

A Service of



Leibniz-Informationszentrum Wirtschaft Leibniz Information Centre for Economics

Beck, Michael; Layh, Michael; Nebauer, Markus; Pinzer, Bernd R.

Article — Published Version A novel tracking system for the iron foundry field based on deep convolutional neural networks

Journal of Intelligent Manufacturing

Provided in Cooperation with: Springer Nature

Suggested Citation: Beck, Michael; Layh, Michael; Nebauer, Markus; Pinzer, Bernd R. (2022) : A novel tracking system for the iron foundry field based on deep convolutional neural networks, Journal of Intelligent Manufacturing, ISSN 1572-8145, Springer US, New York, NY, Vol. 33, Iss. 7, pp. 2119-2128,

https://doi.org/10.1007/s10845-022-01970-9

This Version is available at: https://hdl.handle.net/10419/307751

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.



WWW.ECONSTOR.EU

https://creativecommons.org/licenses/by/4.0/

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.





A novel tracking system for the iron foundry field based on deep convolutional neural networks

Michael Beck¹ · Michael Layh¹ · Markus Nebauer¹ · Bernd R. Pinzer¹

Received: 10 August 2021 / Accepted: 18 May 2022 / Published online: 2 July 2022 $\ensuremath{\textcircled{}}$ The Author(s) 2022

Abstract

In modern manufacturing the ability of retracing produced components is crucial for quality management and process optimization. Tracking is essential, especially for analyzing the influence of the production parameters on the final quality of the castings. In the iron foundry industry, common marking methods, such as a datamatrix code, cannot be used due to harsh environmental conditions and the rough surface of the cast parts. This work presents a new coding and reading system that guarantees unique marking in the casting process. The coding is built up over several beveled pins and is read out using an optical 2D handheld scanner. With a deep convolutional neural network approach of object detection and classification, a stable image processing algorithm is presented. With a first prototype a reading accuracy of 99.86% for each pin was achieved with an average scanning time of 0.43 s. The presented code is compatible with existing foundry processes, while the handheld scanner is intuitive and reliable. This allows immediate benefits for process optimization.

Keywords Cast part tracking \cdot Digitalization \cdot Deep convolutional neural network \cdot Deep learning \cdot Cast iron \cdot Labeling \cdot handheld Scanner \cdot Code reading

Introduction

The capability of tracing produced parts is a key aspect in modern manufacturing. For a reliable quality control and process optimization, it is necessary to be able to identify individually labelled components to get information about their respective production parameters at any time. This task is accomplished for most industries with standard processes using data matrix codes, bar codes or other established codifications (Ustundag & Cevikcan, 2017).

Tracing produced parts in a foundry is a key aspect as well, but the implementation is much more demanding. Especially for iron sand casting the requirements for a traceable coding unit are very different due to harsh environmental conditions. High temperatures, the mechanical impact on the components when removing the sand, as well as the rough surface itself lead to a challenging task to label every individual cast part. Typically, the process of removing the sand in a foundry (e.g. vibrating conveyor belts, rotary drums) destroys the order in which the parts were produced: all parts from one production batch get mixed, and the relation to their respective casting time stamp is lost. Therefore, it is required to place the label directly during the casting process itself to ensure correct time stamping. Hence, the geometry of the label needs to be formed inside the casting sand mold.

Several solutions for individual labeling of cast parts have been proposed in the literature, but none of them could be widely applicable to iron sand casting. The reason is that there is a wide spectrum on demands that such a system has to fulfill: the sand mould is brittle and the transfer between filigree structures and the metal is limited, mechanical impacts during sand removal can be significant, the placement has to be quick due to cycle times, the casting process should not be modified, safety precautions should not be disproportionally high, and the code should be automatically readable in an easy and reliable way. Beyond that the costs for the placement system and the scanner system need to be economically justifiable. After consulting with an iron foundry company, any codification and the reading system has to work preferably with a handheld scanner: the requirement of being able to

Bernd R. Pinzer bernd.pinzer@hs-kempten.de

¹ Laboratory for Optical 3D Metrology and Computer Vision (3Dvisionlab), University of Applied Sciences Kempten, Bahnhofstraße 61, 87435 Kempten, Germany

scan components of different sizes and at different locations can only be accomplished with such a mobile scanner.

We suggest a novel codification and reading method that satisfies all these requirements. This paper presents a solution with a directly castable codification based on rotational pins and a 2D scanner system which reads the code with the help of deep convolutional neural networks (CNN). The codification consists of multiple beveled pins with a sloped plane (Sandt et al., 2020). The placement of the code into the sand mold can be realized with a stamp unit either inside the model plate or with an external robot for pressing the stamp from the outside into the sand mould. For the reading system a standard 2D camera including an LED light is used. For identification of the code number the image processing algorithm makes use of two different neural networks. The first network is responsible for detecting all pins inside the image. The second network is a classifier to identify the angular position of each pin.

