Sandblasting Defect Inspection of Investment Castings Based on Deep Convolutional Neural Network

Jenn-Kun Kuo
National Sun Yat-sen University
Jun-Jia Wu
National Yunlin University of Science and Technology
Pei-Hsing Huang (huangph@yuntech.edu.tw)
National Yunlin University of Science and Technology https://orcid.org/0000-0002-5287-3060
Chin-Yi Cheng
National Yunlin University of Science and Technology

Research Article

Keywords: Automated optical inspection, investment castings, sandblasting defects, convolutional neural network, deep learning

Posted Date: December 13th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1139647/v1

Version of Record: A version of this preprint was published at The International Journal of Advanced Manufacturing Technology on February 23rd, 2022. See the published version at https://doi.org/10.1007/s00170-022-08841-w.

Abstract

Investment castings often have surface impurities and pieces of shell molds 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 labor costs and increase the health and safety of employees, we applied automated optical inspection (AOI) combined with a deep learning framework based on convolutional neural networks (CNNs) to the detection of sandblasting defects. We applied the following four classic CNN models for training and predictive classification: AlexNet, VGG-16, GoogLeNet, and ResNet-34. In terms of predictive classification, AlexNet, VGG-16, and GoogLeNet v1 could accurately determine whether there were defects. Among the four models, AlexNet was the most accurate, with prediction accuracy of 99.53% for qualifying products and 100% for defective products. We demonstrate a direct detection technique based on the AOI and CNN structure with a fast and flexible computational interface.

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 molds. 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 optimal 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 to good effect in processes such as battery laser welding, flat steel manufacturing, automotive manufacturing, and semiconductor packaging [1,2]. 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, bridge deck assessment, and railway track inspection[3,4].

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%. Dorafshan et al. [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 pres, pinholes, burrs, shrinkage defects, mold 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 aluminum 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 aluminum castings achieved better performance than Faster-RCNN.

Existing studies have rarely focused on the detection of impurities or remnants of shell molds 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, in this study we employed AOI for defect detection. However, different castings and surface properties require changes to the underlying algorithm, thereby reducing detection compatibility. Thus, we paired the AOI technology with DL techniques to develop defect-detection software for lost-wax investment castings.

2. Research Procedures And Methods

The Pre-processing of DL involves the classification and labelling of training data. A detailed data collection process, compressed-file establishment, and labelling of the dataset were given as follows:

Step 1. Camera setup and image capture

We used the camera function of a smartphone to capture images of investment castings provided by a manufacturer. To obtain images of equal size, we installed the smartphone on a tripod at a fixed height (h = 130 cm), as shown in Fig. 1(a). Furthermore, to reduce the number of images captured and to capture individual samples completely, we used white paper with 6 × 4 grids lined in black 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. 1(b). 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. We took images of 16 types of investment castings and obtained around 100 images like those shown in Fig. 1(b).

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, we employed the OpenCV library [18] for grayscale processing, Gaussian blurring, edge detection for ROI identification, and the cropping and saving of individual sample images. We set the size

of the ROIs at 416 × 416 pixels. Using edge detection, rectangles with a certain width and length were selected, and using the coordinates of the upper left corner, we calculated the coordinates of the rectangle centers, as shown in Fig. 2(a). Next, using the coordinates of the centers and the target width and length, we calculated the coordinates of the upper left corner, as shown in Fig. 2(b). Once the rectangle coordinates were revised, we could crop the target rectangle. In total, we obtained 1,591 cropped images, some of which are exhibited in Fig. 3.

Step 3. Image labelling

Before training, the images must be labeled. This provides samples for training, which the CNNs must learn to be able to classify and recognize un-labelled data. We divided the investment casting images into two groups based on whether they had undergone sandblasting: (1) those in which there were clearly-visible remnants of shell molds 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 molds 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, we saved the labelled and cropped datasets in an npz file that can be loaded every time training is conducted. To save time and prevent the training data from taking up too much storage, we compressed the original 416 × 416 pixels to 224 × 224 pixels and saved the data in an npz file. To determine whether data size influences the predictive capabilities of the model, we also created an npz file with the data in 128 × 128 pixels. To match the size of the input data in the original GoogLeNet v4 paper [19], we also saved an npz file with the data in 299 × 299 pixels to compare the predictive capability of our approach with this model. As the amount of data collected was not large enough, we employed angle rotation, brightness adjustment, horizontal shift, vertical shift, scaling, and vertical flipping 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 neighboring data. CNNs have made 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 neighboring layers, and the last fully-connected layer is used to classify problems.

As done by Alencastre-Miranda et al. [3],[4], we compared the performances of the following four architectures: AlexNet, VGG-16, GoogLeNet, and ResNet. AlexNet is a CNN model proposed by Krizhevsky et al. [19] with an eight-layer architecture: five convolution layers and three fully-connected layers, combined with three maximum pooling layers. Its default input images are color images with 224 × 224 pixels. VGG is a CNN model proposed by Simonyan et al. [20]. 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, we employed VGG-16, which has fewer parameters. It contains 13 convolution layers and three fully-connected layers and is somewhat similar to AlexNet in structure. Its default input images are color images with 224 × 224 pixels. GoogLeNet is a CNN model proposed by Szegedy et al. [21]. 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. [22]. 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 detecting sandblasting defects, we 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 et al. [23] 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 Discussions

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

We trained four CNN model architectures and compared the results of training 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, we performed a comparison using GoogLeNet v4, which also employs residual learning. We additionally added a comparison using the default size of input data in GoogLeNet v4, which is 299 × 299 pixels.

