Sensors and Materials, Vol. 36, No. 2 (2024) 655–670
MYU Tokyo

S & M 3553

Vision-based Robotic Arm in Defect Detection and Object Classification Applications

Cheng-Jian Lin,[1] Jyun-Yu Jhang,[2*] Yi-Jyun Gao,[3] and Hsiu-Mei Huang[3]

[1]Department of Computer Science & Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan
[2]Department of Computer Science and Information Engineering, National Taichung University of Science and Technology, Taichung 404, Taiwan
[3]Department of Information Management, National Taichung University of Science and Technology, Taichung 404, Taiwan

(Received June 15, 2023; accepted January 12, 2024)

Keywords: robotic arm, deep learning network, object measurement, You Only Look Once (YOLO), defect detection

Robotic arms have been widely used in industrial fields. However, researchers have seldom considered the factors affecting the actual factory environment. For example, when objects are conveyed in a factory, conveyor belts are often used to dynamically plan the overall production line. In addition, each object requires multiple checkpoints for repeated audits and inspections to ensure its quality. In this study, a vision-based robotic arm system equipped with multiple functionalities was developed. The development process consisted of three steps: detecting multiple dynamic objects, determining the size of each object, and identifying object defects. In the first step, You Only Look Once was used to detect multiple dynamic objects on a conveyor belt in real time. In the second step, the original image of the object was converted into a grayscale image, and the edge contour of the object was drawn using a Canny edge detection algorithm. Objects in the image were then rotated for vertical and horizontal projections, and then an artificial neural network (ANN) was used to calculate the size of each object. In the third step, a convolutional fuzzy neural network (CFNN) was used to identify object defects. This network was divided into an input layer, a convolution pooling layer, a feature fusion layer, a fuzzy layer, a regular layer, and a defuzzification layer. According to the experimental results, the standard error of the mean between the object size obtained by the ANN and the actual size was 0.009. In addition, the accuracy, recall, precision, and F1-score obtained by the CFNN in object defect detection were 0.9580, 0.9535, 0.9535, and 0.9535, respectively. Compared with other deep neural network models, such as AlexNet and LeNet, the proposed CFNN has fewer parameters and higher performance.

[*]Corresponding author: e-mail: jyjhang@nutc.edu.tw
https://doi.org/10.18494/SAM4683

ISSN 0914-4935 © MYU K.K.
https://myukk.org/

1. Introduction

With advancements in technology, robotic arms have gradually replaced workers in repetitive tasks. Robotic arms enhance the stability and quality of production in factories, increase overall efficiency, reduce the likelihood of occupational accidents, and reduce operating costs. They also reduce the hidden damage caused by a person constantly repeating the same action.(1–3) However, some problems remain to be addressed in robotic arms, such as the need to continuously rebuild the robotic arm module in certain production environments and the difficulty of integrating robotic arms into stable production lines.(4) Despite the considerable benefits of robotic arms, the industry has remained hesitant regarding their widespread adoption and implementation. However, the manufacturing industry exhibits a strong inclination toward introducing AI technology. By harnessing the computational power of AI, companies can optimize their production lines and enhance personnel efficiency and capabilities. AI can also be integrated into robots to increase their flexibility and safety.(5)

Image recognition is a crucial component of robotic arm control. In the past, robotic arm control relied on coordinate positioning, and objects had to be placed in the same position. However, in actual industrial environments, objects are scattered and constantly changing their position. Therefore, imaging equipment must be installed on robotic arms or in the environment to determine the actual position of an object before it can be accurately grabbed.(6) Many researchers have combined robotic arms with cameras to conduct in-depth research. They used various algorithms to determine the speed and time required for object detection and identify object defects. Object defect identification can provide production line alerts to ensure high production yields.(7) Therefore, both object and defect detections are essential components in imaging technology for robotic arms.

Depending on the position of the camera, the integration of cameras and robotic arms can be classified as either internal or external. In internal integration, a camera is mounted on the robotic arm, and its field of view changes as the robotic arm moves. By contrast, in external integration, cameras are installed in the immediate environment of the robotic arm and often fixed at the same location. These external cameras have a wider field of view and are required to focus only on the error between the coordinates and the actual distance between them and the robotic arm.

Shahzad *et al.*(8) used an external dual-lens camera to plan the path of a robotic arm. However, according to Mitzias and Mertzios,(6) regardless of the type of camera used, the ultimate goal is to clearly identify objects in the work area and guide the robotic arm to grab them. The majority of robotic arms use machine kinematics, prefixed point calibration, and coordinate positioning to correctly determine their own movement path and the position of the object. Fang *et al.*(9) used local Gaussian regression without pretraining to improve the parameters of a robotic arm equipped with a camera. They reported that this technique allowed the robotic arm to rely less on kinematics and camera calibration.

Deep learning technology has been widely used in object detection to increase the accuracy of detection and shorten the time required for calculation. Because convolutional neural networks (CNNs) achieve high accuracy in object recognition,(10) multiple CNN-based architectures, such as region-based CNN (R-CNN),(11) Faster R-CNN,(12) You Only Look Once version 4 (YOLOv4),(13) and single-shot detection (SSD),(14) have been developed for object detection. These architectures have been used by researchers to enhance detection efficiency through model improvements. For example, Ji *et al.*(15) integrated a soft-CIOU loss function into a YOLOv4-based model and added preenhancement and postenhancement features to a neck structure to accurately identify the characteristics of small objects. Li *et al.*(16) proposed a bidirectional attention network to improve spatial loss and information features in path set feature pyramid networks (FPNs) to achieve performance higher than that of YOLOv5 in detecting small and multitarget objects. Butić *et al.*(17) used YOLO to detect moving objects on a conveyor belt. To effectively detect objects with a small number of samples, Xia *et al.*(18) proposed an algorithm called Attentive DropBlock to mitigate the effect of certain judgment areas, increase the overall efficiency of the YOLO model, and increase the speed and accuracy of object detection. Varna and Abromavičius(19) developed a detection system for electronic source parts that use YOLOv4 for real-time detection and classification during the dynamic process of conveyor belt transportation, and they reported excellent detection results. In conclusion, object detection can be used in both static and dynamic environments, and accuracy can be increased by modifying the architecture of deep learning models.