The conceptual and geometric simplicity of the code allow for simple process integration into existing production lines, while the robust reading procedure with handheld mobile scanners is intuitive and does not require significant investments. Once an individual part tracking possibility is established, quality issues can be directly traced back to production parameters, allowing for effective and sustained process optimization in foundries.

Related work

Part-specific traceability is the key for modern quality management (Kopper & Apelian, 2022) and thus intelligent manufacturing. There is a vast body of literature on tracing problems in general; for the special application of tracing cast parts, a very recent overview by Deng et al. (2021) compares different marking methods. Their review article presents a good overview and discussion of published marking methods, which they classify into three categories "simple marking methods", "innovative marking methods", and "advanced marking methods", according to their complexity of application.

However, for the sake of metal sand mould casting, a classification with respect to the production step in which the component is labeled seems useful: the label is placed either immediately during the casting process, or added afterwards ("post-casting").

Post-casting marking techniques like laser marking (Bähr et al., 2004; Landry et al., 2018), pin punching (Atzli, 2008; Bähr et al., 2004), scratch marking (Atzli, 2008; Bähr et al., 2004), or even water jet cutting (Moser, 2012) are prevailing in aluminum high pressure die casting or permanent mold casting. In these processes, the controlled part extraction after the casting ensures a unique relation with the production time stamp. However, these methods are not suitable for iron sand casting, because due to the shake-out process the ordering of the parts gets destroyed and an unambiguous relation to production time is not guaranteed (Bähr et al., 2004).

With the traditional cast clock (Bähr et al., 2004; Meißner & Brahmann, 2009) it is possible to label iron cast parts during casting. Unfortunately the time resolution is very low, since the clock has to be modified manually and is used to statically label all parts of one shift; therefore, the traceability is only down to one work shift.

For *individual* marking during the casting process, several different techniques can be found in the literature. It is possible to use pin punching or a laser to write a number into the sand mold before casting (Knoth, 2009). But because of time issues (cycle time) and the possibility of damaging the sand mold this method is not practical. Also, it is a cost intensive investment and high safety precautions are needed when strong lasers are involved.

The "new cast clock" (Meißner & König, 2011) was invented for permanent mould casting and imprints a data matrix code on the surface of the cast part. An adjustable negative geometry of the code is placed inside the mold and hence the component is labelled during the casting process. The structures of the data matrix code are small (pins with 1.5 mm diameter and height of 1 mm, total code size $24 \text{ mm} \times 24 \text{ mm}$). Such small structures would not form on iron sand casts, because the transfer of features between sand mould and metal surface is much worse in that case. Furthermore, the delicate structures would not survive the mechanical impact during shot blasting. The "new cast clock" offers a range of unique serial numbers of more than 16 Million, which is more than required in the case of iron sand casting. Here, only a few thousand parts have to be distinguished in one batch. Reading the code of the "new cast clock" requires a special 3D camera.

Freytag et al. (2013) presented a method for high pressure die casting. A small magnetic plate, that had been treated by a laser in order to show a spatial magnetic flux distribution reminiscent of a data matrix code, was positioned right below the surface of the component during casting. The code could be read afterwards with a magnetic field sensor. A similar approach is the insertion of an RFID chip inside a metal part (Battini et al., 2009; Ranasinghe et al., 2004). A proof of concept was reported for aluminum high pressure casting by Fraunhofer IFAM (Institute for manufacturing technology and advanced materials) (Pille & Rahn, 2017), where the chip was specially protected against the high temperature during the casting process. Both these techniques modify the mechanical properties of the cast part by introducing a foreign material, which could be undesirable in certain situations. Furthermore, the insertion of additional parts requires a special tool-technology and an adaptation of the casting

process, and the relatively high costs for each cast part limit the wide spread application.

In summary, most practical solutions have been developed for aluminum high pressure die casting or permanent mold casting and not for the case of iron sand mold casting. Thus it appears crucial to investigate a new method for traceability of iron sand mold casting.

Methods

In the following sections all the essential techniques to ensure a solid and reliable reading system are explained in detail. First, we elaborate on the code design ("Code design and reading principle" section), which is motivated by the specific boundary conditions of metal sand casting.