For ease of description, we use [128, 4] 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 we attached more importance 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).

$$recall = \frac{TP}{TP + FN} \tag{1}$$

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$F1\ score = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

We used recall, precision, and the F1 score 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 average time spent on the test dataset, AlexNet was the fastest among the four models, which took less than 3 milliseconds per image, followed by GoogLeNet v1, which took 1.9 milliseconds to 3.2 milliseconds. GoogLeNet v4 was the slowest, taking 9.6 milliseconds to 12.4 milliseconds.

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	99.53%	100.00%	99.76%
[224, 8]	51	1.1221	98.60%	100.00%	99.30%
[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	83.00%	96.70%	89.32%
[128, 8]	20	2.0037	99.51%	96.70%	98.09%
[128, 16]	12	1.9985	99.43%	81.60%	89.64%
[128, 32]	10	1.9783	98.87%	82.55%	89.97%
[224, 4]	96	2.5093	99.48%	89.62%	94.29%
[224, 8]	52	2.2668	97.12%	95.28%	96.19%
[224, 16]	31	2.2733	98.97%	90.57%	94.58%
[224, 32]	27	2.2327	98.26%	53.30%	69.11%
[299, 4]	171	3.2084	96.80%	100.00%	98.38%
[299, 8]	96	2.7287	98.97%	90.57%	94.58%
[299, 16]	57	2.7042	-	0.00%	-
[299, 32]	50	2.6987	-	0.00%	-
VGG-16					
[128, 4]	84	2.9479	98.56%	96.70%	97.62%
[128, 8]	54	2.2591	24.24%	7.55%	11.51%
[128, 16]	25	2.5013	92.17%	100.00%	95.93%
[128, 32]	17	2.4478	66.46%	100.00%	79.85%
[224, 4]	194	4.6935	99.07%	100.00%	99.53%
[224, 8]	109	4.8705	98.95%	88.68%	93.53%

Compared parameters	Training time (min)	Average prediction time (ms)	Recall	Precision	F1 score
ResNet-34					
[128, 4]	63	3.4315	-	0.00%	-
[128, 8]	32	3.1633	-	0.00%	-
[128, 16]	16	3.1761	100.00%	0.94%	1.87%
[128, 32]	10	3.1393	-	0.00%	-
[224, 4]	113	3.6855	-	0.00%	-
[224, 8]	60	3.6504	-	0.00%	-
[224, 16]	32	3.6421	-	0.00%	-
[224, 32]	28	3.6302	-	0.00%	-
[299, 4]	181	4.2100	96.79%	99.53%	98.14%
[299, 8]	96	4.1471	100.00%	99.06%	99.53%
[299, 16]	58	4.1420	99.53%	99.53%	99.53%
[299, 32]	50	4.1393	99.53%	100.00%	99.76%
GoogLeNet v4					
[128, 4]	43	2.2590	83.00%	96.70%	89.32%
[128, 8]	20	2.0037	99.51%	96.70%	98.09%
[128, 16]	12	1.9985	99.43%	81.60%	89.64%
[128, 32]	10	1.9783	98.87%	82.55%	89.97%
[224, 4]	96	2.5093	99.48%	89.62%	94.29%
[224, 8]	52	2.2668	97.12%	95.28%	96.19%
[224, 16]	31	2.2733	98.97%	90.57%	94.58%
[224, 32]	27	2.2327	98.26%	53.30%	69.11%
[299, 4]	171	3.2084	96.80%	100.00%	98.38%
[299, 8]	96	2.7287	98.97%	90.57%	94.58%
[299, 16]	57	2.7042	-	0.00%	-
[299, 32]	50	2.6987	-	0.00%	-

Note: "-" indicates that a denominator equaled 0 during the calculation process, so the value could not be calculated.

Recall measures the proportion of correctly-predicted positives in the positive class; its equation is as shown in Eq. (1). AlexNet and GoogLeNet v1 predicted high proportions of the "unqualified" samples as "unqualified", their 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. We therefore speculate that batch size may affect the model training results. Precision measures the proportion of true positives among all predicted positives; its equation is as 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 equation is as 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. We therefore speculate that the investment castings used in this study were not suitable for training models with residual learning architectures in defect detection.

We employed YOLO v3 to detect whether investment castings appeared in the machine vision images, so we only defined one YOLO v3 category: “sample”. Thus, when training YOLO v3, we did not investigate the impact of data size or batch size. 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, we employed the PyQt5 library to design a graphical user interface (GUI). After images are captured under sufficient lighting, YOLO v3 analyzes them to detect investment castings, and then the images are sent to the 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, we added a detection boundary, 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 center 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. (a). In contrast, if YOLO v3 detects an investment casting and the center 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. (b): the green frame in the right image indicates a “qualified” casting and the red frame indicates an “unqualified” casting.

4. Conclusions

In our 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. We speculated that batch size influences the training results to some degree; however, limited by our hardware, we could not add more training data to prove this. However, we can be sure 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. We therefore speculate that DL performed using CNN models with residual learning architectures is not suitable for the detection of sandblasting defects in the 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. We suggest that future studies 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, we re-adjusted the sizes of the input images. Thus, when the application needs to detect objects, it captures images that are the same size as the input images based on the centers of the objects. We therefore added a detection boundary to the left image of the application (the actual YOLO v3 object detection image) and only predicted the type of investment casting sandblasting 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.

