real-time prescriptive analytics	Edge architecture. Processing time for one stator: 15.8 seconds
Modularity	Container technology and publish-subscribe mechanism
Data-driven modelling	CNN with an accuracy of $99,21\%$
Iterative improvement of the model	VM in the cloud and transparent database in the cloud
Central management	Azure CI/CD pipeline
Independence of control hardware	Container technology
Relabeling	Interaction Module
Operational data storage	Storage module, SQL module and directories (Edge)
Long-term data storage	SQL module and blob storage (Cloud)

REFERENCES

[1] A. Kampker, "Electric vehicle production (English) Elektromobilproduktion (original title)," Springer Vieweg, pp. 2471-2476, 2014.

[2] D. Hou, T. Liu, Y. Pan, J. Hou, "AI on edge device for laser chip defect detection," IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 247-251, 2019.

[3] S. Trinks, C. Felden, "Image Mining for Real Time Fault Detection within the Smart Factory," IEEE 21st Conference on Business Informatics (CBI), pp. 584-593, 2019.

[4] C. Lin, D. Deng, Y. Chih, H. Chiu, Hsin-Ting, "Smart Manufacturing Scheduling with Edge Computing Using Multi-class Deep Q Network," IEEE Transactions on Industrial Informatics, 2019.

[5] A. Luckow, K. Kennedy, M. Ziolkowski, E. Djerekarov, M. Cook, E. Duffy, et al., "Artificial Intelligence and Deep Learning Applications for Automotive Manufacturing," 2018 IEEE international conference on Big Data, pp. 3144-3152, 2018.

[6] T. Borangiu, "Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future," Springer, 2019.

[7] M. Syafrudin, N. Fitriyani, G. Alfian, J. Rhee, "An Affordable Fast Early Warning System for Edge Computing in Assembly Line," Multidisciplinary Digital Publishing Institute, 2019.

[8] A. Mayr, M. Weigelt, J. von Lindenfels, J. Seefried, M. Ziegler, N. Urban, et al., "Electric Motor Production 4.0 - Application Potentials of Industry 4.0 Technologies in the Manufacturing of Electric Motors," 8th International Electric Drives Production Conference, 2018

[9] A. Riedel, M. Masuch, M. Weigelt, T. Glabel, A. Kuhl, S. Reinstein, et al., "Challenges of the Hairpin Technology for Production Techniques," 21st International Conference on Electrical Machines and Systems, pp. 2471-2476, 2018.

[10] P. Schmidt, "Laser beam welding of electrical contacts of Lithium-ion batteries for electric- and hybrid-electric vehicles," Doctoral Thesis, Technichal University of Munich, 2015.

[11] Vater, J. and Schamberger, P., Knoll, A. and Winkle, D., "Evaluation of Machine Learning for Quality Monitoring of Laser Welding Using the Example of the Contacting of Hairpin Windings," 9th International Electric Drives Production Conference, 2019.