Optical code reading requires first the detection of the pins on the surface and, as a second step, determination of the orientation of the individual pins. Classical image processing with a shape-from-shading illumination approach was tested for this goal, but was found not to be robust enough in the industrial setting. A solution based on 3D point clouds of the surface was successfully implemented and tested (Sandt et al., 2020), but required a significant invest in a 3D scanning device. Based on our experience, we built a hand scanner ("Handheld scanner design" section) and developed a code reading algorithm with convolutional neural networks (CNN), which can be deployed on low cost hand held devices. This approach is described in "Pin detection and code orientation" and "Classification network" sections.

Code design and reading principle

Based on the special conditions in the foundry field and in particular in the iron sand casting, the coding unit must meet certain boundary conditions. The code must be easily moldable, robust against mechanical influences, visually readable and, in addition, have a sufficiently high data density. To meet these requirements, the cast part code is built-up by a set of individually orientated pins. Each pin is cylindrical in shape with a partially sloped cylinder cover as shown in Fig. 1a. In a first prototype each pin has a diameter of 6 mm, a height of 1.1 mm and a sloped plane with an angle of 20°. Depending on the casting quality and the component size, the geometries can be selected accordingly. Due to the very simple geometry, which does not have any filigree structures, robustness and moldability is guaranteed. In addition, the structure allows both a 3D (Sandt et al., 2020) and a 2D scanning procedure with a high data density. For the 2D optical readout process the structure enables two different contrast-rich features when using a coaxial top illumination. The features are a shadow, in the shape of a dark semicircle, and the top side of the pin which is divided into a lighter and a darker area due



(c) Pin orientations and orientation vector $(V_{\rm or})$

Fig. 1 Pin layout for a code with 3 pins

(d) Example of a real code on an iron cast part

to the sloped plane. These features are used to identify the orientation of the pins.

The basic principle is based on the idea of coding the component number via different orientations of the pins. The pin direction is defined with the normal vector of the pin sloped plane which is projected into the base plane (see Fig. 1c). For reference, an orientation vector V_{or} based on the entire pin arrangement is determined. As a consequence, a non-symmetrical layout needs to be provided. In our case it was an L-shaped layout where the rotational symmetry is only at 360°. By overcoming this constraint other layouts are possible and can for example be adjusted for different part geometries. Another example of a longer shaped code is shown in Fig. 1b. Additionally, an example of a real code on an iron cast part is show in Fig. 1d.

For a first prototype, the angular differentiation was defined with an angle of 22.5° which leads to 16 possible rotational positions for each pin. With a total number of 3 pins 4096 (16³) individual part numbers (*N*) can be generated. This number rises exponentially with an increase of the number of pins. With a code of 4 pins it is possible to generate 65,536 (16⁴) individual combinations.

For calculating a final part number N it is necessary to make an explicit pin number definition. In the example of the L-shape the pin numbering is shown in Fig. 1c. Based on that configuration the part number could be calculated with Eq. 1, where o_n is the orientation position of the appropriate pin as specified in Table 1.

$$N = o_1 \cdot 16^0 + o_2 \cdot 16^1 + o_3 \cdot 16^2 \tag{1}$$

 Table 1
 Angle values for pin positions

Angle [°]	Orientation position []
0 ± 11.25	0
22.5 ± 11.25	1
45 ± 11.25	2
337.5 ± 11.25	15

Handheld scanner design

In Fig. 2 a first prototype with a 3D printed chassis is shown. The scanner system includes a 2D camera as well as a blue LED light. The camera is an IDS UI313xCP-M monochrome camera with a resolution of 800 by 600 pixel. The lens has a focal length of 16 mm and is used with a blue filter. Together with a powerful 700 mA blue LED the system is not affected by ambient light. Additionally, a trigger button was added for activating the scanning process and to switch on the LED light. This first prototype was connected to a laptop through cables, which processed the images and displayed the live image and the results. All cables were guided through the handle bar to the back of the scanner. The design was kept as simple as possible, with the 3D printing process in mind.

Pin detection and code orientation

For locating every pin inside the image, an object detection algorithm was needed. This goal was achieved with the use of the programming language Python, OpenCV and the Tensorflow object detection API (TODA) (Huang et al., 2018). TODA is a commonly used open source high level API framework which is built on top of tensorflow. It already comes with a bundle of pretrained models for object detection with different detection methods such as faster R-CNN (Ren et al., 2015) or a single shot multibox detector (SSD) (Liu et al., 2016). These pretrained models can be used for transfer learning. For reasons of performance (Shanmugamani, 2018) an SSD model (ssd_mobilenet_v1_coco_11_06_2017) was used which was originally trained on the COCO dataset (Lin



