Inspection of sandblasting defect in investment castings by deep convolutional neural network

Jenn-Kun Kuo1,2 · Jun-Jia Wu3 · Pei-Hsing Huang2,∗ · Chin-Yi Cheng3

Received: 7 December 2021 / Accepted: 27 January 2022 / Published online: 23 February 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract
Investment castings often have surface impurities, and pieces of shell moulds can remain on the surface after sandblasting. Identification of defects involves time-consuming manual inspections in working environments of high noise and poor air quality. To reduce labour costs and increase the health and safety of employees, automated optical inspection (AOI) combined with a deep learning framework based on convolutional neural networks (CNNs) was applied for the detection of sandblasting defects. Four classic CNN models, including AlexNet, VGG-16, GoogLeNet, and ResNet-34, were applied for training and predictive classification. A comprehensive comparison reveals that AlexNet, VGG-16, and GoogLeNet v1 could accurately determine whether there were defects. Among the four models, AlexNet and VGG-16 were the most accurate, with prediction accuracy of 99.53% and 99.07% for qualifying products and both 100% for defective products. GoogLeNet v4 and ResNet-34 did not perform as expected in defect prediction. The reasoning behind the poor performance of GoogLeNet v4 and ResNet-34 is attributed to the restrictedness of the investment casting dataset to use models with residual learning architectures. Finally, a direct detection technique based on the AOI and CNN structure with a fast and flexible computational interface was demonstrated.

Keywords Automated optical inspection · Investment castings · Sandblasting defects · Convolutional neural network · Deep learning

1 Introduction

The casting industry in Taiwan is immense, within a working environment that is often hazardous. Lost-wax casting involves high temperatures, high levels of noise and dust, and a significant amount of environmental pollution. In particular, at the sandblasting stage of this process, workers must manually inspect workpiece surfaces for impurities or remaining shell moulds. This is time-consuming and can result in eye fatigue, which ultimately affects the quality of inspection. Prolonged exposure to noise and poor air quality during the inspection process also undermines the health of workers.

Automated optical inspection (AOI) technology has matured in recent years. It uses optical instruments and image processing to detect product defects. It achieves non-contact detection with greater stability, speed, and accuracy than manual detection, as well as reducing production costs. Many industries are using AOI for quality inspection of processes such as laser welding of cell phone batteries [1], flat steel manufacturing [2], automotive manufacturing, and semiconductor packaging. Furthermore, recent advances in computer hardware technology have lowered storage costs and enhanced computing power. As a result, deep learning (DL) techniques have become popular. A number of network architectures have been developed, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and restricted Boltzmann machines (RBMs). Among these, CNNs offer good performance when applied to image recognition and defect detection, successfully solving problems that conventional image processing techniques cannot, such
as quality detection in agriculture [3], bridge deck assessment, and railway track inspection [4].

Conventional defect detection is based on image processing, involving grayscale preprocessing, feature extraction, and then comparison. Researchers have employed grayscale histogram techniques as well as texture analysis and wavelet transformation for defect detection. For instance, Kuo et al. [5] applied a grayscale histogram technique to detecting defects in LED packaging. They obtained good recognition rates and helped to reduce the errors caused by manual inspection, thereby increasing product yield and quality. Li et al. [6] used X-rays and a wavelet transformation technique to detect and assess defects such as cracks, gas holes, and impurities in castings. Their approach successfully detects defects for most cases, but issues remain, such as the need to manually confirm the number of multiresolution levels for each image. The features of grayscale images of steel billets vary; Jeon et al. [7] therefore proposed a method based on wavelet transformation to detect defects on steel billets. Their approach could effectively detect defects such as cracks on steel billet surfaces. However, using conventional image processing techniques such as grayscale histograms or texture analysis alone can result in errors, as these detection processes are susceptible to interferences, causing instability. Moreover, they often can only recognize a single defect. These approaches also require substantial amounts of time and effort to characterize and define defects, making them inadequate for current needs.