Declarations

Acknowledgments

The authors gratefully acknowledge the support provided all casting samples for this research by Hei Full Industrial Co., LTD, Taiwan, for providing all casting samples for testing.

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.

Author information

Affiliations

Department of Mechanical and Electro-Mechanical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, R.O.C.

Department of Greenergy, National University of Tainan, Tainan 70005, Taiwan, R. O. C.

Jenn-Kun Kuo

Department of Mechanical Engineering, National Yunlin University of Science and Technology, Yunlin 912, Taiwan, R. O. C.

Jun-Jia Wu, Pei-Hsing Huang, & Chin-Yi Cheng

Contributions

Jenn-Kun Kuo: Conceptualization, Formal analysis, Writing - review & editing. Jun-Jia Wu: Investigation. Pei-Hsing Huang: Supervision. Chin-Yi Cheng: AOI technology.

Corresponding authors

Correspondence to Pei-Hsing Huang.

Ethics 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 to publish

Not applicable.

Competing interests

The authors declare no competing interests.

References

1. A. Korodi, D. Anitei, A. Boitor, and I. Silea. (2020). Image-Processing-Based Low-Cost Fault Detection Solution for End-of-Line ECUs in Automotive Manufacturing. *Sensors, 20*(12), 3520.

2. Q. Luo *et al.* (2020). Automated Visual Defect Classification for Flat Steel Surface: A Survey. *IEEE Transactions on Instrumentation and Measurement, 69*(12), 9329-9349.

3. M. Alencastre-Miranda, R. M. Johnson, and H. I. Krebs. (2020). Convolutional Neural Networks and Transfer Learning for Quality Inspection of Different Sugarcane Varieties. *IEEE Transactions on Industrial Informatics, 17*(2), 787-794.

4. S. Dorafshan and H. Azari. (2020). Deep learning models for bridge deck evaluation using impact echo. *Construction and Building Materials, 263*, 120109.

5. C.-F. J. Kuo, T.-y. Fang, C.-L. Lee, and H.-C. Wu. (2019). Automated optical inspection system for surface mount device light emitting diodes. *Journal of Intelligent Manufacturing, 30*(2), 641-655.

6. X. Li, S. K. Tso, X.-P. Guan, and Q. Huang. (2006). Improving Automatic Detection of Defects in Castings by Applying Wavelet Technique. *IEEE Transactions on Industrial Electronics, 53*(6), 1927-1934.

7. Y.-J. Jeon, D.-c. Choi, S. J. Lee, J. P. Yun, and S. W. Kim. (2014). Defect detection for corner cracks in steel billets using a wavelet reconstruction method. *JOSA A, 31*(2), 227-237.

8. B. Bakir. (2007). Defect Cause Modeling with Decision Tree and Regression Analysis.

9. I. Y.-H. Gu, H. Andersson, and R. Vicen. (2010). Wood defect classification based on image analysis and support vector machines. *Wood science and technology, 44*(4), 693-704.

10. M. Li, S. Wan, Z. Deng, and Y. Wang. (2019). Fabric defect detection based on saliency histogram features. *Computational Intelligence, 35*(3), 517-534.

11. X. Ye, X. Wu, and Y. Guo. (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. F. Riaz, K. Kamal, T. Zafar, and R. Qayyum. (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. S. Iyer, T. Velmurugan, A. Gandomi, V. N. Mohammed, K. Saravanan, and S. Nandakumar. (2020). Structural health monitoring of railway tracks using IoT-based multi-robot system. *Neural Computing and Applications*, 1-19.

14. J. Li, Z. Su, J. Geng, and Y. Yin. (2018). Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network. *IFAC-PapersOnLine, 51*(21), 76-81.

15. V. G. Raj, M. Srihari, and A. Mohan. CASTING DEFECT DETECTION USING YOLO V4.

16. W. Shi, Z. Lu, W. Wu, and H. Liu. (2020). Single-shot detector with enriched semantics for PCB tiny defect detection. *The Journal of Engineering, 2020*(13), 366-372.

17. W. Du, H. Shen, J. Fu, G. Zhang, and Q. He. (2019). Approaches for improvement of the X-ray image defect detection of automobile casting aluminum parts based on deep learning. *NDT & E International, 107*, 102144.

18. OpenCV Reference Guide. [Electronic resource] URL: http://docs.opencv.org

19. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. (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. S. Albawi, T. A. Mohammed, and S. Al-Zawi. (2017). Understanding of a convolutional neural network. Paper presented at the 2017 International Conference on Engineering and Technology (ICET).

21. A. Krizhevsky, I. Sutskever, and G. E. Hinton. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems, 25*, 1097-1105.

