19th IMEKO TC10 Conference
"MACRO meets NANO in Measurement for Diagnostics, Optimization and Control"
Delft, The Netherlands, September 21–22, 2023

Multi-task Learning Based on Deep Convolutional Neural Networks for Surface Defect Detection of Metal Gaskets

Chao-Ching Ho[1,2], Chun-Han Liu[2], Ming-Fu Chen[3], Ming-Chieh Kao[4]

[1] *Graduate Institute of Manufacturing Technology and Department of Mechanical Engineering, National Taipei University of Technology, Taipei, Taiwan, HoChao@ntut.edu.tw*
[2]*Graduate Institute of Automation Technology National Taipei University of Technology, Taipei, Taiwan, t110618021@ntut.org.tw*
[3]*Taiwan Instrument Research Institute, NARLab. mfchen@narlabs.org.tw*
[4]*Metal Industries Research and Development Centre, Kaohsiung, Taiwan, ming0910@mail.mirdc.org.tw*

***Abstract* – This paper proposes a multi-task learning system for industrial defect detection, focusing on surface scratches on titanium gaskets as the detection target. Our model is built using a deep convolutional neural network (CNN). As defects are infrequent in actual production lines, we adopt the Faster-RCNN object detection network architecture and integrate it with a multi-task learning system. A classification task model is introduced into the backbone network to filter out a significant number of defect-free images. This step reduces the computation and data transmission time of the subsequent target detection task, accelerating the detection process. Experimental results show that our proposed method reduces inference time by 37.6% and 42.46% in actual production lines with defect rates of 12% and 5%, respectively, while maintaining 96% of the original model's performance.**

Keywords – Deep Convolutional Neural Networks, Multi-task Learning, Object Detection, Classification, Defect Detection, Metal Gaskets.

I. INTRODUCTION

Defect detection is an essential task in industrial production lines. Traditional methods for detecting defects rely on manual inspection, which is both time-consuming and expensive. However, with the development of deep learning technology, the application of convolutional neural networks (CNN) in defect detection tasks has achieved remarkable results. Object detection networks can simultaneously detect the position, quantity, and class of multiple target objects in an image, providing higher flexibility and generality compared to classification networks.

In actual production lines, defective products are often a small proportion of all products. Even with the use of deep learning technology, the detection efficiency is still limited by the processing of non-defective images using conventional object detection networks. Therefore, this paper proposes a multi-task learning system that introduces a classification task model into the backbone network. The system uses the deep layer features generated by the front part of the object detection network for the classification task. Only when the classification task model determines that an image contains defects, the proposed network will pass the features to the back part of the network for the detection of the location and category of defect targets.

II. RELATED RESULTS IN THE LITERATURE

Deep learning is a machine learning method based on artificial neural networks that can learn features and relationships from a large amount of input data. This method is widely used in industrial defect detection because it can quickly identify defects in a large amount of image data. Deep learning models typically include multi-layer neural networks that can learn different features and relationships through training data. In industrial defect detection, these models learn how to identify various types of defects [6], such as cracks, scratches, and stains.

Convolutional Neural Network (CNN) is a type of deep learning technology that is particularly suitable for image recognition, classification, and semantic segmentation tasks [5]. It uses convolutional and pooling layers to capture features in images and performs classification through multi-layer neural networks.

Multitask Learning [10] is a deep learning technology that aims to enable a model to solve multiple tasks simultaneously. It has better performance than single-task learning and can be used as one of the methods to improve model performance by sharing training data and model parameters. Y. Sun et al. [1] used a deep multitask learning architecture to detect fine-grained building changes. This model can use data from all tasks and learn more features while solving multiple tasks. It is particularly effective when tasks have similar features or there is correlation between tasks. It can also improve the model's generalization performance to some extent when the training data is limited.