The emergence of Industry 4.0 has sparked increasing interest in zero-defect manufacturing, prompting the development of strategies aimed at achieving maximum zero-defect manufacturing through defect prevention and compensation.(20) Traditionally, defect detection is primarily conducted manually, which requires adequate professional knowledge and strong judgment ability. However, in industrial manufacturing environments, defect detection personnel are typically located in high-temperature or other dangerous environments, which increases the difficulty of judging defects.(21) Among the reasons underlying the difficulty of image defect detection are the lack of defect data, the presence of similar types of defect in industrial scenarios, and the difficulty of identifying defects.(22,23) In the case of an insufficient number of defect images, researchers use generative adversarial networks to generate defect map images and increase detection accuracy.(23,24) For example, to improve channel attention on YOLOv4, Li *et al.*(16) used multiple spatial pyramid pooling (SPP) to increase the efficiency of judging additive manufacturing defects. Mao *et al.*(24) used YOLOv3 and YOLOv4 to detect vehicle rim defects. Zhao *et al.*(25) used ResNet to enhance the feature extraction capability of Faster R-CNN to classify defects in vehicle parts. Chen *et al.*(26) developed a wafer defect detection network based on a CNN model to accurately identify wafer defects. Kim *et al.*(27) proposed a bidirectional convolutional recurrent reconstruction network to automatically detect welding defects with reference to time series effects. Yang *et al.*(28) detected defects in microparts. They used the size of parts and image lighting, adjusted the speed of the conveyor belt, used an SSD algorithm, and established a defect detection system to achieve an accuracy higher than that of YOLOv3. In conclusion, deep learning technology can achieve high accuracy in defect detection.

In this study, we developed a vision-based robotic arm system to detect multiple dynamic objects, determine the size of each object, and identify object defects. The major contributions of this study are as follows:

1) YOLOv4-tiny can be used to detect multiple dynamic objects on a conveyor belt in real time.
2) An artificial neural network (ANN) can be used to determine the size of an object. In this process, the original image of the object is converted into a grayscale image, and then image processing technologies such as Canny edge detection and vertical and horizontal projection are used. Finally, an ANN is used to calculate the size of the object.
3) A convolutional fuzzy neural network (CFNN) model can be used to identify object defects. In a CFNN, the fully connected network is replaced by a fuzzy neural network.
4) The rest of this paper is organized as follows. Section 2 describes the proposed vision-based robotic arm system. Section 3 presents the experimental results of object size, multiple dynamic object detection, and object defect identification obtained using an ANN and the proposed CFNN model. Finally, Section 4 presents the conclusions of this study and recommendations for future research.

2. Materials and Methods

The proposed vision-based robotic arm system is shown in Fig. 1. A six-axis Niryo Ned robotic arm equipped with a 2 MP camera and a flat-grab gripper was used to detect multiple dynamic objects, determine the size of each object, and identify object defects in a production line.

2.1 Object detection using YOLOv4-tiny

In this study, a vision-based robotic arm system was developed by simulating a factory scenario. With multiple objects placed on a conveyor belt, the robotic arm was required to detect these objects in real time and accurately pick them up. Therefore, YOLO,(13) which provides

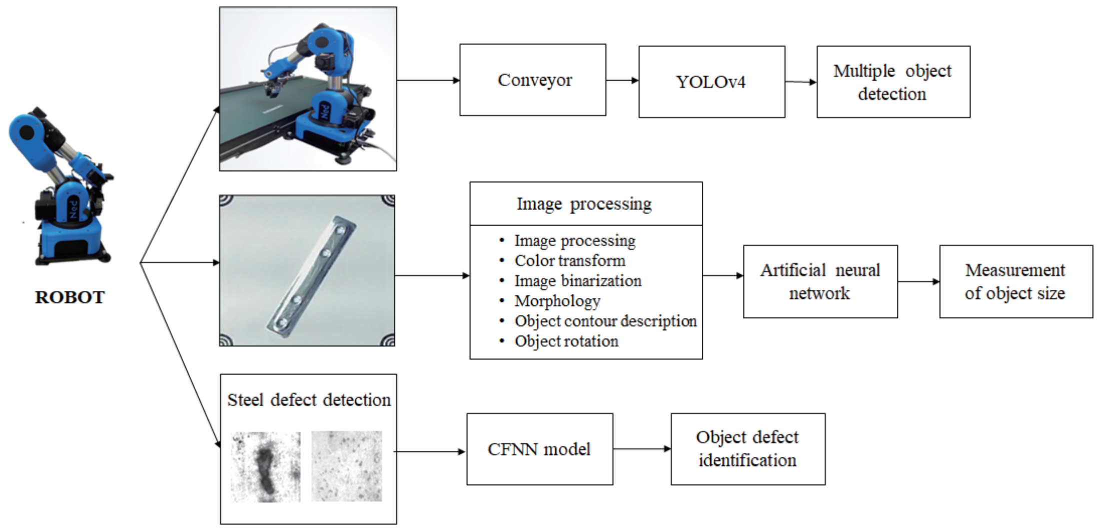

Fig. 1. (Color online) Proposed vision-based robotic arm system.

high stability and performance, was used to assist the robotic arm during the object detection process.