22. K. Simonyan and A. Zisserman. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

23. C. Szegedy *et al.* (2015). Going Deeper with Convolutions. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

24. K. He, X. Zhang, S. Ren, and J. Sun. (2016). Deep Residual Learning for Image Recognition. Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

25. J. Redmon and A. Farhadi. (2018). YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.

Figures

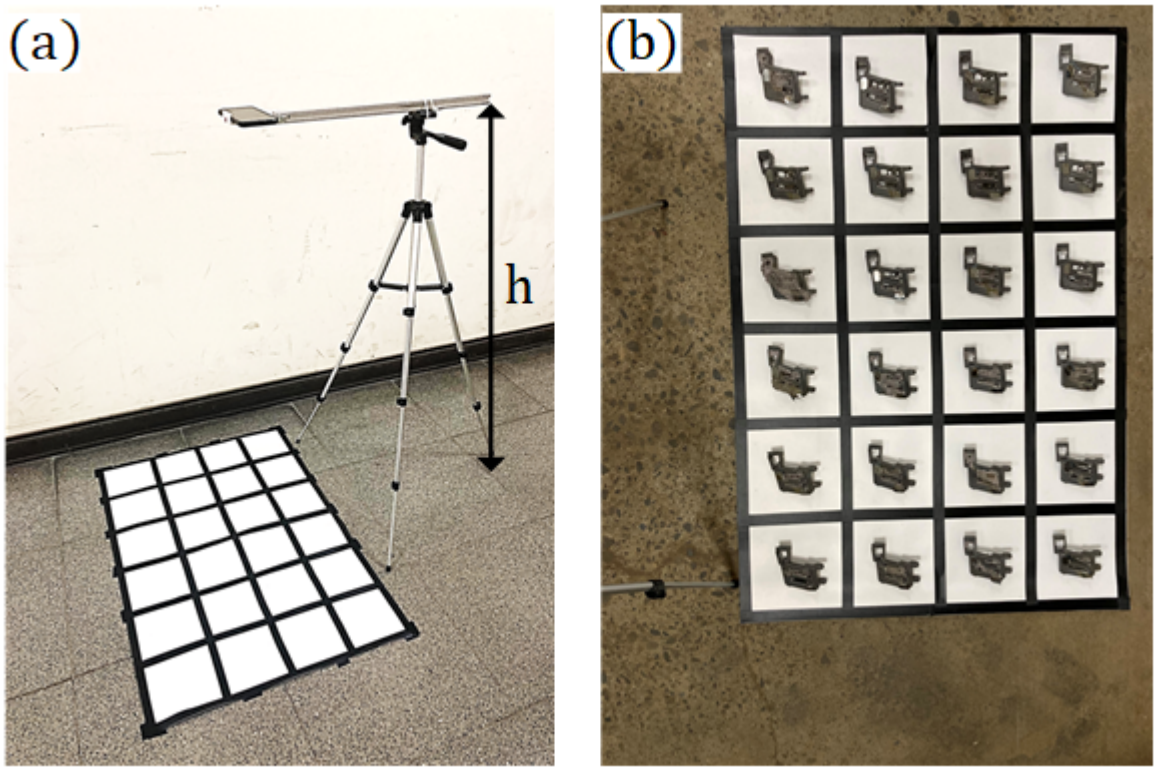

Figure 1