Edge computing is a computing approach that runs on edge devices, and users do not need to connect to servers for deep learning model inference. This method can reduce the latency caused by transmission and reduce data privacy issues. However, due to the limited computing power of edge devices, how to allocate resources between cloud computing and edge computing is crucial. J.C. Lee et al. [2] proposed a split-target detector architecture that supports edge-cloud collaboration for high-throughput inference. Y. Matsubara et al. [7] introduced bottleneck structures [3] into the model to reduce the latency caused by a large amount of data transmission and used knowledge distillation to ensure the model's performance while reducing the transmission data.

III. DESCRIPTION OF THE METHOD

3.1 Multi-task learning model

This paper proposes a multi-task learning model as a combination of a classification task model and an object detection task model.

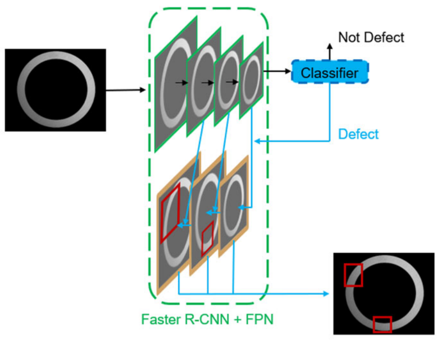

Fig. 1. Multi-task learning model

3.1.1 Classification task

ResNet [9] is a type of deep convolutional neural network (CNN) that is commonly designed for tasks such as image classification and object detection. It is mainly composed of convolutional layers, residual layers, and fully connected layers. The convolutional layers are used to extract low-level features from images, such as contours, edges, and colors. The feature that sets ResNet apart from other CNN models is its use of a shortcut connection structure, which contains two branches. One branch passes the input x across layers, while the other is F(x). The two branches are added together and then passed through an activation function, a process known as residual learning. This technique helps to solve the problem of model degradation. In this paper, we use the ResNet pre-trained model provided by Pytorch as the classifier.

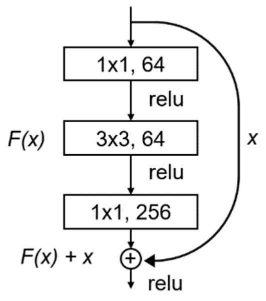

Fig. 2. ResNet50 bottleneck block

layer name	output size	50-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4_x	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5_x	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 100-d fc, softmax
FLOPs		3.8×10^9

Table. 2. ResNet50 architecture

The loss function is a method used to measure the difference between predicted and actual results. When the predicted and actual results are inconsistent, the value of the loss function increases. Commonly used loss functions for classification tasks include cross-entropy loss (1), mean square error (MSE) (2), and so on. The training process of the model involves minimizing the loss function to make the predicted result as close as possible to the actual result.

$$H(p,q) = -\sum_x p(x) log q(x) \qquad (1)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (2)$$

For the classification task in this paper, we use accuracy (3) as the evaluation metric. This metric measures the consistency between the classifier's predicted results and the actual results. The accuracy is calculated by dividing the number of samples correctly predicted by the model by the total number of samples. The accuracy value usually ranges from 0 to 1, and the closer it is to 1, the better the performance of the classifier.

$$Accuracy = \frac{TP+TN}{TP+FP+TNFN} \quad (3)$$

3.1.2 Object detection task

Faster R-CNN [8] is an object detection algorithm developed based on R-CNN and Fast R-CNN. It introduces the Region Proposal Network (RPN) to achieve faster detection speed and higher accuracy. To better handle targets of different scales, Faster R-CNN also introduces the Feature Pyramid Network (FPN) structure. FPN is composed of two main parts: bottom-up feature extraction and top-down feature propagation. Bottom-up feature extraction leverages a convolutional neural network to extract feature maps, while top-down feature propagation fuses the deep feature maps and low-level feature maps, making the feature maps rich in feature information at multiple scales. After FPN processing, the feature pyramid of the original image is mapped to the feature pyramid, enabling object detection on feature maps of different scales. Moreover, since the mapped feature maps contain feature information of different depths, this helps to improve the accuracy of the model.

