Sensors and Materials, Vol. 36, No. 4 (2024) 1431–1439
MYU Tokyo

S & M 3609

Clamp Dot Image Classification Using Neural Network

Krittapak Srikam and Anakkapon Saenthon*

Department of Robotics and AI Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, 1, Soi Chalongkrung 1, Chalongkrung Street, Ladkrabang, Bangkok 10520, Thailand

(Received January 29, 2024; accepted March 5, 2024)

Keywords: CNN, OpenCV, image processing, TensorFlow, neural network

In this paper, we discuss the classification of images captured by a machine camera while assembling components. To crop out specific points of interest, we employ image processing. Additionally, we utilize deep learning techniques, specifically convolutional neural networks, to identify the type of equipment being assembled. This approach allows us to determine and record specific parts within a device. However, the main challenge of this project is to achieve both high accuracy and the shortest possible prediction time.

1. Introduction

In the electronic industry, one of the methods of assembling a product is the clamp mount method, and screws are used to tighten the platter stack in the product. Each clamp used is unique to each manufacturer. In the past, the clamp manufacturer was not specified during the assembly stage. When customers provide feedback regarding issues of the clamp installation process, it becomes challenging to determine which manufacturer assembled the product. Therefore, production team consider incorporating image processing[(1)] techniques and machine learning to enhance the manufacturing process. However, we found problems with image processing and machine learning. For image processing, too many methods are used (e.g., threshold,[(2)] xor, floodfill, and Convex Hull). The image captured by the machine shows clamp stacking, which can be seen through another clamp. Such problems could impact the original image, potentially altering its appearance, and affecting assembling time during production. For machine learning, the present model was trained only on images that detect the dots. Thus, it is difficult to clarify all images and the prediction result will be inaccurate if there are not enough images relative to the number of manufacturers.

To further enhance the efficiency of the manufacturing process, we propose to perform image reprocessing and utilize convolutional neural networks (CNNs)[(3–7)] in this study, we achieve an accuracy above 95%, maintain a model file size of less than 20 MB, and process each image for a prediction time of less than 30 ms.

*Corresponding author: e-mail: anakkapon.sa@kmitl.ac.th
https://doi.org/10.18494/SAM5000

ISSN 0914-4935 © MYU K.K.

2. Data, Materials, and Methods

2.1 Data collection

The overall images of eight manufacturers are shown in Fig. 1. Images were captured by a machine camera and saved in a folder that is categorized by labels with the names of different manufacturers. This study includes a total of 13,625 images that were taken during the production process. The size of each image is 640 × 480 pixels.

To determine the clamp of a manufacturer, count the dots starting from the timing mark and count the dots clockwise including the white space, as shown in Fig. 2.

Table 1 shows the specifications of the dot locations of eight manufacturers; black circles refer to dots and white circles refer to the absence of dots.

2.2 Image processing

Image processing refers to operations applied to an image to enhance or transform it. The input is an image, and the output can be another image or features related to that image, such as

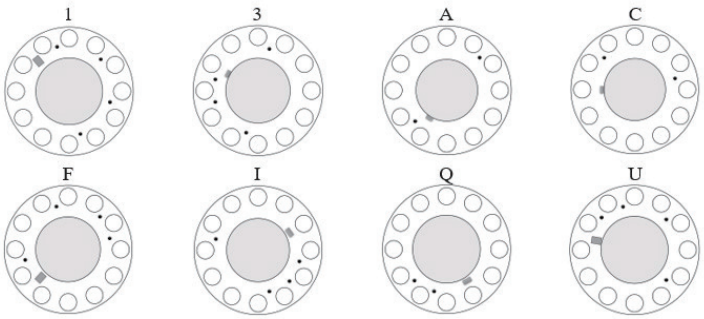

Fig. 1. (Color online) Images of eight classes (manufacturers).