To address these issues, machine learning and DL have been incorporated into defect detection. Past applications of machine learning include that developed by Bakir [8], in which logistic regression and a decision tree were used to identify the process variables of investment castings and the corresponding values that result in product defects. Bakir demonstrated that decision trees are superior to logistic regression, with accuracy rates of 92.15% and 60.3%, respectively. Gu et al. [9] solved a wood defect classification problem using support vector machines (SVMs) and derived a mean classification rate of 96.5% and a false alarm rate of 2.25% from 400 test datasets. Li et al. [10] proposed a fabric defect detection method in which the algorithm extracts the grayscale histogram and engages in defect training using SVMs. Their algorithm was shown to be superior to other methods such as the two-way visual attention mechanism (TVAM) and dictionary-based visual saliency (DVS). For continuous casting quality prediction, Ye et al. [11] developed an approach based on weighted random forests (WRFs) and compared it with a decision tree and an SVM. Their results indicated that the mean true positive rate of WRF prediction was 93%, which was higher than both the 68% of the decision tree and the 65% of the SVM. To detect casting surface defects such as pores, pinholes, and cracks, Riaz et al. [12] passed images through a Gaussian filter to smooth them and then detected and classified defects in the images using k-means. Past applications of DL to defect detection include that by Alencastre-Miranda et al. [3], in which four classic types of CNNs (AlexNet, VGG-16, GoogLeNet, and ResNet-101) were used to assess the surfaces of sugarcane billets. Their results indicated that AlexNet had the best prediction performance; depending on the sugarcane variety, their approach increased yield per hectare from 33 to 80%. Dorafoshan and Azari [4] compared a one-dimensional DL model (i.e., biLSTM) with two-dimensional DL models (namely AlexNet, GoogLeNet, and ResNet-101) for bridge deck assessment. The results revealed that the one-dimensional model had the best average true positive rate at 70%, and the lowest was the ResNet at 53%. On the whole, the one-dimensional model was more accurate than the two-dimensional models because the input of the former comprised signals rather than images. Iyer et al. [13] proposed a railway track inspection system and compared the performances of an artificial neural network (ANN), a CNN, a random forest, and an SVM. Their results indicated that the CNN performed better than the other algorithms. Li et al. [14] used a you-only-look-once (YOLO) algorithm to detect defects on steel strip surfaces. Their results showed that the mean average precision (mAP) for six types of defects was 97.55%, and at 83 frames per second (FPS), it could achieve 99% detection accuracy. Raj et al. [15] developed a graphical user interface using YOLO and applied it to detect casting surface defects such as pinholes, burrs, shrinkage defects, mould material defects, casting metal defects, and metallurgical defects. Their approach achieved an accuracy rate of 99% when applied to classifying investment castings in the test dataset. Shi et al. [16] proposed an algorithm based on a single shot object detector (SSDT), a modified single shot multibox detector (SSD) algorithm, to detect tiny defects in printed circuit boards (PCBs). Their approach achieved good performance with an mAP of 81.3%, which was better than SSD (mAP = 79.5%). Du et al. [17] developed a defect detection system for aluminium castings based on X-ray oriented DL. They incorporated a multiscale feature pyramid network (FPN) and RoIAlign into Faster-RCNN to strengthen information from bottom structures. Their results showed that using FPN or RoIAlign to detect defects in X-ray images of aluminium castings achieved better performance than Faster-RCNN.

Existing studies have rarely focused on the detection of impurities or remnants of shell moulds on the surface of investment castings after sandblasting; most studies investigated the detection of pores, pinholes, and cracks associated with the casting process [6, 11, 12, 15]. To protect the eyesight of workers and reduce their exposure to environments with high levels of noise and poor air quality, AOI based on the package of open-source computer vision library (OpenCV library [18]) was employed to develop
defect-detection software for lost-wax investment castings. However, different castings and surface properties require changes to the underlying algorithm, thereby reducing detection compatibility. Thus, the AOI technology was paired with DL techniques [19–24], including AlexNet [20], VGG-16 [21], GoogLeNet v1 [22], GoogLeNet v4 [23], and ResNet-34 [24] models, to promote the accuracy and flexibility of defect-detection software. Furthermore, a CNN-based object detection algorithm, i.e., YOLO v3 [25], was used to speed up position detection and search for castings. A detailed introduction for aforementioned DL techniques and CNN-based algorithms are given in Sect. 2.