Faster R-CNN supports pre-trained deep neural networks such as VGG and ResNet as backbone networks for training. In this paper, pre-trained ResNet50 is used as the backbone network, and the deepest feature map of FPN is extracted for multi-task learning classification.

The loss function of RPN is represented as follow (4):

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i (p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p^* L_{reg}(t_i,t_i^*) \quad (4)$$

which includes classification loss (5) and box regression loss (6). Specifically, RPN predicts whether the anchors contain targets through binary classification and uses cross entropy as the loss function. For positive samples, the difference between the predicted bounding box and the actual bounding box is calculated, and the regression loss is computed using the smooth L_1 loss function. Faster R-CNN uses the cross entropy loss function to perform multi-class prediction on positive samples and further predict the target class.

$$L_{cls}(p_i,p_i^*) = -\log[p_i^* p_i + (1-p_i^*)(1-p_i)] \quad (5)$$

$$L_{reg}(t_i,t_i^*) = smooth_{L_1}(t_i - t_i^*) \quad (6)$$

Here, p_i is the predicted probability that anchor i corresponds to an object. p_i^* is the ground truth label. is a vector of the predicted bounding box coordinates. N_{cls} and N_{reg} are parameters for normalization.

RPN predicts whether an anchor contains an object or not with binary classification using cross-entropy as the loss function. For positive anchors, it calculates the difference between the predicted and actual bounding boxes and computes regression loss using the Smooth L1 loss function. Faster R-CNN uses cross-entropy loss function for multi-class prediction on positive anchors to further predict the target class.

3.2 Training

The model was trained using a dataset consisting 540 images of surface scratches and 200 images of defect-free surfaces, totaling 740 images. The images were of size 2448×2048 and were labeled for classification task. Additionally, the 540 images of scratched surfaces were annotated for object detection using the PASCAL VOC standard. The training parameters are listed in Table 2.

Table 2. Training parameters.

Epochs	200 epochs
Input image size	2448×2048 pixels
Train images	432
Validation images	108
Batch size	4
Optimizer	SGD
Learning rate	0.01
StepLR	step=3, gamma=0.33

3.3 Evaluation metrics

This paper utilizes mean Average Precision (mAP) (7) and Average Recall (AR) (8) as performance evaluation metrics for our model.

$$Precision = \frac{TP}{TP+FP} \quad (7)$$

$$Recall = \frac{TP}{TP+FN} \quad (8)$$

Where, TP is the number of true positive, FP is the number of false positive, FN is the number of false negative.

IV. RESULTS AND DISCUSSIONS

All the experiments in this paper were completed using the PyTorch deep learning framework on an Ubuntu 22.04 system. The hardware setup consisted of an Intel® Core™ i9-10980XE CPU, two NVIDIA GeForce RTX 3090 24GB GPUs, and 128GB of memory.

The learning curves of the proposed model and Faster R-CNN, as shown in Figure 1. Faster R-CNN achieved an mAP of 0.455 on the titanium gaskets surface scratches dataset, while the proposed model achieved an mAP of 0.437.

For the experiment of reducing inference time, datasets with defect image proportions of 5%, 12%, and 50% were respectively used to calculate the ratio of model inference time and reduced inference time. The figure shows the average time required to detect an image in datasets with different defect ratios and the ratio of reduced inference time in datasets with different defect ratios.

The experimental results show that while maintaining 96% performance of Faster R-CNN, the average time required to detect an image in datasets with defect image proportions of 5%, 12%, and 50% were 3.04ms, 6.45ms, and 25.7ms, respectively. The ratio of reduced inference time was 42.46%, 37.6%, and 21.28%, respectively.