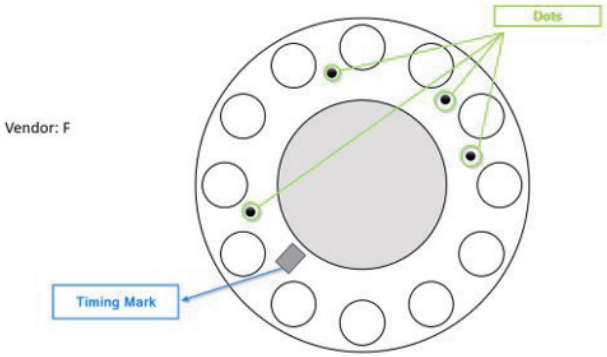

Fig. 2. (Color online) Sample dots and timing mark.

Table 1
Specifications of dot locations of eight manufacturers.

Manufacturer	Dot location											
	1	2	3	4	5	6	7	8	9	10	11	12
1	●	○	●	○	●	○	●	○	○	○	○	○
3	●	○	●	●	○	○	●	○	○	○	○	○
A	●	○	○	○	○	○	●	○	○	○	○	○
C	●	○	○	○	●	○	○	○	○	○	○	○
F	●	○	○	●	○	●	●	○	○	○	○	○
I	●	●	●	○	○	○	●	○	○	○	○	○
Q	●	●	○	○	○	○	○	○	○	○	○	○
U	●	●	○	●	●	○	●	○	○	○	○	○

pixel values, vectors, or RGB colors. In this study, we utilized the Open-Source Computer Vision Library (OpenCV),[7] which is an open-source library. OpenCV is a powerful tool for computer vision and is useful in machine learning applications. It enables us to process images and videos, allowing us to perform tasks such as object detection and facial recognition. Specifically, we used OpenCV to process images using various techniques such as adaptive histogram equalization,[8] contrast-limited adaptive histogram equalization (CLAHE),[8] HoughCircles, and findContours.[9] These methods allow us to crop regions of interest[9] and reduce image size.

With our latest image processing technology, we have revolutionized the reprocessing of images. As shown in Fig. 3, our new method requires significantly fewer steps compared with the previous method. This translates into quicker and more efficient image reprocessing, saving valuable time and resources.

The problem with capturing images of machines is that one may accidentally capture the lower layer of another clamp when clamps are stacked in a slot before assembling the product, as shown in Fig. 4.

2.3 CNN model

The CNN model is widely used for image classification, and we have chosen three pre-trained CNN models that could work well with our data. We have also created a custom model with a basic architecture to test the data's validity.

2.3.1 Custom model

CNN architecture is designed to extract features from complex visual data using specialized layers within the network. It is composed of three fundamental layer types.

The first is the convolution layer, which extracts features from input data. It uses a set of learnable filters called kernels applied to the image, with each kernel specifying specific characteristics that the convolution operation aims to filter. In this study, we utilized the Conv2D layer.

The second is the pooling layer, which serves to reduce the dimensionality of the input and decrease the number of parameters. Pooling filters do not possess weights. In this article, we utilize the MaxPooling2D layer.

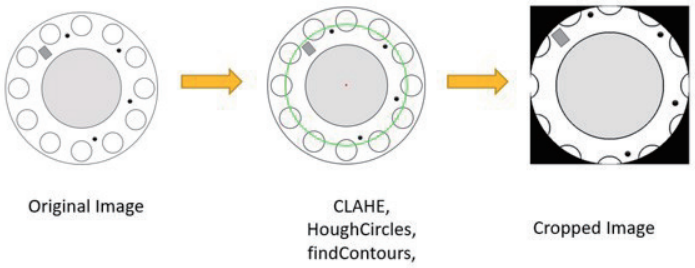

Fig. 3. (Color online) Image processing steps.