Overall, the applications of deep learning in detection can be categorized as either two-stage detection or single-stage detection. Although two-stage detection offers superior accuracy, it does not meet the requirements of the industry in terms of efficiency and computational power. Therefore, single-stage detectors have gained prominence in industrial defect detection.(16) YOLOv4 is a single-stage detector that is capable of target detection by extracting a single feature, which is faster than two-stage detection. Although YOLOv4 has slightly low accuracy, its residual structure allows it to effectively obtain additional feature information.

As shown in Fig. 2, the architecture of YOLOv4 is divided into three parts: a backbone, a neck, and a head. The backbone utilizes CSPDarknet53, which aims to facilitate calculations, enhance the learning ability of the model, and maintain accuracy. The neck, which is used to fuse feature information of different sizes, uses spatial pyramid pooling and a path aggregation network (PAN). This PAN combines the top-down feature fusion characteristics of FPNs and increases the bottom-up direction. The head uses the previous version of YOLOv3.

Although traditional YOLOv4 has excellent detection capabilities, its large number of network training parameters hinders its ability to effectively realize real-time detection in embedded systems. YOLOv4-tiny is a lightweight version of YOLOv4, which includes fewer YOLOv4 parameters and has a higher training speed and stronger detection capabilities, making it more suitable for use in embedded systems. In this study, we used YOLOv4-tiny to locate steel plates on a production line.

2.2 Object size measurement

In this study, the original image of an object was initially converted into a grayscale image. The object's edge outline was then drawn using a Canny edge detection algorithm. Subsequently,

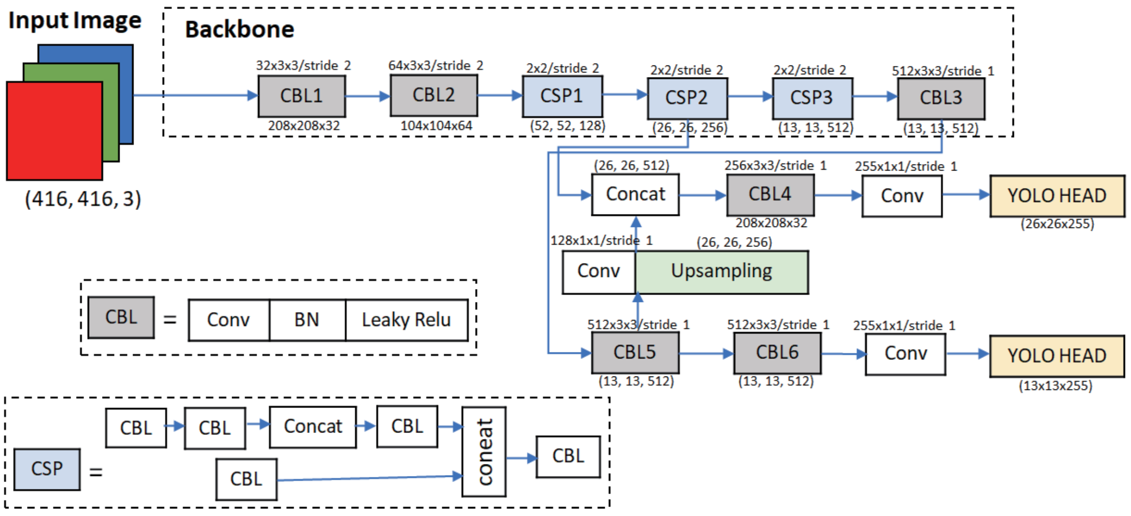

Fig. 2. (Color online) Architecture of YOLOv4-tiny.

morphology was used to clarify the image content, and the objects in the image were rotated for vertical and horizontal projections. Finally, an ANN was used to determine the actual size of the object in the image.

2.2.1 Color transformation

To set the color of an object, each specific color should be initially determined from scattered objects. To reduce the number of calculations and conform to image judgment standards in subsequent operations, each image is converted into a grayscale image. The following is the equation for color transformation:

$$RGB\left[A\right]\ to\ Gray: Y \leftarrow 0.299\ R + 0.587\ G + 0.114\ B\,. \tag{1}$$

2.2.2 Morphology

To determine the actual shape of an object in an image, the content of the object must be presented clearly through morphology. Morphology includes erosion, expansion, opening operation, and closing operation. To determine the nature of the operation, two parameters are inputted, namely, the original image and the structuring element:

$$A \ominus B = \left\{ z \middle| \left(B\right)_z \cap A^c = \varnothing \right\}, \tag{2}$$

$$A \oplus B = \left\{ z \middle| \left[\left(\hat{B}\right)_z \cap A \right] \subseteq A \right\}, \tag{3}$$

$$A \circ B = \left(A \ominus B\right) \oplus B\,, \tag{4}$$

$$A \cdot B = \left(A \oplus B\right) \ominus B\,. \tag{5}$$

2.2.3 Image binarization

Image binarization refers to setting a threshold for a grayscale image. If the gray value of an image is greater than the threshold, the image is set to white; otherwise, it is set to black. The following is the formula for image binarization:

$$dst\left(x,y\right) = \left\{ \begin{matrix} maxval & \text{if } src\left(x,y\right) > thresh \\ 0 & \text{otherwise} \end{matrix} \right\}. \tag{6}$$

2.2.4 Object rotation

If the position of an object in an image is skewed, then the size of the object cannot be calculated. Therefore, all objects in the image must be rotated to ensure that they are either vertical or horizontal. The following is the formula for object rotation:

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{7}$$

2.2.5 Edge detection

Canny edge detection algorithms(29) extract useful information from different images. They reduce the number of calculations, have a low error rate, achieve accurate positioning, and have a high resolution in processing. Canny edge detection algorithms are multistage algorithms that determine actual edges through pixel changes. To extract object edges in an image, the image is processed in grayscale, and then Gaussian blur is added to remove noise in the image. The following is the formula for edge detection:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{8}$$

Subsequently, Sobel edge detection is used to calculate the edge gradient strength and image direction. This step allows for determining the maximum gradient direction and nonmaximum suppression. It also allows the pixels on the edge of the image to be retained. Finally, a weak edge is used to connect all edges. This step facilitates the process of edge detection by setting hysteresis thresholds, which involve an upper boundary and a lower boundary:

$$G = \sqrt{G_x^2 + G_y^2}, \tag{9}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right). \tag{10}$$

2.2.5 Object projection

To obtain the area of an object in an image, objects are projected both horizontally and vertically. Our object projection results are presented in Fig. 3. This process confirms the area of the object by calculating the projected black pixels.