2 Research procedures and methods

The pre-processing of DL involves the classification and labelling of training data. Detailed data collection, compressed-file establishment, and labelling of the dataset were conducted as follows:

Step 1. Camera setup and image capture
The camera function of a smartphone was used to capture images of investment castings. To obtain images of equal size, the smartphone was installed on a tripod at a fixed height \( h = 130 \text{ cm} \), as shown in Fig. 1a. Furthermore, to reduce the number of images captured and to capture individual samples completely, white paper with 6 \( \times \) 4 grids lined in black was used as the background. The investment castings were placed in the grids, and a cropped image of each grid served as training data (or testing data), as shown in Fig. 1b. In consideration of the fact that sandblasting inspection areas mostly have fluorescent lights and to enhance the environmental compatibility of this study, the images of training data were taken under fluorescent lights in the factory. This set-up was used to photograph 16 types of investment castings. Around 100 images such as those shown in Fig. 1b were obtained.

Step 2. Cropping regions of interest (ROI)
The original images included the floor background and 24 white grids. To effectively extract the ROI within the grids, the OpenCV library [18] was employed for grayscale processing, Gaussian blurring, edge detection for ROI identification, and the cropping and saving of individual sample images. The size of the ROIs was set at 416 \( \times \) 416 pixels. Using edge detection, rectangles with a certain width and length were selected, and using the coordinates of the upper left corner, the coordinates of the rectangle centres were calculated as shown in Fig. 2a. Next, using the coordinates of the centres and the target width and length, the coordinates of the upper left corner were calculated as shown in Fig. 2b. Once the rectangle coordinates were revised, the target rectangle was cropped. In total, 1591 cropped images were obtained. Some of these are exhibited in Fig. 3.

Step 3. Image labelling
Before training, the images must be labelled. This provides samples for training, which the CNNs must learn to be able to classify and recognize un-labelled data. The investment casting images were divided into two groups based on whether they had undergone sandblasting: (1)
those in which there were clearly visible remnants of shell moulds or impurities on the surfaces of the investment castings were labelled as “unqualified” (i.e., defective); (2) those in which the investment castings had undergone one to several rounds of acid pickling and sandblasting until there were no traces of shell moulds or impurities on their surfaces were labelled as “qualified.” After preprocessing, one-hot encoding was used to label the images.

*Step 4. Saving compressed files*

We divided the data into a training dataset, a validation dataset, and a test dataset, which respectively accounted for 60%, 20%, and 20% of the total data. To avoid re-labelling the data before each training, the labelled and cropped datasets were saved in an npz file to be loaded; each time training is conducted. To save time and prevent the training data from taking up too much storage, the original 416×416 pixels were compressed to 224×224 pixels and saved in an npz file. To determine whether data size influences the predictive capabilities of the model, an npz file was created with the data in 128×128 pixels. To match the size of the input data in the original GoogLeNet v4 paper [19], an npz file was saved with the data in 299×299 pixels to compare the predictive capability of the proposed approach with this model. As the amount of data collected was not large enough, angle rotation, brightness adjustment, horizontal shift, vertical shift, scaling, and vertical flipping were employed to augment the number of images and avoid overfitting.