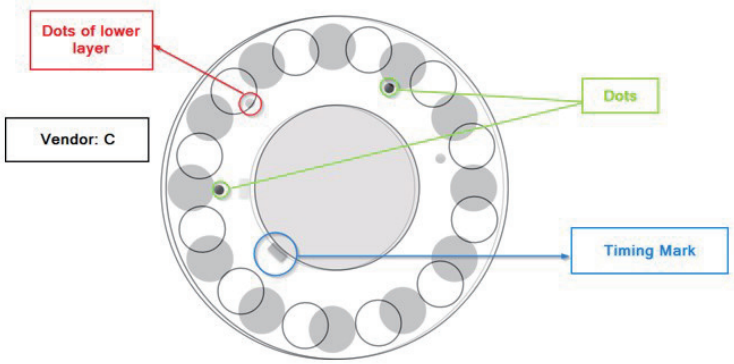

Fig. 4. (Color online) Problem of capturing images from a machine, which can fail to detect lower-layer dots.

Finally, the fully connected layer, also known as the dense layer, of the CNN model, is connected to the previous layer and computes the last classification.

In this study we use a custom model architecture 2-layered CNN model, as shown in Table 2. The model includes two CNN layers, with the first layer being an input CNN layer with a shape of 110 × 110 × 3. All the layers use rectifier linear unit activation (ReLU).[10] The max pool layer with a filter size of 2 × 2 is fed with the output of the first and second CNN layers. The image kernel size is set to 4 × 4, with a required padding and stride of one. The output of the 2D CNN layers is flattened, and the extracted features are then fed to a block of two fully connected layers with ReLU activation and SoftMax,[11] as shown in Fig. 5.

In the training log, the model training was observed to be effective for eighty epochs, as evident from the accuracy and loss training results. However, beyond this point, the model started to overfit, as illustrated in Fig. 6. To compare the variation of validation loss and accuracy over the training steps, the data in Fig. 6 was analyzed. The results showed that as the validation loss decreased, the validation accuracy increased subsequently.

2.4 TFLite converter

TFLite Converter[12] is a TensorFlow tool that converts machine learning models to a mobile and embedded device-friendly format. TFLite is a lightweight TensorFlow version optimized for

Table 2
Custom model architecture.

Layer	Output Shape	Parameter
rescaling_5 (Rescaling)	(None, 110, 110, 3)	0
conv2d_9 (Conv2D)	(None, 107, 107, 16)	784
max_pooling2d_8 (MaxPooling 2D)	(None, 53, 53, 16)	0
conv2d_10 (Conv2D)	(None, 50, 50, 32)	8224
max_pooling2d_9 (MaxPooling 2D)	(None, 25, 25, 32)	0
flatten_5 (Flatten)	(None, 20000)	0
dense_10 (Dense)	(None, 64)	1280064
dense_11 (Dense)	(None, 8)	520

Total parameters: 1289592
Trainable parameters: 1289592
Non-trainable parameters: 0

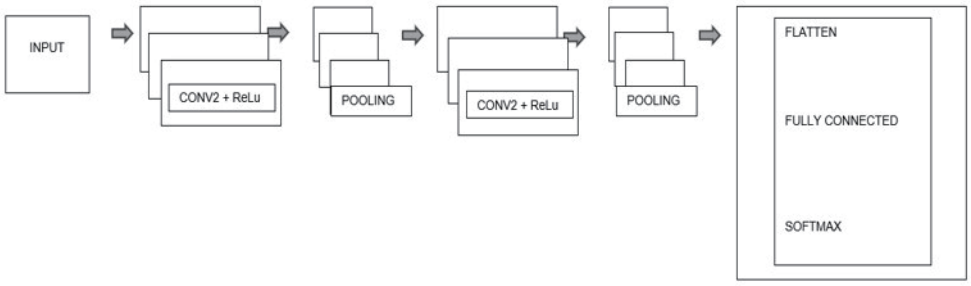

Fig. 5. (Color online) Two-layered CNN model.