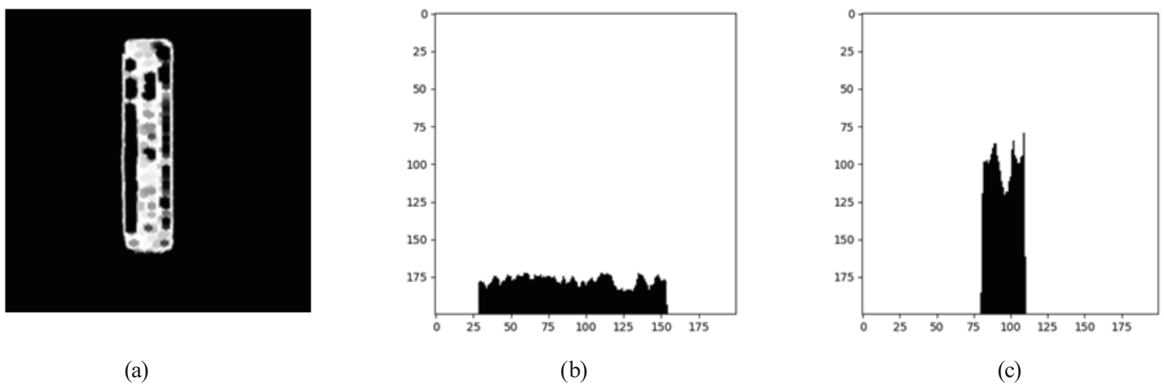

Fig. 3. Our object projection results: (a) rotated image, (b) horizontal projection results, and (c) vertical projection results.

2.2.7 Object size measurement using an ANN

ANNs are used to convert pixels obtained after horizontal and vertical projections into an actual object size. As shown in Fig. 4, an ANN is divided into three layers: an input layer, a hidden layer, and an output layer. The hidden layer utilizes an activation function to transform the input data into nonlinear features to obtain additional feature combinations. The output layer weighs the results of the hidden layer for output.

$$y_i(Z) = \text{sigmoid}\left[\sum_{i=1}^{n} X_i(Z) \times W_{ij}(Z) - \theta_j\right] \tag{11}$$

$$y_k(Z) = \text{sigmoid}\left[\sum_{j=1}^{m} X_{jk}(Z) \times W_{jk}(Z) - \theta_k\right] \tag{12}$$

2.3 Defect detection using a CFNN

In this study, a CFNN[(30)] was used to detect object defects (see Fig. 5). In a CFNN, the CNN and the pooling layer are retained, and a fuzzy neural network (FNN) is used to replace the fully connected network. FNNs simulate human logical thinking and effectively understand the correlation between input and output. An FNN is divided into an input layer, a fuzzy layer, a fuzzy rule layer, and a defuzzification layer. In the fuzzy layer, the Gaussian membership function is used to fuzzify the input data. In the rule layer, the excitation intensity of the relevant membership function of each fuzzy rule is subjected to a fuzzy AND operation. In this study, we adopted a product operation as the fuzzy AND operation. Lastly, in the defuzzification layer, a crisp value is obtained. Table 1 shows the architecture parameters of a CFNN.

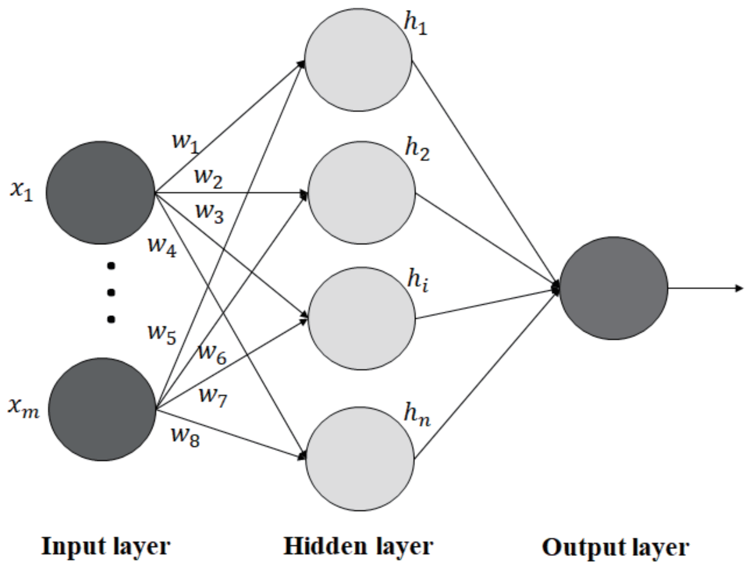

Fig. 4. Architecture of an ANN.