Albawi et al. [20] pointed out that CNNs are currently one of the most popular types of neural network architectures.
They are generally superior to ANNs because the convolution and pooling layers in their architectures reinforce the relationships between image recognition and neighbouring data. CNNs have offered relatively good achievements in various applications such as image recognition and voice recognition, to the point of exceeding human performance in recent years. They are thus one of the main forces in DL progress at present. A typical CNN architecture includes a convolution layer, a pooling layer, and a fully connected layer, as shown in Fig. 4. Techniques such as padding, strides, and dropping neurons are often incorporated. The convolution layer is the core of a CNN. The operations involve multiplying and summing corresponding elements in the input data and the kernel. To lower the amount of computation and increase computational efficiency, a pooling layer is often added to CNNs. Pooling refers to the dimensional reduction of the input data in the width and length directions, thereby reducing the amount of data while preserving important information; it can also lower the possibility of overfitting. A fully connected layer is connected to all of the neurons in neighbouring layers, and the last fully connected layer is used to classify problems.

As done by Alencastre-Miranda et al. [3] and Dorafshan and Azari [4], the performances of the following four architectures were compared: AlexNet, VGG-16, GoogLeNet, and ResNet. AlexNet is a CNN model proposed by Krizhevsky et al. [21] with an eight-layer architecture: five convolution layers and three fully connected layers, combined with three maximum pooling layers. Its default input images are colour images with 224×224 pixels. VGG is a CNN model proposed by Simonyan and Zisserman [22]. Its variants include VGG-11, VGG-13 VGG-16, and VGG-19, among which VGG-16 and VGG-19 are the best in performance. In this study, VGG-16, which has fewer parameters, was employed. It contains 13 convolution layers and three fully connected layers and is somewhat similar to AlexNet in structure. Its default input images are colour images with 224×224 pixels. GoogLeNet is a CNN model proposed by Szegedy et al. [23]. In the deepening of this network, it replaces the original pure convolution and pooling layers with Inception structures, unlike the concepts of AlexNet or VGG. It also has fewer parameters than AlexNet but with a deeper network and greater accuracy. ResNet is a CNN model proposed by He et al. [24]. They pointed out that degradation is often encountered during the training of deep network models; to address this, they proposed residual learning architectures.

Prior to detection of sandblasting defects, it was confirmed that the investment casting appeared in the image using YOLO v3. If the investment casting was detected, then a certain range was extracted, and CNN defect detection was performed. YOLO v3 is a CNN-based object detection algorithm proposed by Redmon and Farhadi [25] which uses the Darknet-53 network architecture. It also refers to FPN methods and uses multi-scale feature maps to recognize objects of different sizes to enhance recognition capacity of small objects.

3 Results and discussion

The DL in this study was established under Anaconda Spyder Python 3.6. Experiments were run on a computer with an Intel Core i7-9700 processor using 16 GB of RAM, and a NAVIDIA GeForce RTX 2060 graphics card in Microsoft Windows 10.

3.1 Deep-learning training results and model evaluation

Four CNN model architectures were trained. The results of training were compared with data in 128×128 pixel format and in 224×224 pixel format, as well as the results of training with batch size = 4, batch size = 8, batch size = 16, and batch size = 32. ResNet-34 performed poorly in the prediction of sandblasting defects. To identify the cause of this, a comparison was performed using GoogLeNet v4, which also employs residual learning. An additional comparison was performed using the default size of input data in GoogLeNet v4, which is 299×299 pixels.

For ease of description, [128, 4] is used to denote input data in 128×128 pixel format with batch size = 4 and [128, 8] to denote input data in 128×128 pixel format with batch size = 8. Similarly, there are [128, 16], [128, 32], [224, 4] [224, 8], [224, 16], [224, 32], [299, 4], [299, 8], [299, 16], and [299, 32].
Figure 5 presents the historical accuracy rates of the AlexNet model. The differences among the results from the 12 sets of training data were small, and the accuracy rates all converged to 1.

Figure 6 presents the historical accuracy rates of the VGG-16 model. Although the accuracy rates involving [128, 4], [128, 8], and [128, 16] fluctuated somewhat, they still converged to 1. The VGG-16 model only presented poor performance when trained using [224, 4], and its accuracy rates could not effectively converge to 1 late in the training period, remaining around 0.65.

As VGG-16 is deeper and has more parameters, the RAM of our computer was unable to process the input data in 224×224 pixel format with batch size = 32 or the input data in 299×299 pixel format. Training with these datasets could not be completed, and consequently, there were no results for these datasets.