Fig. 2 CAD model of the handheld scanner

et al., 2014). COCO stands for "common objects in context" and consists of labelled images of complex everyday scenes containing objects like cars, animals or people. To identify pins instead of these objects inside the images a stack of 250 labelled pin images was created by manually drawing quadratic bounding boxes around the pins using an open source software called "labelImg" (Tzutalin, 2015). With the information of the bounding boxes and the use of TODA, the model could be retrained to recognize iron cast pins. We determined the general mean average precision (mAP) of the trained model to be 47.32 for 50 test images. Although there were many false positive detections, we could filter out the irrelevant detections by keeping only the three bounding boxes with the highest confidence level. With this prior knowledge as well as knowledge about the approximate distance of the pins, we never observed any misdetections in the test data and later in the experiments. In Fig. 3a an example of detected pins and their bounding boxes is shown.

For the following steps it was important to assign each pin to its explicit pin number as well as identifying the orientation vector $V_{\rm or}$ of the code as described in "Code design and reading principle" section. Based on the center point coordinates of all pin bounding boxes $P_{\rm n}$ the center of mass $P_{\rm c}$ of the ensemble can be calculated (see Fig. 3b). By computing the distances between P_n and P_c pin number 1 (P_1) could be uniquely identified by its smallest distance. Finally the angles between vector $\overrightarrow{P_1P_c}$ and $\overrightarrow{P_1P_n}$ (α , β) were calculated. Sorting the angles according to size, it was possible to assign pin number 2 (P_2) and 3 (P_3) . The most convenient way for identifying the orientation vector is the use of the vector between the center point of pin 1 and 2. In case of an inaccurate pin detection an exact orientation vector is not ensured. For a more precise identification a perfect L-shape was fitted into the coordinates of all midpoints (see Fig. 3ared L-arrow). Hence the total error is compensated for in case of inaccurate detection.

To recognize the rotational position of each pin with the classification CNN, it was necessary to rotate the image to



(a) Bounding boxes as a result of TODA and L-fit

Fig. 3 Pin detection and numbering



(b) Assignment of each explicit pin number

Fig. 4 Pin rotation to target vector

a reference position with the offset angle γ (Eq. 2). The definition of that target vector V_{tar} is $(0,1)^T$, which means that it is a vector in a vertical direction. In Fig. 4 an example of the rotating process is shown.

$$\gamma = \arccos\left(\frac{V_{\rm or} \circ V_{\rm tar}}{\mid V_{\rm or} \mid \cdot \mid V_{\rm tar} \mid}\right) \tag{2}$$

The offset angle γ is approximately 75° in the example shown. The image is rotated according to the offset angle and the center points of the bounding boxes are transformed accordingly. Due to the circled geometry each pin can be cropped with its original bounding box size in the rotated image. As a result, the final pin images are produced which are independent from any prior code orientation. These three images needed to be classified individually for their rotational position with the classification CNN network.

Classification network

To identify the rotational position of each pin, a custom CNN was constructed. Figure 5 shows a prototypical layout; the hyperparameters of the network are optimized in "Identifying the best hyper-parameters" section. In contrast to the object detection network (see "Pin detection and code orientation" section), the classification network was not based on a pre-trained network. Therefore, no transfer learning was used. The network was built from scratch with python based on tensorflow and keras as a backend.

Generating training and test data

Due to 16 possible orientations of the pin, the network needed to be trained on 16 classes. In order to generate training as well as test data, 41 test plates—each containing 16 codes—were cast under realistic environmental conditions. We observed no detrimental mechanical deformation on any of the codes coming from the "shot blasting" procedure. The plates were separated into 31 training plates and 10 testing plates. In order to create the data sets for training and testing, the codes were located and oriented using the pin detection algorithm ("Pin detection and code orientation" section). With an image of a three-pin-code, the rotational information and the corresponding bounding boxes, every



Fig.5 Prototypical CNN for classification of the individual pins: starting with an input image of size 100×100 , there are three convolutional layers with two max pooling layers in between. The number of feature maps is 32 after the first convolutional layer, increases to 64 after the second convolutional layer, and is kept to 64 after the third convolutional layer. Once the feature maps are flattened, two fully connected layers (64,16) complete the classification network. Hyper parameters like number of convolutional layers, number of feature maps, and size of dense layers are optimized in "Identifying the best hyper-parameters" section



Fig. 6 Example of pin classes training image

pin could be rotated back to its zero position. Based on that fundamental image all 16 rotational positions could be created (data augmentation). Since the individual pin images were cropped from a larger image after rotation, we avoided the problem of undefined gray values in the corners of the pin images, which would typically occur when rotating images. Furthermore, it was possible to create even more data by mirroring the zero positioned pin image and repeating the rotation procedure for the 16 positions. Hence, with a single code consisting of three pins it was possible to produce 6 images per class and thus $3 \times 2 \times 16 = 96$ images in total. In Fig. 6 an example of a rotated pin for different classes is shown. All shown pin images are based on a single pin from a code image.