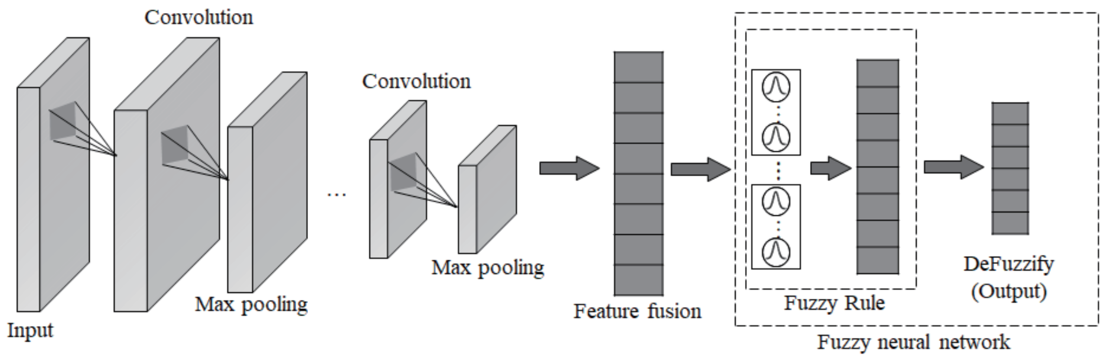

Fig. 5. Architecture of a CFNN.[30]

Table 1
Architecture parameters of a CFNN.

Layer	Kernel size	Number of filters	Stride	Units
Input	—	—	—	—
Convolution_1	3 × 3	32	1,1	—
Max_pooling_1	2 × 2	—	2,2	—
Convolution_2	5 × 5	64	1,1	—
Max_pooling_2	2 × 2	—	2,2	—
Flatten	—	—	—	—
Fuzzy rule layer	—	64	—	—
Batch normalization	—	—	—	—
Dense	—	—	—	6

2.4 Model evaluation metrics

To evaluate the performance of the proposed CFNN model in defect detection and classification, we used metrics as model evaluation criteria. Confusion matrices are often used as measurement indicators. In this study, we used precision, recall, predictive value, and F1-score as evaluation criteria:

$$Accuracy = \frac{TP + TN}{\left(TN + TP + FN + FP\right)}, \tag{13}$$

$$Recall = \frac{TP}{\left(TP + FN\right)}, \tag{14}$$

$$Precision = \frac{TP}{\left(TP + FP\right)}, \tag{15}$$

$$F1 - score = \frac{2 \times precision \times recall}{precision + recall}, \tag{16}$$

where *TP*, *TN*, *FP*, and *FN* represent true positive, false positive, true negative, and false negative, respectively.

3. Experimental Results

In this study, we simulated the scenario of a steel sheet production line. We used a robotic arm to determine the position of the steel sheet, calculate its size, and identify defects on it. Specifically, we first used YOLOv4 to determine the position of the steel sheet. Subsequently, we used image processing and an ANN to determine the size of the steel sheet. Finally, we used a CFNN to identify defects on the steel sheet.

3.1 Object detection results

We used a six-axis Niryo Ned robotic arm and YOLOv4 to conduct detection experiments on a steel sheet (Fig. 6). The experimental data contained a total of 1238 images. We applied a ratio of 8:2 for training and testing and set the learning rate to 0.001 and the number of iterations to 8000. According to the training results, the multiobject detection function of YOLOv4-tiny accurately identified the position of the steel plate, which aided in subsequent machine arm gripping and related judgments.

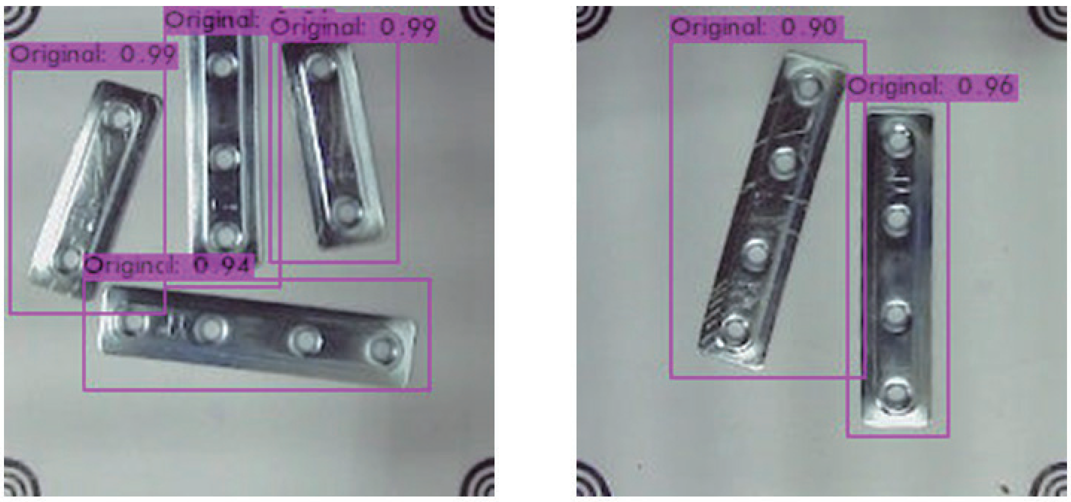

Fig. 6. (Color online) Steel sheet detection results with YOLOv4-tiny.

3.2 Object measurement results

We used image processing and an ANN to measure the length of the steel sheet. We then used the standard error of the mean (*SEM*) to determine the difference between the results obtained by the ANN and the actual length of the steel sheet. The following is the formula for the *SEM*:

$$SEM = \frac{s}{\sqrt{n}}, \tag{17}$$

where s is the sample standard deviation and n is the total number of samples.

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2} \tag{18}$$

Here, x_1 is the sample data, $\overline{x}$ is the sample average, and N is the total number of samples. Figure 7 shows the image processing results of the steel sheet.

We conducted 30 tests to measure the size of the steel sheet. Table 2 shows the results obtained by the ANN and the actual values of the steel sheet. According to the experimental results, the *SEM* was 0.009.

3.3 Object defect inspection results

To validate the proposed model, we used the NEU-DET public data set and a self-created defect data set to compare defects on our production line.