Figure 7 presents the historical accuracy rates of the GoogLeNet v1 model. The differences among the results from the 12 sets of training data were small, and the accuracy rates all converged to 1.

Figure 8 displays the historical accuracy rates of the training results of the GoogLeNet v4 model. The training accuracy rates resulting from the 12 sets of training data all converged to 1; however, the validation accuracy rates fluctuated and could not effectively converge to 1.

Figure 9 shows the historical accuracy rates of the training results of the ResNet-34 model. The training accuracy rates resulting from the 12 sets of training data all converged to 1; however, the validation accuracy rates fluctuated sharply and could not converge to 1.

The objective of this study was to detect sandblasting defects in investment castings, so more importance was attached to the “unqualified” prediction results. An “unqualified” casting predicted as “unqualified” represents a true positive (TP), whereas an “unqualified” casting predicted as “qualified” represents a false negative (FN). Similarly, a “qualified” casting predicted as “qualified” represents a true negative (TN), whereas a “qualified” casting predicted as “unqualified” represents a false positive (FP).

\[
\text{recall} = \frac{TP}{TP + FN} \quad (1)
\]

\[
\text{precision} = \frac{TP}{TP + FP} \quad (2)
\]

**Fig. 5** Historical accuracy of AlexNet model
Eqs. (1) through (3). Table 1 displays the training time, their formulas are as shown in Eqs. (1) through (3). Table 1 displays the training time, average time per image, and the other evaluation indices resulting from applying 54 weights to the test datasets.

In terms of overall training time, AlexNet took the least time among the five models. GoogLeNet v1 was close to AlexNet in training time, followed by ResNet-34. On the whole, GoogLeNet v4 took the longest time. In terms of

\[ \text{F1 score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \]  

(3)

Recall, precision, and the F1 score were used to evaluate model training results. Their formulas are as shown in Eqs. (1) through (3). Table 1 displays the training time, average time per image, and the other evaluation indices resulting from applying 54 weights to the test datasets.

In terms of overall training time, AlexNet took the least time among the five models. GoogLeNet v1 was close to AlexNet in training time, followed by ResNet-34. On the whole, GoogLeNet v4 took the longest time. In terms of
GoogLeNet v4 was the slowest, taking 9.6 to 12.4 ms, followed by GoogLeNet v1, which took 1.9 to 3.2 ms. Among the four models, ResNet-34 was the fastest, taking less than 3 ms per image. GoogLeNet v4 was the slowest, taking 9.6 to 12.4 ms.

Recall measures the proportion of correctly predicted positives in the positive class; its formula is shown in Eq. (1). AlexNet and GoogLeNet v1 predicted high proportions of the "unqualified" samples as "unqualified," their...
### Table 1 Evaluation index for each training model