With 31 training plates (n_{pl}) , 16 codes for each plate (n_c) , 3 pins per code (n_p) and the mirroring factor of 2 (m) it was possible to generate 2976 individual pin images for each class (see Eq. 3). For reasons of variation, multiple images of the same code were taken, but the orientation of the code varied and therefore the light reflection changed. The camera angle was set to 0° (View direction 1) (see Fig. 7). Based on the

fact that the scanner is handheld the acquisition angle of the camera needs variation as well. For that reason, images with an acquisition angle of 5° (View direction 2) and 10° (View direction 3) were added. Altogether a total number of 12900 training images per class were available, of which each view direction consisted of 4300 images. With 10 testing plates it was possible to create 960 individual test images



Fig. 7 Different viewing angles of the camera

per class. Furthermore, test images were created for all view directions as well, which led to a total number of 2880 testing images per class. The test images constituted 22.33% of the training data.

$$n_{\rm img} = (n_{\rm pl} \cdot n_{\rm c} \cdot n_{\rm p}) \cdot m \tag{3}$$

Total reading process

To improve the accuracy of the reading system, it runs with a real time image stream while multiple confidence thresholds are implemented. The reading process can be compared to reading a QR-Code. The system automatically stops the reading process when all quality requirements are accomplished. Figure 8 gives an overview of the total reading process and Fig. 9 shows the handheld scanner and GUI for displaying the live image.

The first confidence threshold is applied right after the pin detection. If the confidence level is lower than 50% for every pin and if there are not exactly three pins detected the evaluation stops and the next frame will be grabbed. If all requirements are achieved the pins are numbered specifically and cropped as described in "Methods" section. The final pin images are individually classified with the second CNN. The categorization of the network is only accepted if the confidence level is above a specific value. The required confidence level can be individually adjusted to either improve speed or accuracy performance. If all 3 pin orientations are classified with the required confidence value, the final code number can be calculated like described in "Code design and reading principle" section. For increased reliability it is possible to set a minimum count for consistent repetitive classification.

For an example of a code number calculation, consider the code image from Fig. 4. After the code has been rotated to normal position and the individual pins have been extracted (rightmost column), the orientation of each pin has to be classified according to the angular ranges defined in Table 1. The projection of the normal vector of the inclined ramp of pin number one points horizontally to the left, corresponding to an angle of 180° or class number 8, respectively. The same vector for pin number two points horizontally to the right, corresponding to an angle of 0° or class number 0. The angular orientation of pin number three is 22.5° or class



Fig. 8 Flow chart of total reading process

number 1. The resulting code number for this image would be, according to Eq. 1, $8 \cdot 16^0 + 0 \cdot 16^1 + 1 \cdot 16^2 = 264$.

Experiments and results

In this section the focus was on the optimization of the classification network. In the first step, the impact of training data variation was analyzed, whereas in the second step the impact of the network structure was reviewed. Furthermore, the total



Fig. 9 Handheld scanner with GUI

reading process with the handheld scanner was validated on all testing plates.

Identifying the best combination of training data

In the first step we used a simple network structure (see Fig. 5) and analyzed the impact of the mixing ratio between the different view directions for the training process. With 4300 images per class for each view direction, all mixing ratios were analyzed with a step width of 5%, which refers to the image number. For comparability, each mixing ratio had a total of 4300 images. This lead to 231 mixing ratio combinations. In Table 2 a few examples are shown for different ratios and the appropriate image number. The network was trained with all different ratio combinations and was tested on the full test dataset (see "Generating training and test data" section). The result of all trained models is visualized in Fig. 10. The x- and y-axes show view direction 2 and 3 ratios, whereas the view direction 1 ratio is indicated inside the plot with contour lines. The color plot is linearly interpolated between measured points. With a ratio combination of 25% view direction 1, 5% view direction 2 and 70% view direction 3 (25/5/70) training images, the highest score of 98.3%was accomplished. The lowest score was 92.34% with a ratio of 80/20/0 images. It is apparent that a high quantity (greater 60%) of view direction 1 and a low quantity (smaller 20%) of view direction 3 images have a negative impact on the validation accuracy. There is also a slight tendency toward improved scores with a ratio greater than 50% of view direction 3 images.

Identifying the best hyper-parameters

With the information of the best mixing ratio combination the impact of the hyper-parameters could be analyzed. Due to a high possible quantity of parameters this test focused particularly on the convolution and dense layer structure (i.e.