(a) Camera setup (h = 130 cm); (b) placing of investment castings on background

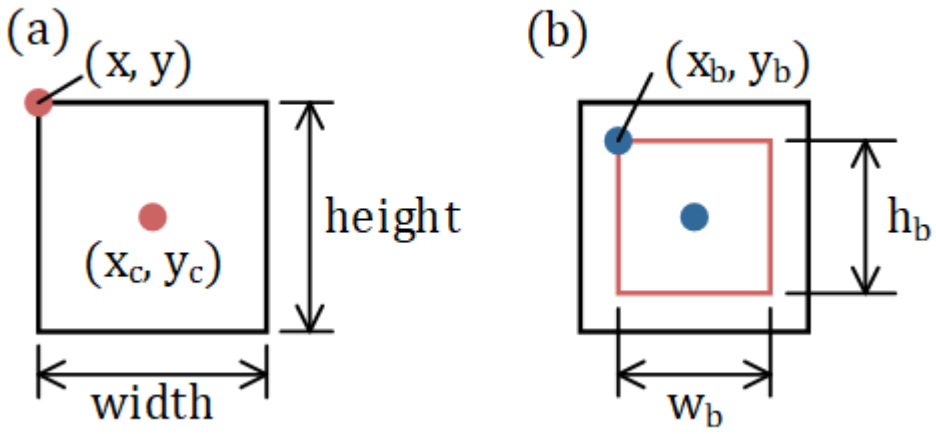

Figure 2

(a) Rectangular area after edge detection of workpiece; (b) ROI after correction

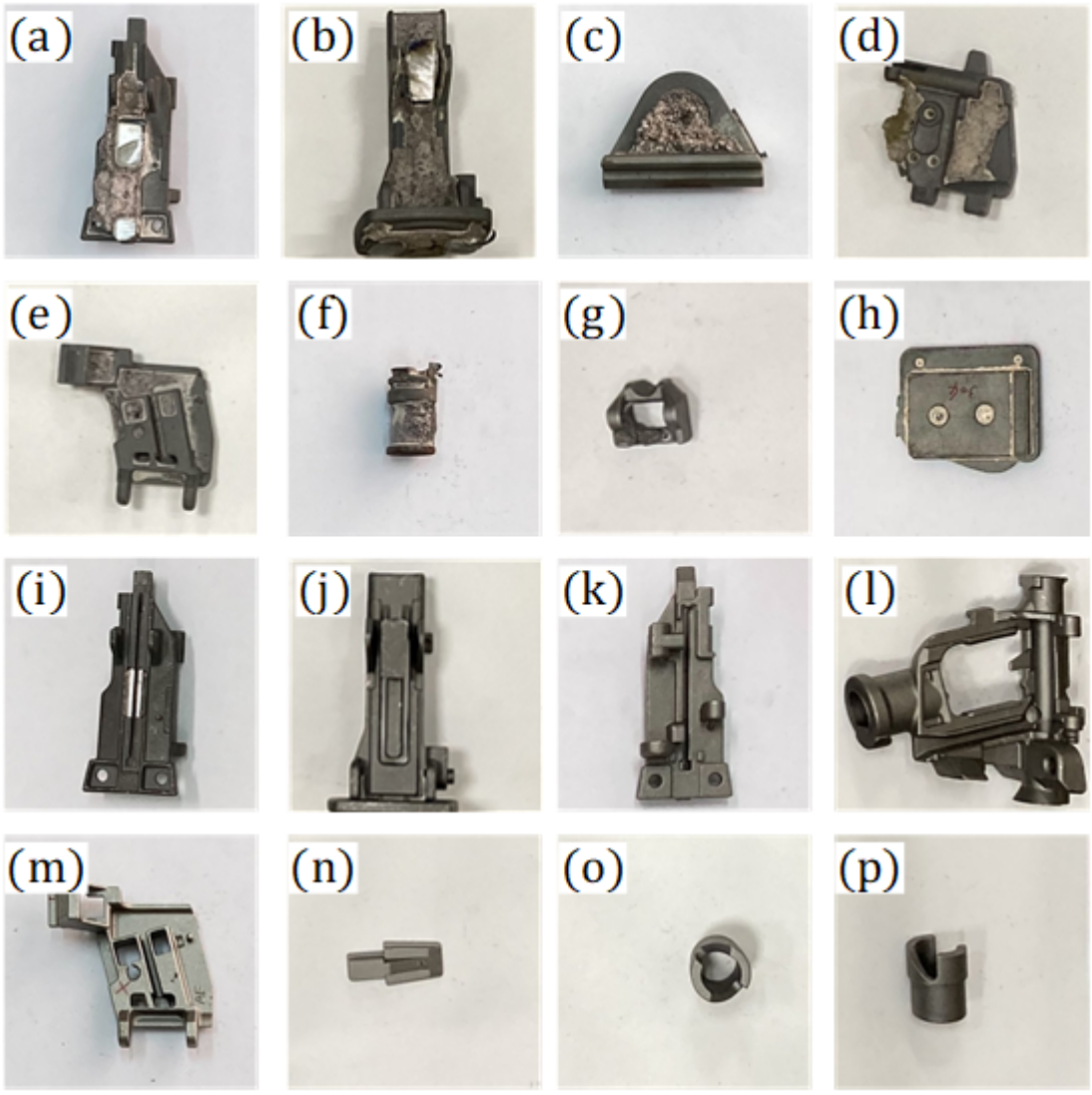

Figure 3

Examples of dataset for 16 types of investment castings: (a)-(h) defective products and (i)-(p) qualifying products

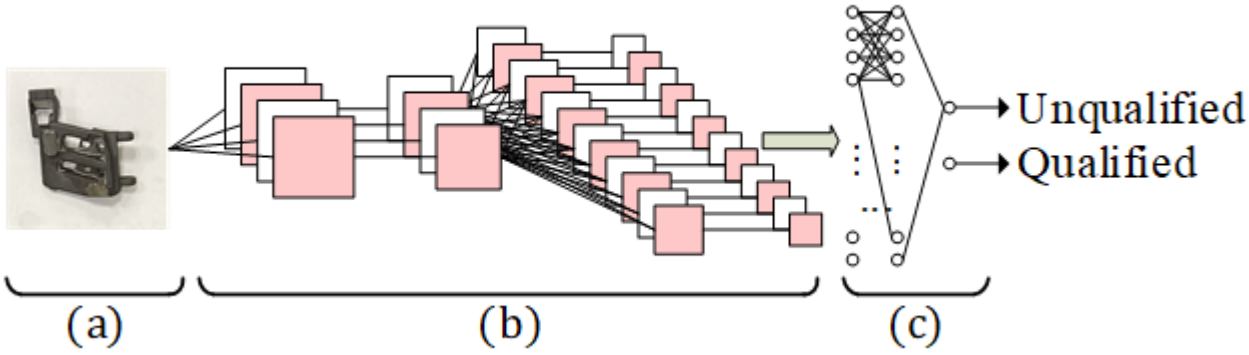

Figure 4

Network architecture of CNN image recognition: (a) input image; (b) convolutional and pooling layer; (c) fully-connected layer