| Compared parameters | Training time (min) | Average prediction time (ms) | Recall | Precision | F1 score |
|---------------------|--------------------|-----------------------------|--------|-----------|----------|
| AlexNet             | [128, 4]           | 39                          | 2.9126 | 96.79%    | 99.53%   | 98.14%   |
|                     | [128, 8]           | 20                          | 0.9566 | 100.00%   | 99.06%   | 99.53%   |
|                     | [128, 16]          | 11                          | 0.9554 | 99.53%    | 99.53%   | 99.53%   |
|                     | [128, 32]          | 10                          | 1.0026 | 99.53%    | 100.00%  | 99.76%   |
|                     | [224, 4]           | 92                          | 1.1961 | 98.60%    | 100.00%  | 99.76%   |
|                     | [224, 8]           | 51                          | 1.1221 | 98.60%    | 100.00%  | 99.76%   |
|                     | [224, 16]          | 30                          | 1.1928 | 99.53%    | 100.00%  | 99.76%   |
|                     | [224, 32]          | 27                          | 1.1171 | 99.52%    | 98.11%   | 98.81%   |
|                     | [299, 4]           | 167                         | 1.6361 | 97.22%    | 99.06%   | 98.13%   |
|                     | [299, 8]           | 94                          | 1.5217 | 90.21%    | 100.00%  | 94.85%   |
|                     | [299, 16]          | 56                          | 1.5170 | 99.07%    | 100.00%  | 99.53%   |
|                     | [299, 32]          | 49                          | 1.4792 | 97.22%    | 99.06%   | 98.13%   |
| GoogLeNet v1        | [128, 4]           | 43                          | 2.2590 | 98.95%    | 88.68%   | 93.53%   |
|                     | [128, 8]           | 20                          | 2.0037 | 99.51%    | 95.75%   | 97.60%   |
|                     | [128, 16]          | 12                          | 1.9985 | 83.00%    | 96.70%   | 89.32%   |
|                     | [128, 32]          | 10                          | 1.9783 | 99.51%    | 96.70%   | 98.09%   |
|                     | [224, 4]           | 96                          | 2.5093 | 99.43%    | 81.60%   | 89.64%   |
|                     | [224, 8]           | 52                          | 2.2668 | 98.87%    | 82.55%   | 89.97%   |
|                     | [224, 16]          | 31                          | 2.2733 | 99.48%    | 89.62%   | 94.29%   |
|                     | [224, 32]          | 27                          | 2.2327 | 97.12%    | 95.28%   | 96.19%   |
|                     | [299, 4]           | 171                         | 3.2084 | 98.97%    | 90.57%   | 94.58%   |
|                     | [299, 8]           | 96                          | 2.7287 | 98.26%    | 53.30%   | 69.11%   |
|                     | [299, 16]          | 57                          | 2.7042 | 96.80%    | 100.00%  | 98.38%   |
|                     | [299, 32]          | 50                          | 2.6987 | 98.97%    | 90.57%   | 94.58%   |
| VGG-16              | [128, 4]           | 84                          | 2.9479 | 66.67%    | 0.94%    | 1.86%    |
|                     | [128, 8]           | 54                          | 2.2591 | 98.56%    | 96.70%   | 97.62%   |
|                     | [128, 16]          | 25                          | 2.5013 | 24.24%    | 7.55%    | 11.51%   |
|                     | [128, 32]          | 17                          | 2.4478 | 92.17%    | 100.00%  | 95.93%   |
|                     | [224, 4]           | 194                         | 4.6935 | 66.46%    | 100.00%  | 79.85%   |
|                     | [224, 8]           | 109                         | 4.8705 | 99.07%    | 100.00%  | 99.53%   |

"-" indicates that a denominator equalled 0 during the calculation process, so the value could not be calculated.
recall values both exceeding 90.00%. The recall of AlexNet with [128, 8] even reached 100.00%. In contrast, VGG-16 presented low recall values when batch size = 4 and batch size = 16. Therefore, batch size may affect the model training results. Precision measures the proportion of true positives among all predicted positives; its formula is shown in Eq. (2). AlexNet displayed exceptional performance in this respect, achieving 100% with six of the datasets. Furthermore, the precision of VGG-16 with [128, 32], [224, 4], and [224, 8] reached 100.00%. Compared to the two models above, GoogLeNet v1 showed slightly poorer performance, only achieving 100.00% precision in one dataset. Although GoogLeNet v4 achieved 100.00% precision in three datasets, the quality of this model cannot be determined based on a single index; other indices such as the F1 score must also be included for comprehensive assessment. The F1 score is a balanced measure of recall and precision; its formula is shown in Eq. (3). On the whole, AlexNet exceeded 98.00% with most of the datasets, even reaching 99.76% with [128, 32], [224, 4], and [224, 16], followed by 99.53% with [128, 8], [128, 16], and [299, 16]. Although the F1 score of VGG-16 with [224, 8] was 99.53%, it was still slightly inferior to AlexNet. GoogLeNet v1 only achieved F1 scores of 98.09% and 98.38% with [128, 32] and [299, 16]; its ten other F1 scores were only around 80.00%. A comprehensive review of Table 1 reveals that GoogLeNet v4 and ResNet-34 did not perform as expected in defect prediction. Therefore, the investment castings used in this study may not be suitable for training models with residual learning architectures in defect detection.