 Table 2
 Examples of mixing ratio between view direction 1, 2 and 3 images (The number of images refers to one class)

	View dir. 1	View dir. 2	View dir. 3	Sum
Ratio	0%	5%	95%	100%
Images	0	215	4085	4300
Ratio	0%	50%	50%	100%
Images	0	2150	2150	4300
Ratio	80%	10%	10%	100%
Images	3440	430	430	4300



Fig. 10 Validation result of all ratio combinations

number of hidden layers and neurons). Furthermore the different structures were tested with a 3 by 3 and a 9 by 9 convolution kernel. Every model had a max-pooling layer with a kernel size of 2 by 2 pixels between each convolutional layer. All different model structures were trained with the "Adam" optimizer, a maximum of 40 epochs, a batch size of 256 and a learning rate of 0.001. Additionally there was an early stop implemented, which stopped the training if the validation accuracy did not change any more (patience = 6). Due to the fact that the accuracy values vary with repeated training under the same conditions, the training was performed five times for each architecture. All different model structures and their average validation accuracy values with the appropriate distributions are shown in Table 3. The tolerances are based on the T-distribution and a confidence interval of 95%. In addition, the results of the table are shown visually in Fig. 11, where the y-axes displays the accuracy and the xaxes the index number of the network structure (see Table 3). The graph illustrates an explicit improvement of the accuracy level with increasing the network structure's depth, even considering the variance. It must be added that the most complex model was not trainable with the 9 by 9 kernel.

Model number 6 achieved the greatest average validation accuracy score with 99.07% and a low variance with 0.10%.

Nr.	Parameter	Value	
1	CNN Feature maps	2, 8	
	CNN Kernel size	3×3	
	Dense layers	128	
	Average score	$96.50 \pm 0.57\%$	
	Ranking	10	
2	CNN feature maps	2, 16	
	CNN kernel size	3×3	
	Dense layers	128	
	Score	$\textbf{96.43} \pm 0.52\%$	
	Ranking	11	
3	CNN feature maps	32, 64	
	CNN kernel size	3×3	
	Dense layers	256	
	Score	$97.67 \pm 0.21\%$	
	Ranking	7	
4	CNN feature maps	32, 64, 128	
	CNN kernel size	3×3	
	Dense layers	256, 128, 64	
	Score	$98.70 \pm 0.09\%$	
	Ranking	4	
5	CNN feature maps	32, 64, 128, 256	
	CNN kernel size	3×3	
	Dense layers	1024, 526, 128, 64	
	Score	$99.04 \pm 0.18\%$	
	Ranking	2	
6	CNN feature maps	32, 64, 128, 256, 512	
	CNN kernel size	3×3	
	Dense layers	2049, 1024, 526, 128, 64	
	Score	$99.07 \pm 0.10\%$	
	Ranking	1	
7	CNN feature maps	2, 8	
	CNN kernel size	9 x 9	
	Dense layers	128	
	Score	$97.12 \pm 0.35\%$	
	Ranking	9	
8	CNN feature maps	2, 16	
	CNN kernel size	9 x 9	
	Dense layers	128	
	Score	$97.43 \pm 0.30\%$	
	Ranking	8	
9	CNN feature maps	32, 64	
	CNN kernel size	9 x 9	
	Dense layers	256	
	Score	$98.07 \pm 0.30\%$	
	Ranking	6	
10	CNN feature maps	32, 64, 128	

 Table 3
 Comparison of different network structure and their validation accuracy for the complete test data

Table 3 continued

Nr.	Parameter	Value
	CNN kernel size	9 x 9
	Dense layers	256, 128, 64
	Score	$\textbf{98.70} \pm 0.24\%$
	Ranking	5
11	CNN feature maps	32, 64, 128, 256
	CNN kernel size	9 x 9
	Dense layers	1024, 256, 128, 64
	Score	$\textbf{99.00} \pm 0.20\%$
	Ranking	3



Fig. 11 Errorbar plot of different model structures based on Tdistribution

This model structure will be considered for further investigations.

For identifying the impact of best ratio combination (see "Identifying the best combination of training data" section) it was necessary to have all combinations at the same image quantity (4300 images per class in total) to ensure comparability. Having the best ratio combination and the best network structure, it is possible to retrain the network with the highest number of training images, but keeping the same ratio. The new combination includes 1535 of view direction 1 images, 307 of view direction 2 images and 4300 of view direction 3 images for each class, which adds up to a total number of 6142 images per class. With the new training images a model with a validation score of 99.28% was accomplished.