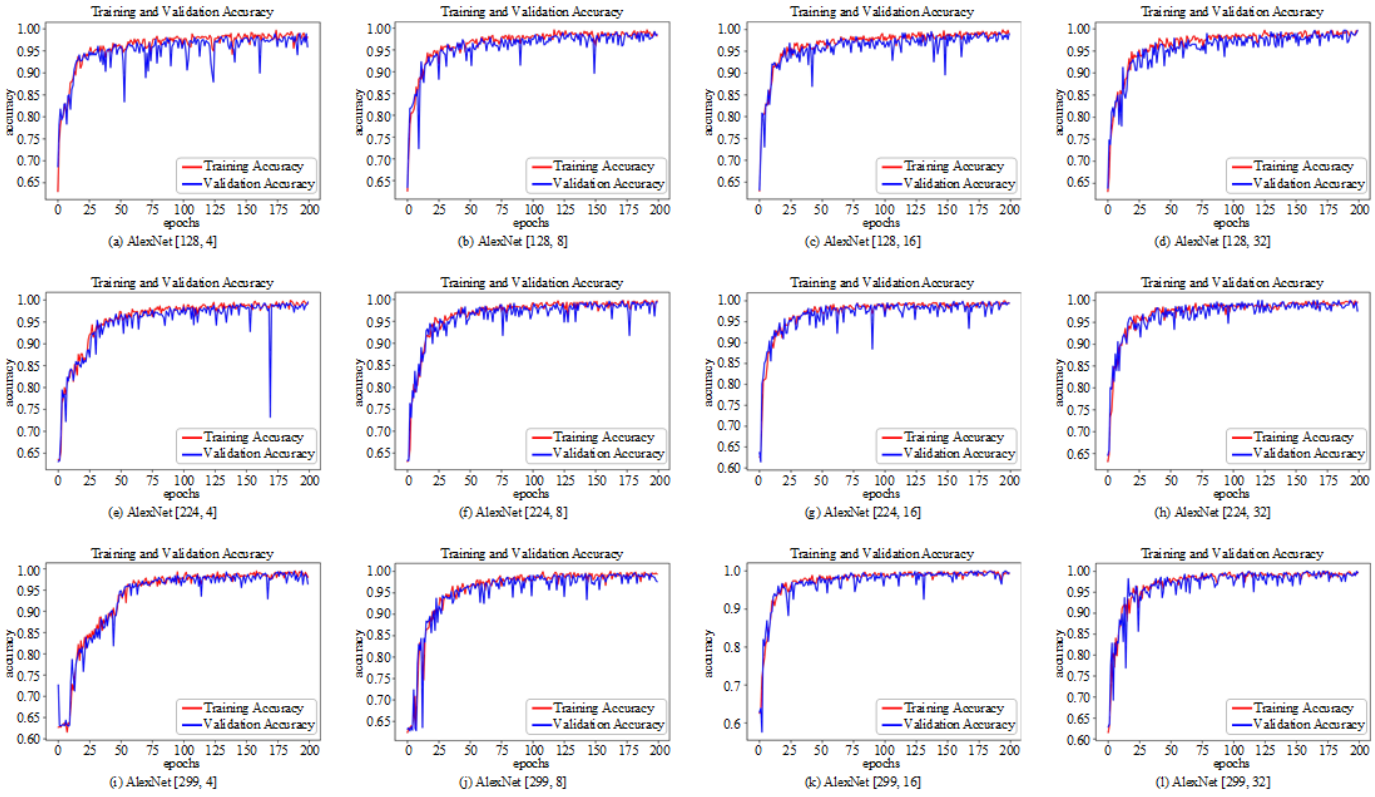

Figure 5

Historical accuracy of AlexNet model

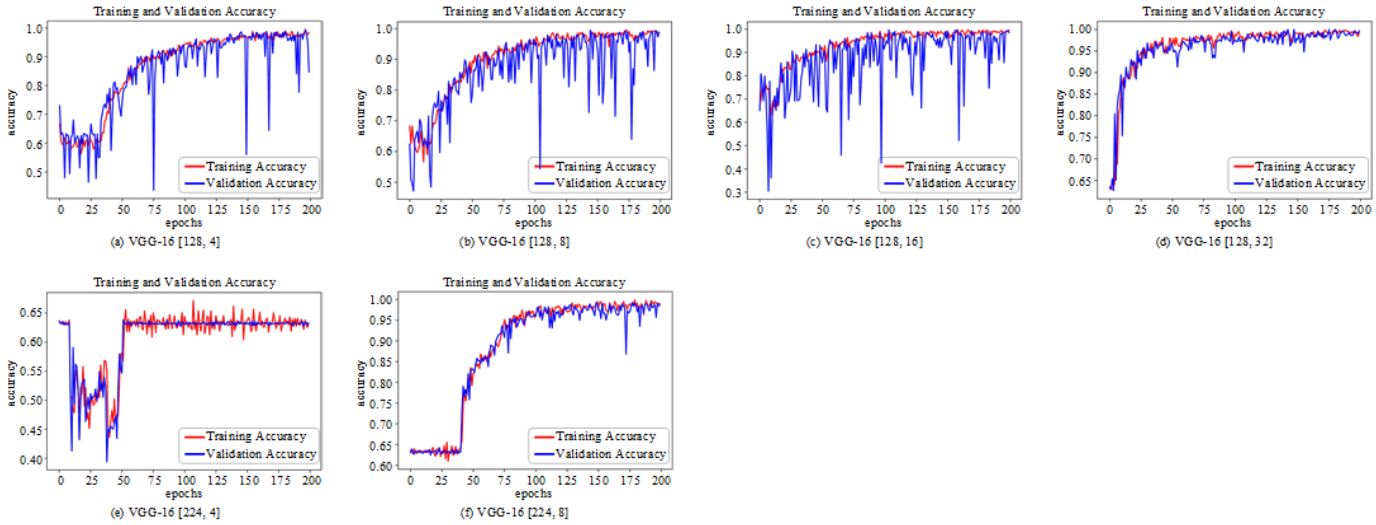

Figure 6

Historical accuracy of VGG-16 model

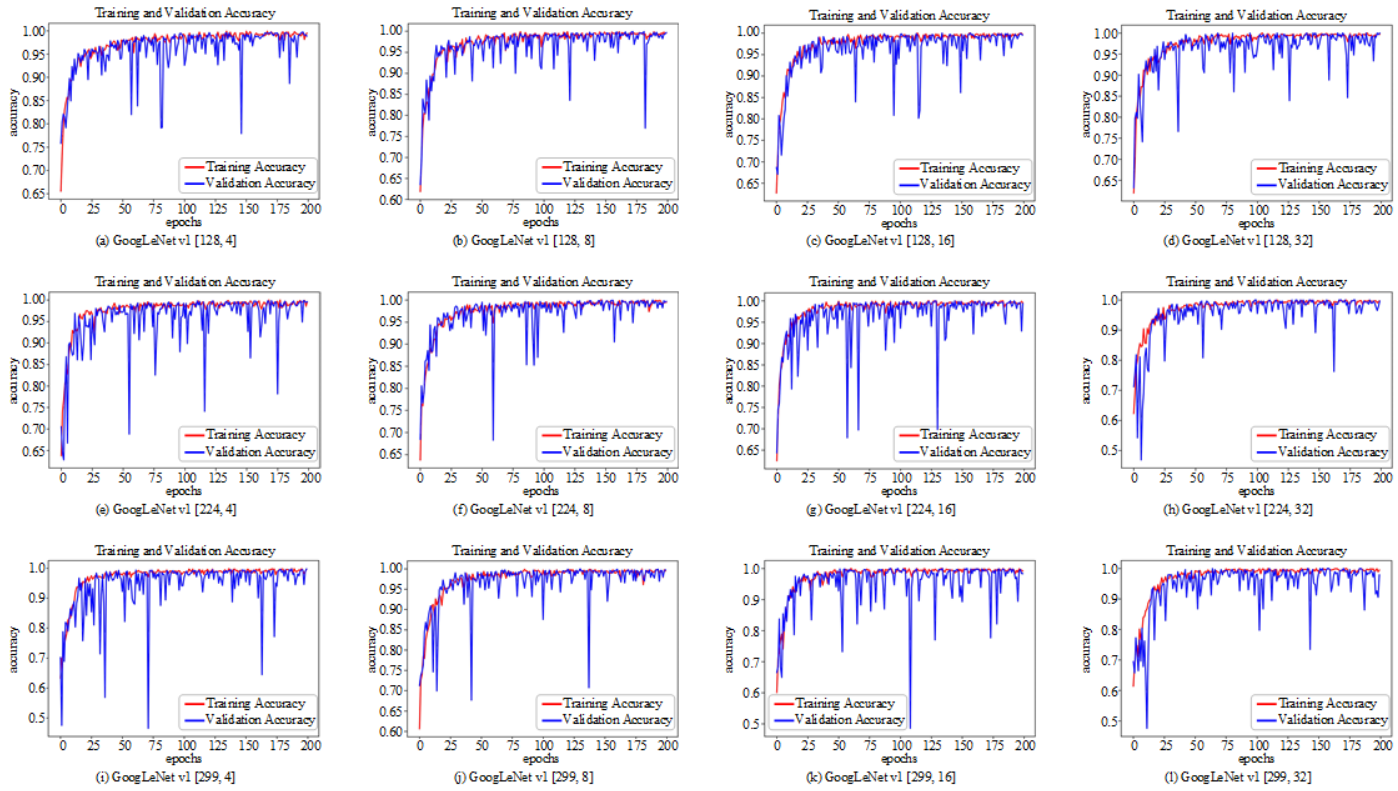