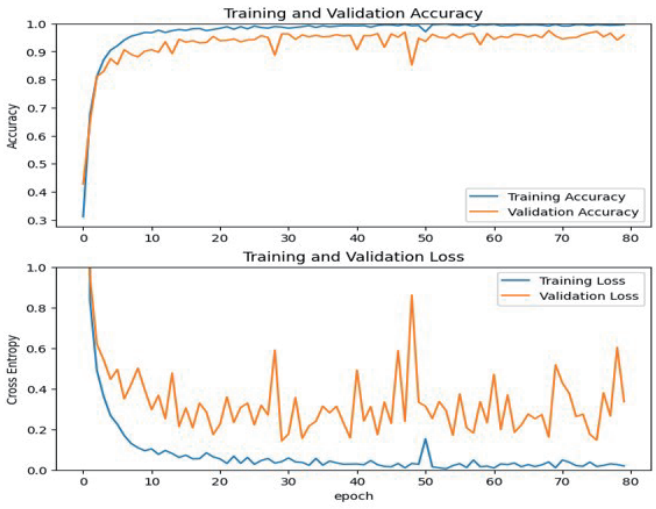

Fig. 6. (Color online) Accuracy and loss of training phase.

these devices. It enables machine learning models to be accessible and usable on a wider range of devices.

In this study, we utilized TensorFlow Lite Converter to minimize the model size and processing time, although this may result in a lower model accuracy. The flow of TFLite conversion is shown in Fig. 7.

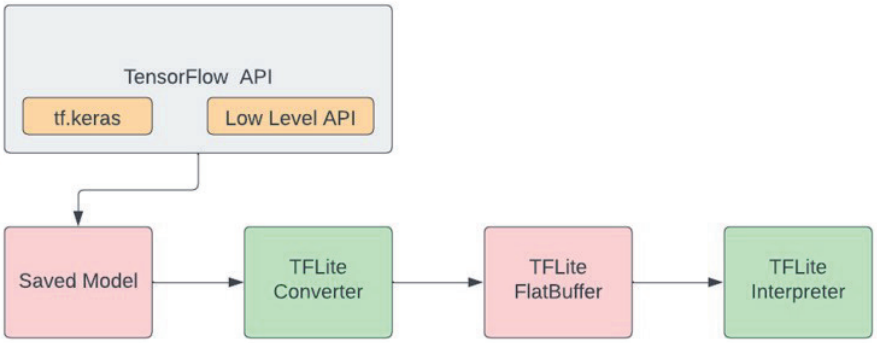

Fig. 7. (Color online) Flow of TFLite conversion.

3. Experimental Results

We tested four different CNN models using 20% of 13,625 images. After analyzing the results of the experiment, we observed that the custom model had the highest accuracy among the models considered. However, its prediction time was still longer than the desired objective of less than 30 ms Additionally, the prediction time of the model with the second highest accuracy, ResNet50,[13–15] was also very high, as shown in Table 3.

3.1 Improvement of result

As shown in Table 4, we converted the model using the TFLite Converter method and revalidated it. The accuracy remained unaffected; however, the prediction time of the custom model significantly decreased from 47.1 to 6.1 ms (87% reduction). Additionally, the model file size decreased from 14 to 5 MB, achieving a 64% reduction. Both improvements align with the objectives of this study and surpass those of the other three models.

3.2 Evaluation of model performance

To evaluate the performance of the four models, we used a confusion matrix[16] with four quadrants: true positive (TP),[17] false positive (FP),[17] true negative (TN),[17] and false negative (FN),[17] as shown in Fig. 8.

To calculate the accuracy based on TP, FP, TN, and FN values, the equation below is used.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

The calculation results shown in Table 5 included precision, recall, and the F1-score to measure the performance of the classification models.

Table 3
Comparison of five models.

Model	Accuracy (%)	Prediction time (ms)/image	File size (MB)
Custom model	97.05	47.1	14
VGG16	89.63	79.8	60
ResNet50	90.52	78.9	104
BiMobileNetV2	82.20	89.7	23
Previous model	77.38	14.8	6

Table 4
Comparison of five converted TFLite models.