Analysis of confidence threshold

For a more reliable reading system a confidence threshold was implemented as described in "Total reading process" section. With the best trained model (see "Identifying the best hyper-parameters" section) the impact of such a threshold value was analyzed. With the increasing confidence thresh-



Fig. 12 Correlation between data efficiency, accuracy and confidence threshold

old, the likelihood of a correct classification also increased, meanwhile the data efficiency declined, because not all images provided the necessary confidence level. Consequently, with an increasing confidence value, the scanning time possibly rises, because the user needs to move the camera to get a satisfying image. The correlation between accuracy and data efficiency is shown in Fig. 12. This graph is based on the test dataset including all view directions. The y-axes on the left side represents the data efficiency for each confidence threshold whereas the y-axes on the right side represents the accuracy score. Apparently the lowest efficiency level is over 90%, which is why a confidence threshold of 99% for the total reading process is acceptable.

Validation of the complete system

For a final validation all available test casting plates were measured with the handheld scanner. The confidence threshold was set to 99% (see "Analysis of confidence threshold" section) and the minimum count for consistent repetitive classification to a value of 2. Because the code can be built with multiple pin numbers, the validation accuracy is referenced to every single pin. Hence for every pin quantity $n_{\rm Pin}$ a reading probability P_{Total} can be calculated with Eq. 4 based on the resulting Pinscore P_{Pin} . For validation every single code on the 10 test casting plates was scanned three times. Moreover the validation process was executed with two different test persons. The first test person (Tp_1) had used the handheld scanner a number of times before the test, whereas the second test person (Tp_2) was a first time user. Table 4 shows the result for both test persons. It is noteworthy that the Pinscore for Tp₁ with 99.86% was only by 0.69% higher than the Pinscore of Tp_2 with 98.96%. This indicates a stable reading process based on minimal difference. Additionally, the minimum and maximum position difference for an incorrectly detected pin for both is only 1. The average scanning time for Tp₁ was 0.43 s, whereas 90.21% of all scanning times were lower than 0.5 s. For Tp₂ the average scanning

Table 4 Validation of complete system by experienced person (Tp_1) and a first time user (Tp_2)

Parameter	Tp_1	Tp ₂
Pinscore (P _{Pin})	99.86 %	99 .17%
Score 3 pins (P _{Total})	99.58%	98.13%
Min. position difference	1	1
Max. position difference	1	1
Av. scanning time	0.43 s	0.93 s
Perc. of scans under 0.5 s	90.21%	70.00%

time was higher with 0.93 s and the fraction under 0.5 s was 70.00%.

$$P_{\text{Total}} = (P_{\text{Pin}})^{n_{\text{Pin}}} \tag{4}$$

Conclusion

A simple to use and reliable handheld reading system for a custom developed cast part marking system, which allows for identifying individual components after casting, was presented. The design of the pins inside the code ensures robustness against mechanical impact and allows integration inside the casting process.

Based on deep convolutional networks and the implementation of the confidence threshold, a high accuracy of pin classification with 99.86% (Tp₁)/99.17% (Tp₂) for the test plates was accomplished. The complete system, including pin detection and classification, works stably. Scanning is intuitive and in most cases (90.21% for an experienced test person Tp₁ and 70.00% for a first time user Tp₂, respectively) takes less than half a second. Due to the automatic stop the user cannot make any mistakes. In our experiments, a person with more scanning experience accomplishes faster scanning times, due to the fact that, in our test-setup, the screen with the camera image was not attached to the scanner itself, which made the scanning in the beginning a little challenging.

While working reliably in our testing environment, different conditions in the industrial environment could have an influence on the reading outcome. For example, different material compositions would result in changes of surface topologies, which would require a fine tuning of the object detection and classification algorithm. By extending the training data set with images from different materials, the object detector as well as the classifier could take these varying conditions into account. The same holds true if the surface of the parts needs to be painted afterwards, or requires further processing. However, in practice, the presented code could even be replaced with a more convenient post-casting marking technique (e.g. laser applied data matrix code) in a subsequent processing step. The advantage of the presented code is that it could bridge the gap between pouring and post processing of the cast parts.

Compared with any other solution for retracing iron sand cast parts, it is now possible to label every single component and to have a very mobile and cheap reading system. With a housing for the camera and the LED light, the scanner can be protected from the harsh and dirty environment inside a foundry. The reading software could be transferred from a tethered to a completely self-sufficient, low power system on a module (e.g. NVIDIA[®]'s Jetson Nano).

Future research will be directed towards the application of the mobile scanner in industrial settings, including a protocol to tune the algorithm to variations in surface topologies. We would like to include the marking and scanning system in a prototype application for data driven process optimization, where extensive production parameters can be related to quality parameters of the cast parts. Correlating quality parameters with production parameters could pave the way for a widespread application of artificial intelligence in iron foundries.