Figure 7

Historical accuracy of GoogLeNet v1 model

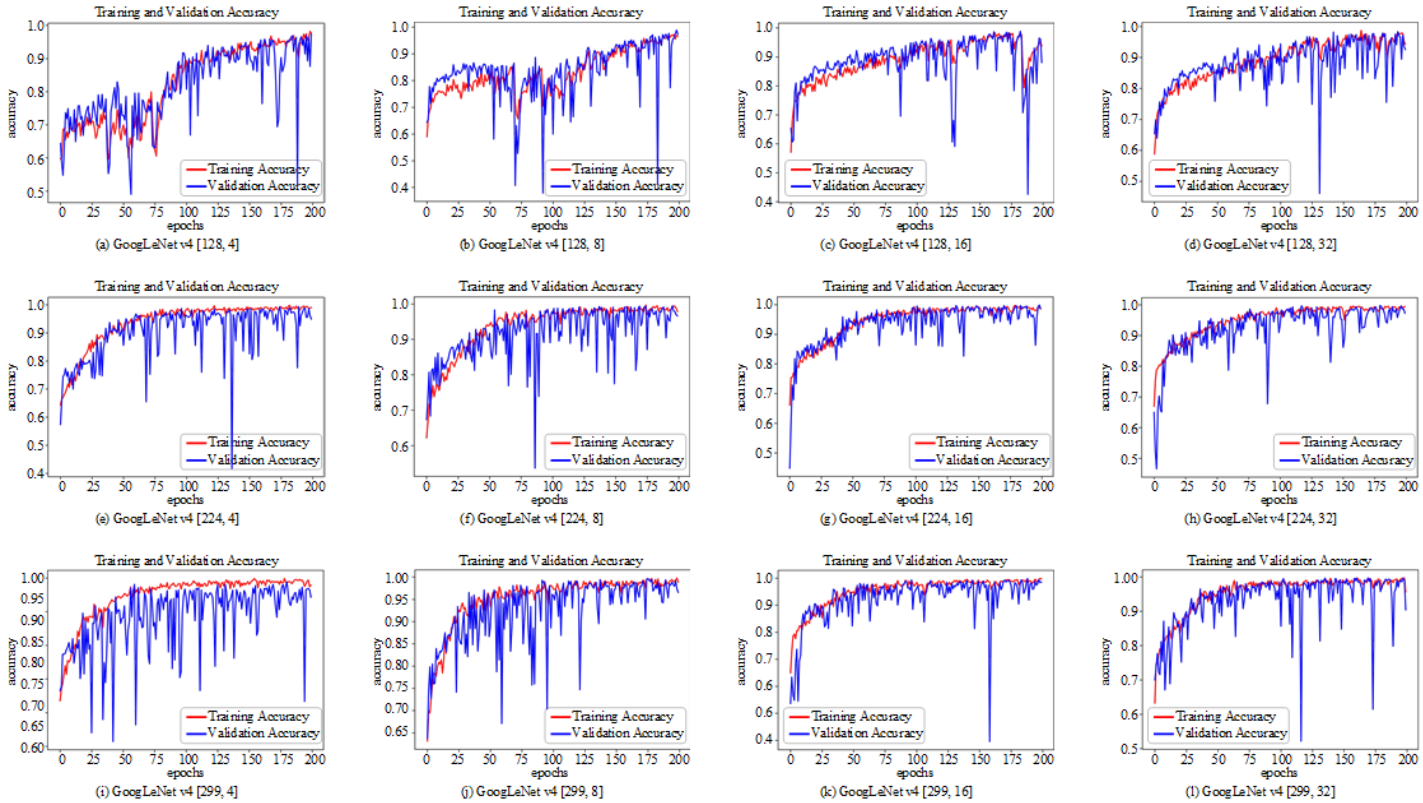

Figure 8

Historical accuracy of GoogLeNet v4 model

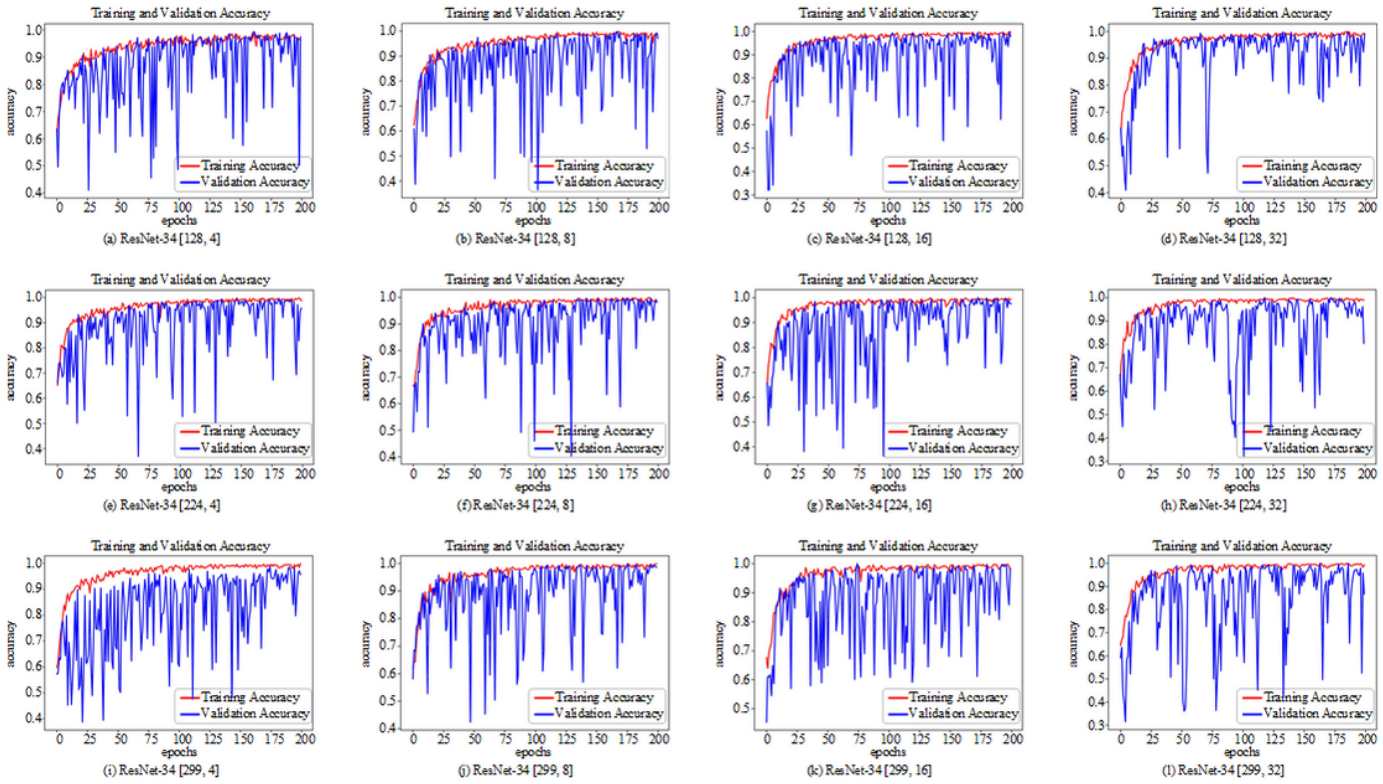

Figure 9

Historical accuracy of ResNet-34 model

Figure 10

Demonstrations of GUI: (a) casting part (orange) exceeds detection boundary; (b) “unqualified” (red) and “qualified” (green) parts