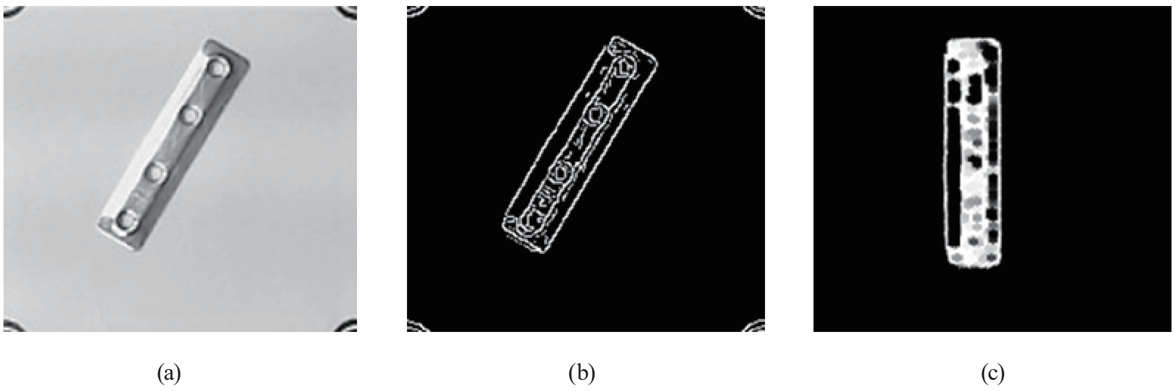

Fig. 7. Image processing results for the steel sheet: (a) original image, (b) Canny edge detection, and (c) image rotation.

Table 2
Results obtained by the ANN and the actual values of the steel sheet.

Times	Actual value		ANN		Error	
	Width	Length	Width	Length	Width	Length
1	1.39	4.81	1.403	4.844	−0.013	−0.034
2	1.39	4.81	1.403	4.784	−0.0138	0.026
3	1.39	4.81	1.285	4.725	0.105	0.085
4	1.39	4.81	1.344	4.725	0.046	0.085
5	1.39	4.81	1.463	4.903	−0.073	−0.093
6	1.39	4.81	1.403	4.784	−0.013	0.026
7	1.39	4.81	1.463	4.784	−0.073	0.026
8	1.39	4.81	1.344	4.725	0.046	0.085
9	1.39	4.81	1.403	4.844	−0.013	−0.034
10	1.39	4.81	1.463	4.725	−0.073	0.085
11	1.39	7.05	1.344	4.725	0.046	−0.047
12	1.39	7.05	1.463	4.844	−0.073	0.19
13	1.39	7.05	1.344	4.725	0.046	0.13
14	1.39	7.05	1.403	7.097	−0.013	−0.047
15	1.39	7.05	1.463	6.86	−0.073	0.19
16	1.39	7.05	1.522	6.92	−0.132	0.071
17	1.39	7.05	1.285	7.097	0.105	0.071
18	1.39	7.05	1.403	6.86	−0.013	0.071
19	1.39	7.05	1.463	6.979	−0.073	0.012
20	1.39	7.05	1.463	6.979	−0.073	0.012
21	1.39	9.85	1.463	6.979	−0.073	0.012
22	1.39	9.85	1.522	7.038	−0.132	0.083
23	1.39	9.85	1.463	7.038	−0.073	0.024
24	1.39	9.85	1.403	9.767	−0.013	0.202
25	1.39	9.85	1.522	9.826	−0.132	0.024
26	1.39	9.85	1.344	9.648	0.046	0.083
27	1.39	9.85	1.403	9.826	−0.013	0.083
28	1.39	9.85	1.344	9.767	0.046	0.024
29	1.39	9.85	1.403	9.767	−0.013	0.083
30	1.39	9.85	1.344	9.826	0.046	0.024

3.3.1 NEU-DET data set

The NEU-DET data set[31] has been cited by many studies investigating the performance of various classifiers. As shown in Table 3, this data set has 1800 photos divided into six categories of steel surface defects, namely, crazes, inclusions, patches, pitted surfaces, rolled-in scales, and scratches. All images in the data set are grayscale images with a size of 200 × 200 pixels (Fig. 8).

In our experiments, we used a ratio of 8:2 for training and testing and set the number of iterations to 100. Our experimental results indicated that the accuracy, recall, precision, and F1-score of the proposed CFNN were 0.9594, 0.9577, 0.9609, and 0.9549, respectively, thereby outperforming AlexNet and LeNet models. Table 4 shows the comparison results of various models in the NEU-DET data set.

Table 3
NEU-DET data set.

Defect categories	Crazes	Inclusions	Patches	Pitted surfaces	Rolled-in scales	Scratches
Training	240	240	240	240	240	240
Testing	60	60	60	60	60	60