Funding Open Access funding enabled and organized by Projekt DEAL. This study was supported by the European Regional Development Fund, project number RvS-SG20-3451-1/26/31 ("network efficient production technology").

Declarations

Conflict of interest The authors declared that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecomm ons.org/licenses/by/4.0/.

References

- Atzli, C. (2008). Aus der Industrie Neues von der Montanuniversität - Personalnachrichten - Tagungsankündigungen. BHM Berg- und Hüttenmännische Monatshefte, 153(9), 356–367. https://doi.org/ 10.1007/s00501-008-0410-5
- Bähr, R., Ernst, W., Schütze, O., & Winter, J. (2004). Teilspezifische Kennzeichnung von Gussstücken. *Giesserei*, 91(06), 40–48.
- Battini, D., Faccio, M., Persona, A., & Sgarbossa, F. (2009). A new methodological framework to implement an RFID project and its application. *International Journal of RF Technologies: Research and Applications*, 1(1), 77–94. https://doi.org/10.1080/ 17545730802320174
- Deng, F., Li, R., Klan, S., & Volk, W. (2021). Comparative evaluation of marking methods on cast parts of Al–Si alloy with image pro-

cessing. International Journal of Metalcasting. https://doi.org/10. 1007/s40962-021-00661-0

- Freytag, P., Kerber, K., & Bach, F. W. (2013). Laserbeschriftung von Hartferritmagneten zur Kennzeichnung von Druckgussteilen. *Forschung im Ingenieurwesen*, 77(1–2), 39–47. https://doi.org/10. 1007/s10010-013-0161-7
- Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. (2018). Tensorflow object detection API. https://github.com/tensorflow/ models/tree/master/research/object_detection
- Knoth, G. (2009). Gravuren im Sand. Giesserei-Erfahrungsaustausch, 06, 14–17.
- Kopper, A. E., & Apelian, D. (2022). Predicting quality of castings via supervised learning method. *International Journal of Metalcasting*, 16(1), 93–105. https://doi.org/10.1007/s40962-021-00606-7
- Landry, J., Maltais, J., Deschênes, J. M., Petro, M., Godmaire, X., & Fraser, A. (2018). Inline integration of Shotblast resistant laser marking in a die cast cell. In *Die casting congress & exposition*. North American Die Casting Association. https://www.semanticscholar.org/paper/Inline-Integration-of-Shotblast-Resistant-Laser-in-Landry-Maltais/ 258897a89efb611178e3996ef9c612cc40652efc
- Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision—ECCV* (pp. 740–755). Springer. https:// doi.org/10.1007/978-3-319-10602-1_48
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In: B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision—ECCV 2016* (pp. 21–37). Springer. https://doi.org/ 10.1007/978-3-319-46448-0_2.
- Meißner, K., & Brahmann, M. (2009). Verfahren zur markierung von gussteilen während des urformprozesses. *Giesserei*, 96(6), 52–61.
- Meißner, K., & König, B. (2011). Spektrum-die neue gie
 ßuhrvollflexibles markiersystem f
 ür gussteile. *Giesserei*, 98(6), 108.
- Moser, J. (2012). Wasserstrahlschneiden ein vielseitiges bearbeitungsverfahren. *Giesserei-Erfahrungsaustausch* 5+6 (pp. 14– 16).
- Pille, C., & Rahn, T. (2017). RFID bringt Giessereien in die Industrie 4.0. *RFID im Blick*, 03, 36–38.
- Ranasinghe, D. C., Hall, D. M., Cole, P. H., & Engels, D. W. (2004). An embedded UHF RFID label antenna for tagging metallic objects. In Proceedings of the 2004 intelligent sensors, sensor networks and information processing conference, ISSNIP '04 (pp. 343–347). https://doi.org/10.1109/issnip.2004.1417486
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn. Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), Advances in neural information processing systems. Curran Associates, Inc. https://doi.org/10.4324/9780080519340-12
- Sandt, M., Beck, M., Linkerhägner, F., Hartmann, D., Layh, M., & Pinzer, B. R. (2020). CastCode - Gussteilrückverfolgbarkeit an automatischen Formanlagen. *Giesserei Special*, 1, 30–39.
- Shanmugamani, R. (2018). Deep learning for computer vision: Expert techniques to train advanced neural networks using TensorFlow and Keras. Packt Publishing.
- Tzutalin. (2015). LabelImg. https://github.com/tzutalin/labelImg
- Ustundag, A., & Cevikcan, E. (2017). *Industry 4.0: Managing the digital transformation*. Springer.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.