Model	Accuracy (%)	Prediction time (ms)/image	File size (MB)
Custom model	97.05	6.1	5
VGG16	89.63	154.1	63
ResNet50	90.52	66.4	95
BiMobileNetV2	82.20	35.7	13
Previous model	77.38	14.8	6

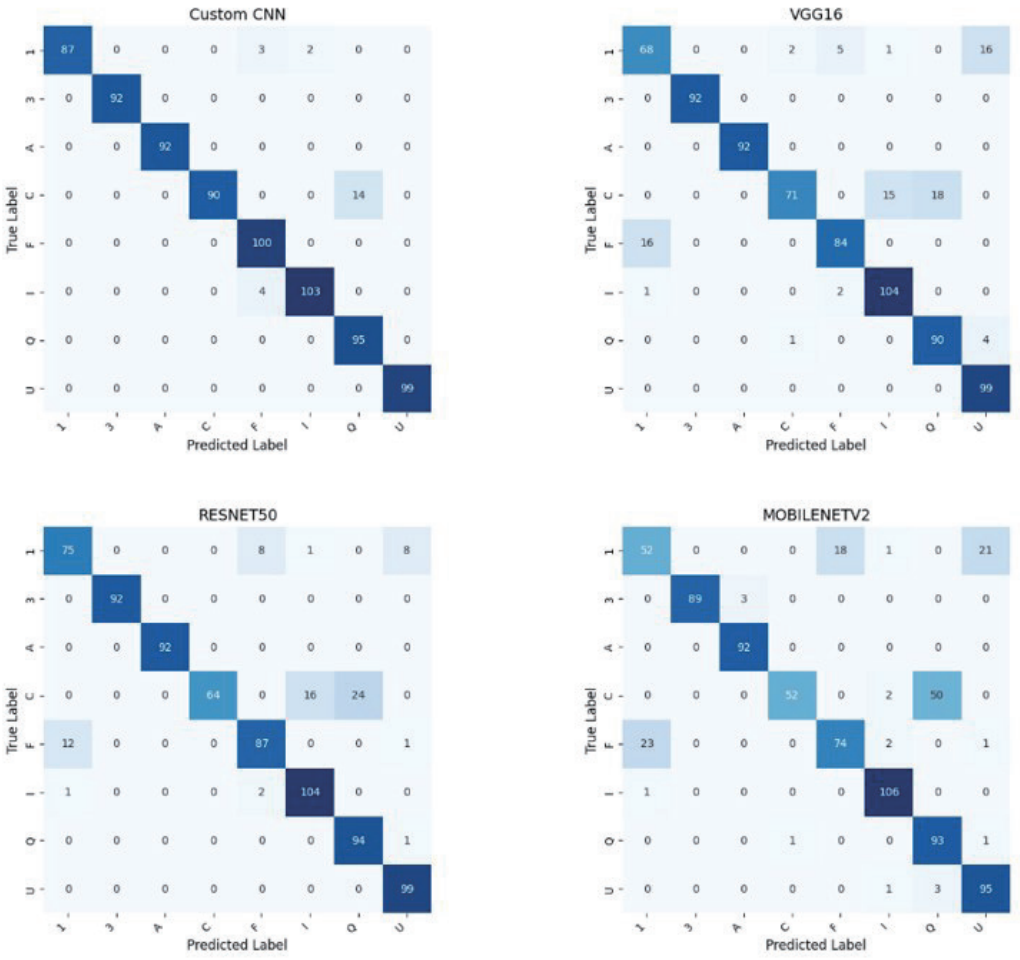

Fig. 8. (Color online) Confusion matrix of four CNN models.

Table 5
Comparison of confusion matrices of four models.

Model	TP	FP	FN	Precision	Recall	F1-score	Accuracy
Custom model	758	19	4	0.9755	0.9948	0.9851	0.9706
VGG16	700	61	20	0.9198	0.9722	0.9453	0.8963
ResNet50	707	59	15	0.9230	0.9792	0.9503	0.9052
BiMobileNetV2	653	99	29	0.8684	0.9575	0.9107	0.8361

4. Conclusions

To enhance the accuracy and prediction time of image classes during product assembly, we revised the image processing method to retain the original production images. This approach helps avoid missing dot detection and employs a more efficient technique. Additionally, we improved the model performance by utilizing CNN models. The accuracy rates for the custom model, VGG16, ResNet50, and BiMobileNetV2 were 97.05, 89.63, 90.52, and 82.20%, respectively; the previous model's accuracy was 77.38%. The prediction times of the custom model, VGG16, ResNet50, and BiMobileNetV2 were 6.1, 154.1, 66.4, and 35.7 ms, respectively.