YOLO v3 was applied to detect whether investment castings appeared in the machine vision images, so only one YOLO v3 category was defined: “sample.” Thus, when training YOLO v3, the impact of data size or batch size was not investigated. Models with better object tracking performance generally display the following: “samples” take up a greater proportion of the recognized images, and when other categories are identified, they identify “samples” as much as possible. In other words, while increasing the recall, a certain level of precision must also be maintained. In addition, the area under the precision-recall curve (i.e., average precision (AP)) is generally used for evaluation. Calculations revealed that the AP of the “samples” in this study was 99.83%, indicating that the YOLO v3 presented excellent performance.

### 3.2 Design and practical application of graphical user interface

To make it convenient for users to perform sandblasting-defect detection, the PyQt5 library was applied to design a graphical user interface (GUI). After images are captured under sufficient lighting, YOLO v3 analyses them to detect investment castings, and then, the images are sent to the

---

**Fig. 10** Demonstrations of GUI: a casting part (orange) exceeds detection boundary; b “unqualified” (red) and “qualified” (green) parts
CNN for defect detection. Upon starting, the application automatically loads the trained YOLO v3 weights and CNN weights. Before predicting the type of investment casting sandblasting, a certain range of the camera is first extracted (the dimensions of this range depend on the image size designated during CNN training). To prevent capture failure, a detection boundary was added, as shown in the off-white frames in the images on the left of Fig. 10. When YOLO v3 detects an investment casting, the type of investment casting sandblasting is not predicted unless the centre point of the investment casting frame falls within the detection boundary. At the same time, the frame will display warning text. The results are as shown with the orange frame in the right image of Fig. 10a. In contrast, if YOLO v3 detects an investment casting and the centre of its frame falls within the detection boundary, then the type of investment casting sandblasting is predicted using CNN. The results are as shown in Fig. 10b: the green frame in the right image indicates a “qualified” casting, and the red frame indicates an “unqualified” casting.

4 Conclusions

In the assessment of various classic CNN models using different indices, the AlexNet model with data in 128 × 128 pixel format and batch size = 32 displayed the best training performance, followed by the AlexNet model with data in 224 × 224 pixel format and batch size = 16, the AlexNet model with data in 224 × 224 pixel format and batch size = 4, and the VGG-16 model with data in 224 × 224 pixel format and batch size = 8. It seems that batch size influences the training results to some degree; however, due to hardware limitations, it was not possible to add more training data to prove this. However, it is clear that the magnitude of this value affects training time. GoogLeNet v4 and ResNet-34 presented poor prediction capabilities, with no discrimination among 24 training datasets. Therefore, DL performed using CNN models with residual learning architectures is not suitable for the detection of sandblasting defects in investment castings.

Although the average precision of YOLO v3 in predicting “samples” is 99.83%, the training datasets were more uniform (the images all contained investment castings against a white background), so for some camera angles, non-background objects in more complex images may be mistaken for “samples” in practical application. Future studies should define investment casting categories based on their shape, such as “sample a,” “sample b,” and “sample c,” or add more images with multiple investment castings in the same image and more investment castings with some overlapping each other to enhance image diversity.

During training, the sizes of the input images were adjusted. Thus, when the application needs to detect objects, it captures images that are the same size as the input images based on the centres of the objects. A detection boundary was therefore added to the left image of the application (the actual YOLO v3 object detection image), and the type of investment casting sandblasting was only predicted within the detection boundary. Furthermore, to implement image monitoring, a proper distance between the camera and the investment castings must be maintained so that complete images of the investment castings can be captured.

Acknowledgements The authors gratefully acknowledge Hei Full Industrial Co., LTD, Taiwan, for providing all casting samples for testing.

Author contribution Jenn-Kun Kuo: conceptualization, formal analysis, writing — review and editing. Jun-Jia Wu: investigation. Pei-Hsing Huang: supervision. Chin-Yi Cheng: AOI technology.