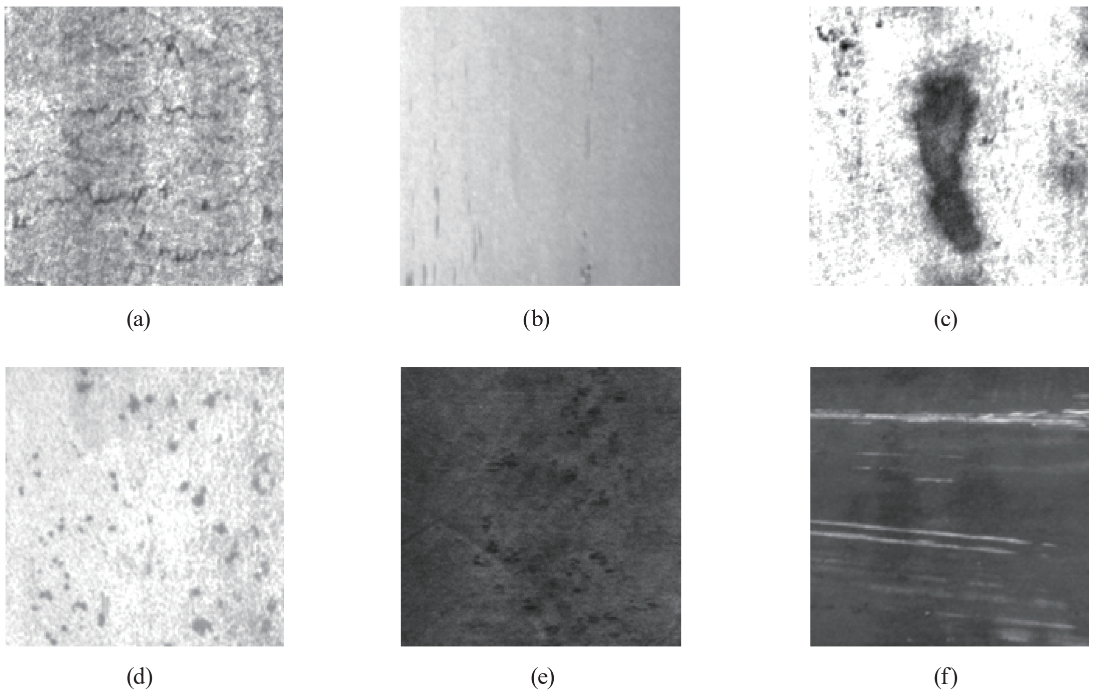

Fig. 8. Images of defects from the NEU-DET data set: (a) crazes, (b) inclusions, (c) patches, (d) pitted surfaces, (e) rolled-in scales, and (f) scratches.

3.3.2 Self-created defect data set

To simulate an actual factory environment, we gathered data on steel sheet defects for subsequent classification. As shown in Table 5, we collected 2400 images and divided steel sheets into two categories: normal steel sheets and defective steel sheets (scratches). As shown in Fig. 9, all images in the data set were grayscale images with a size of 200 × 200 pixels. Table 6 shows the defect detection results of the proposed CFNN and LeNet and AlexNet models. The results indicated that the accuracy, recall, precision, and F1-score of the proposed CFNN exceeded those of the AlexNet and LeNet models.

Table 4
Comparison results of various models in the NEU-DET data set.

	Accuracy	Recall	Precision	*F*1-score
AlexNet	0.8299	0.8198	0.8230	0.8168
LeNet	0.7716	0.7527	0.7671	0.7413
CFNN	0.9594	0.9577	0.9609	0.9549

Table 5
Comparison of normal and defective steel sheets.

Categories	Normal steel sheets	Defective steel sheets
Training	960	960
Testing	240	240

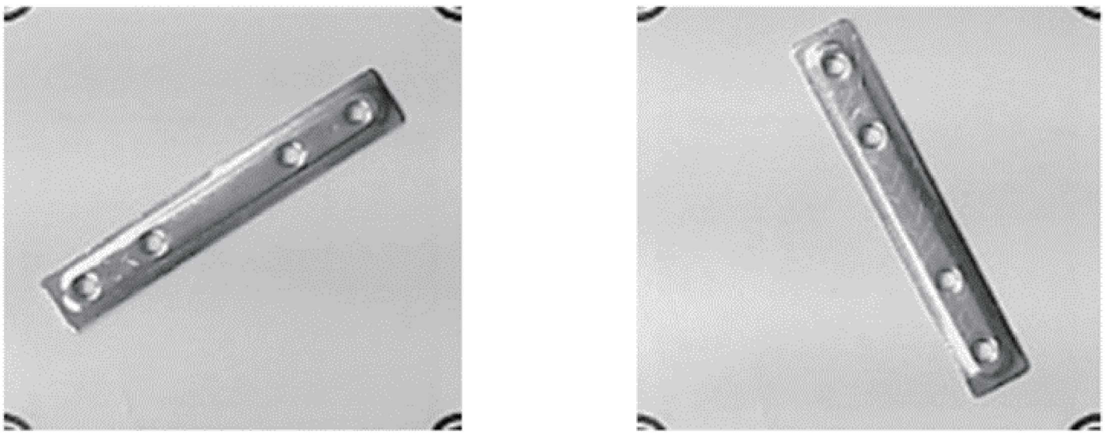
(a) (b)

Fig. 9. (a) Normal and (b) defective steel sheets.

Table 6
Comparison of various models in our self-created defect data set.

	Accuracy	Recall	Precision	*F*1-score
AlexNet	0.9340	0.9293	0.9293	0.9293
LeNet	0.9440	0.9417	0.9417	0.9417
CFNN	0.9580	0.9535	0.9535	0.9535

4. Conclusions

In this study, we developed a vision-based robotic arm system to detect multiple dynamic objects, determine the size of each object, and identify object defects. Specifically, we used YOLOv4-tiny to detect multiple dynamic objects on a conveyor belt in real time. We then used an ANN to determine the size of each object and a CFNN to identify object defects. According to the experimental results, the *SEM* between the size obtained by the ANN and the actual size was 0.009. In addition, the accuracy, recall, precision, and F1-score obtained by the CFNN in object defect detection were 0.9580, 0.9535, 0.9535, and 0.9535, respectively. Compared with other deep neural network models, such as AlexNet and LeNet, the proposed CFNN involved fewer parameters and had higher accuracy, recall, precision, and F1-score.

To obtain distinct experimental results, we set various parameters in the ANN, CFNN, and YOLOv4-tiny. In future studies, we intend to use certain parameter optimization methods, such as the Taguchi, uniform distribution, and Bayesian optimization hyperparameter methods, to address this problem.

Acknowledgments

This work was supported by the National Science and Technology Council of the Republic of China under Grant MOST 110-2221-E-167-031-MY2 and Grant NSTC 111-2222-E-025-001-.