In future studies, we can explore image processing techniques for object detection. This is particularly relevant because clamp manufacturers can exhibit a higher complexity than simple dots.

Acknowledgments

This work was financially supported by the School of Engineering, King Mongkut's Institute of Technology Ladkrabang Research Fund: (65-02-01-049).

References

1 A. Wanga, W. Zhangb, and X. Weia: Comput. Electron. Agric. **158** (2019) 1. https://doi.org/10.1016/j.compag.2019.02.005
2 Y. Zou, J. Zhang, M. Upadhyay, S. Sun, and T. Jiang: IEEE Access. **8** (2020) 17217. https://doi.org/10.1109/ACCESS.2020.3024718
3 C. Lin, C. Wu C. Sun and C. Lin: Sens. Mater. **10** (2020) 3137. https://doi.org/10.18494/SAM.2020.2771
4 S. K. Shetty, A.a Siddiqa: Int. J. Eng. **7** (2019) 195. https://doi.org/10.26438/ijcse/v7i7.195201
5 Y. Chen, G. Li, S. Jhong, P. Chen, C. Tsai, and P. Chen: Sens. Mater. **32** (2020) 3157. https://doi.org/10.18494/SAM.2020.2838
6 S. S. Yadav, S. M. Jadhav: J. Big Data **6** (2019) 1. https://doi.org/10.1186/s40537-019-0276-2
7 Mo. G. Galety, F. H. Almukhtar, and R. J. Maaroof: Conf. 2022 8th Int. Conf. Smart Structures and Systems (ICSSS) (IEEE, 2022) 1–6.
8 Y. Chen, Q. Fang, H. Tian, S. Li, Z. Song, and J. Li: Sens. Mater. **34** (2022) 1043. https://doi.org/10.18494/SAM3780
9 Y. Chen, C. Tsai, F. Ding, and Q. Hsu: Sens. Mater. **33** (2021) 4087. https://doi.org/10.18494/SAM.2021.3251
10 L. Chen, S. Li, Q. Bai, and J. Yang: Remote Sens. **13** (2021) 1. https://doi.org/10.3390/rs13224712
11 J. Qin, W. Pan, X. Xiang, Y. Tan, and G.Hou: Ecol. Inf. **58** (2020) 1. https://doi.org/10.1016/j.ecoinf.2020.101093
12 H. Kim, W. Jung, Y. Park, Ja. Lee, and S. Ahn: Expert Syst. Appl. **188** (2022) 1. https://doi.org/10.1016/j.eswa.2021.116014
13 P. Wang, H. Tseng, T. Chen, and C. Hsia: Sens. Mater. **33** (2021) 2299. https://doi.org/10.18494/SAM.2021.3277

14 L. Alzubaidi, J. Zhang, A. J. Humaidi, and A. Al-Dujaili: J. Big Data **8** (2021) 1. https://doi.org/10.1186/s40537-021-00444-8
15 Q. Zhou, Z. Huang, and M. Ding: IEEE J. Biomed. Health. Inf. **27** (2023) 1991. https://doi.org/10.1109/JBHI.2023.3241439
16 S. B. Rosende, J. FERNÁNDEZ-ANDRÉS, and J. SÁNCHEZ-SORIANO: IEEE Access **11** (2023) 104593. https://doi.org/10.1109/ACCESS.2023.3316618
17 A. Tayal, J. Gupta, A. Solanki, K. Bisht, A. Nayyar, and M. Masud: Multimedia Syst. **28** (2022) 1417. https://doi.org/10.1007/s00530-021-00769-7