Funding This study received financial support from the Ministry of Science and Technology, R.O.C., under grants MOST 110–2221-E-224–031 and MOST 109–2221-E-224–014.

Declarations

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Consent to participate Not applicable.

Consent for publication Not applicable.

Competing interests The authors declare no competing interests.

References

1. Korodi A, Anitei D, Boitor A, Silea I (2020) Image-processing-based low-cost fault detection solution for end-of-line ECUs in automotive manufacturing. Sensors 20(12):3520
2. Luo Q et al (2020) Automated visual defect classification for flat steel surface: a survey. IEEE Trans Instrum Meas 69(12):9329–9349
3. Alencastre-Miranda M, Johnson RM, Krebs HJ (2020) Convolutional neural networks and transfer learning for quality inspection of different sugarcane varieties. IEEE Trans Ind Inform 17(2):787–794
4. Dorafshan S, Azari H (2020) Deep learning models for bridge deck evaluation using impact echo. Constr Build Mater 263:120109
5. Kuo C-FJ, Fang T-Y, Lee C-L, Wu H-C (2019) Automated optical inspection system for surface mount device light emitting diodes. J Intell Manuf 30(2):641–655
6. Li X, Tso SK, Guan X-P, Huang Q (2006) Improving automatic detection of defects in castings by applying wavelet technique. IEEE Trans Ind Electr 53(6):1927–1934
7. Jeon Y-J, Choi D-C, Lee SJ, Yun JP, Kim SW (2014) Defect detection for corner cracks in steel billets using a wavelet reconstruction method. JOSA A 31(2):227–237
8. Bakir B (2007) Defect cause modeling with decision tree and regression analysis
9. Gu IY-H, Andersson H, Vicen R (2010) Wood defect classification based on image analysis and support vector machines. Wood Sci Technol 44(4):693–704
10. Li M, Wan S, Deng Z, Wang Y (2019) Fabric defect detection based on saliency histogram features. Comput Intell 35(3):517–534
11. Ye X, Wu X, Guo Y (2018) Real-time quality prediction of casting billet based on random forest algorithm. Paper presented at the 2018 IEEE International Conference on Progress in Informatics and Computing (PIC)
12. Riaz F, Kamal K, Zafar T, Qayyum R (2017) An inspection approach for casting defects detection using image segmentation paper presented at the 2017 International Conference on Mechanical, System and Control Engineering (ICMSC)
13. Iyer S, Velmurugan T, Gandomi A, Mohammed VN, Saravanan K, Nandakumar S (2020) Structural health monitoring of railway tracks using IoT-based multi-robot system. Neural Comput Appl 1–19
14. Li J, Su Z, Geng J, Yin Y (2018) Real-time detection of steel strip surface defects based on improved YOLO detection network. IFAC-PapersOnLine 51(21):76–81
15. Raj VG, Srijahi M, Mohan A (2021) Casting defect detection using YOLO V4. Int Res J Mod Eng Technol Sci 3(4):1581–1585
16. Shi W, Lu Z, Wu W, Liu H (2020) Single-shot detector with enriched semantics for PCB tiny defect detection. J Eng 13:366–372
17. Du W, Shen H, Fu J, Zhang G, He Q (2019) Approaches for improvement of the X-ray image defect detection of automobile casting aluminum parts based on deep learning. NDT E Int 107:102144
18. OpenCV Reference Guide. [Electronic resource] URL: http://docs.opencv.org
19. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. Paper presented at the Thirty-first AAAI conference on artificial intelligence
20. Albawi S, Mohammed TA, Al-Zawi S (2017) Understanding of a convolutional neural network. Paper presented at the 2017 International Conference on Engineering and Technology (ICET)
21. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Adv Neural Inform Proc Syst 25:1097–1105
22. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
23. Szegedy C et al (2015). Going deeper with convolutions. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition
24. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition
25. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. arXiv preprint arXiv:1804.02767

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