References

1. J. Dixon, B. Hong, and L. Wu: Manage. Sci. **67** (2021) 9. https://doi.org/10.1287/mnsc.2020.3812(20)
2 D. Maturana and S. Scherer: 2015 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS, 2015) 922–928. https://doi.org/10.1109/IROS.2015.7353481
3 C. Heyer: 2010 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS, 2010) 4749. https://doi.org/10.1109/IROS.2010.5651294
4 M. Afrin, J. Jin, A. Rahman, Y. C. Tian, and A. Kulkarni: Future Gener. Comput. Syst. **97** (2019) 119. https://doi.org/10.1016/j.future.2019.02.062
5 W. L. Chien, Y. C. Ko, and J. H. Wang: Int. J. Multinational Corporation Strategy **3** (2020) 2. https://doi.org/10.1504/IJMCS.2020.114690
6 D. A Mitzias and B. G. Mertzios: Measurement **36** (2004) 3. https://doi.org/10.1016/j.measurement.2004.09.008
7 M. Rezaei-Malek, M. Mohammadi, J. Y. Dantan, A. Siadat, and R. Tavakkoli-Moghaddam: Int. J. Prod. Res. 57 (Taylor, 2019) 15-16. https://doi.org/10.1016/j.rcim.2021.102229
8 A. Shahzad, X. Gao, A. Yasin, K. Javed, and S. M. Anwar: IEEE Access **8** (2020) 203158. https://doi.org/10.1109/ACCESS.2020.3037540
9 G. Fang, X. Wang, K. Wang, K. H. Lee, J. D. Ho, H. C. Fu, and K. W. Kwok: IEEE Rob. Autom. Lett. **4** (2019) 2. https://doi.org/10.1109/LRA.2019.2893691
10 M. Liang, and X. Hu: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2015) 3367–3375. https://doi.org/10.1109/CVPR.2015.7298958
11 R. Girshick, J. Donahue, T. Darrell, and J. Malik: Proc. IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2014) 580–587. https://doi.org/10.48550/arXiv.1311.2524
12 S. Ren, K. He, R. Girshick, and J. Sun: Adv. Neural Inf. Process **28** (2015) 1. https://doi.org/10.48550/arXiv.1506.01497
13 A. Bochkovskiy, C. Y. Wang, and H. Y. Mark Liao: arXiv preprint arXiv:2004 (2020) 10934. https://doi.org/10.48550/arXiv.2004.10934
14 W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg: European Conf. Computer Vision **17** (Springer, 2016) 3. https://doi.org/10.1007/978-3-319-46448-0_2

15 S. J. Ji, Q. H. Ling, and F. Han: Comput. Electr. Eng. **105** (2023) 108490. https://doi.org/10.1016/j.compeleceng.2022.108490
16 W. Li, H. Zhang, G. Wang, G. Xiong, M. Zhao, G. Li, and R. Li: Rob. Comput. Integr. Manuf. **80** (2023) 102470. https://doi.org/10.1109/TASE.2020.2967415
17 F. Butić, A. Radovan, M. Pajas, and B. Mihaljević: 2022 45th Jubilee Int. Conv. Information, Communication and Electronic Technology (MIPRO, 2022) 1485–1488. https://doi.org/10.23919/MIPRO55190.2022.9803704
18 R. Xia, G. Li, Z. Huang, H. Meng, and Y. Pang: Pattern Recognit. Lett. **165** (2023) 91. https://doi.org/10.1109/TASE.2020.2967415
19 D. Varna and V. Abromavičius: Appl. Sci. **12** (2022) 11. https://doi.org/10.3390/app12115608
20 D. Powell,M. C.Magnanini, M. Colledani, and O. Myklebust: Comput. Ind. **136** (2022) 103596. https://doi.org/10.1016/j.compind.2021.103596
21 C. Laofor, and V. Peansupap: Autom. Constr. **24** (2012) 160. https://doi.org/10.1016/j.autcon.2012.02.012
22 Y. Gao, X. Li, X. V. Wang, L. Wang, and L. Gao: J. Manuf. Syst. **62** (2022) 753. https://doi.org/10.1016/j.jmsy.2021.05.008
23 S. Niu, B. Li, X. Wang, and H. Lin: IEEE Trans. Autom. Sci. Eng. **17** (2020) 3. https://doi.org/10.1109/TASE.2020.2967415
24 W. L. Mao, Y. Y. Chiu, B. H. Lin, C. C. Wang, Y. T . Wu, C. Y. You, and Y. R. Chien: Sensor **22** (2022) 10. https://doi.org/10.3390/s22103927
25 B. Zhao, M. Dai, P. Li, R. Xue, and X. Ma: IEEE Access **8** (2020) 136808. https://doi.org/10.1109/ACCESS.2020.3009654
26 X. Chen, J. Chen, X Han, C. Zhao, D. Zhang, K. Zhu, and Y. Su: IEEE Access **8** (2020) 24006. https://doi.org/10.1109/ACCESS.2020.2970461
27 Y. M. Kim, I. U. Yoon, H. Myung, and J. H. Kim: IEEE Access **9** (2021) 135316. https://doi.org/10.1109/ACCESS.2021.3116799
28 J. Yang, S. Li, Z. Wang, and G. Yang: IEEE Access **7** (2019) 89278. https://doi.org/10.1109/ACCESS.2019.2925561
29 J. Canny: IEEE Trans. Pattern Anal. Mach. Intell. **6** (1986) 679. https://doi.org/10.1109/TPAMI.1986.4767851
30 C. J. Lin and J. Y. Jhang: IEEE Access **10** (2022) 14120. https://doi.org/10.1109/ACCESS.2022.3147866
31 Y. He, K. Song, Q. Meng, and Y. Yan: IEEE Trans. Instrum. Meas. **69** (2019) 4. https://doi.org/10.1109/TIM.2019.2915404