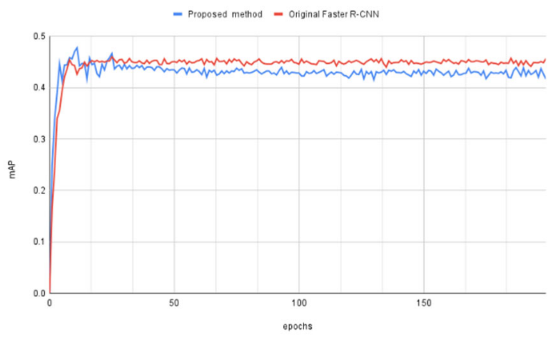

Fig. 3. Learning curves of Faster R-CNN and proposed model

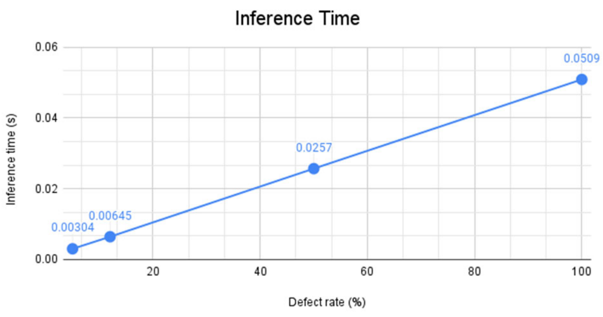

Fig. 4. Inference time per image under different defect rate

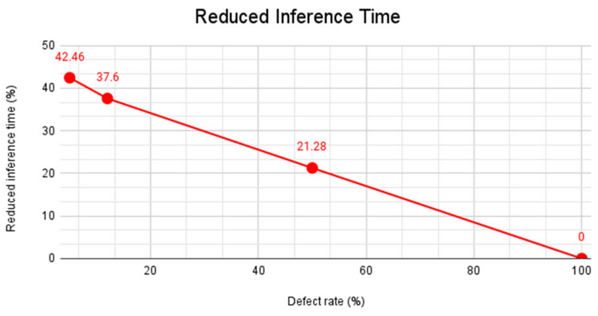

Fig. 5. Reduced inference time under different defect rate

V. CONCLUSIONS AND OUTLOOK

This paper proposes a multi-task learning system for industrial defect detection, with a particular focus on detecting surface scratches on titanium gaskets using a Faster-RCNN object detection network architecture. Our experimental results show that our proposed method significantly reduces inference time, with an average time required to detect an image in datasets with defect image proportions of 5%, 12%, and 50% being 3.04ms, 6.45ms, and 25.7ms, respectively. Furthermore, the ratio of reduced inference time was 42.46%, 37.6%, and 21.28%, respectively, compared to the Faster-RCNN model. These findings demonstrate the effectiveness of our proposed model in improving the efficiency of industrial defect detection while maintaining the performance of the Faster-RCNN model.

REFERENCES

[1] **Y. Sun, X. Zhang, J. Huang, H. Wang and Q. Xin.**: Fine-grained building change detection from very high-spatial-resolution remote sensing images based on deep multitask learning, IEEE Geosci. Remote Sens. Lett, vol. 19, 2022.

[2] **J. C. Lee, Y. Kim, S. Moon and J. H. Ko.**: A splittable DNN-based object detector for edge–cloud collaborative real-time video inference, *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveillance (AVSS)*, pp. 1-8, 2021.

[3] **J. Shao and J. Zhang.**: BottleNet++: An end-to-end approach for feature compression in device-edge co-inference systems, *Proc. IEEE Int. Conf. Commun. Workshop*, pp. 1-6, 2020.

[4] **J. H. Ko, T. Na, M. F. Amir and S. Mukhopadhyay.**: Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms, *Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, pp. 1-6, 2018

[5] **Y. Zhou, G. G. Yen and Z. Yi.**: Evolutionary compression of deep neural networks for biomedical image segmentation, *IEEE Trans. Neural Netw. Learn. Syst*, vol. 31, no. 8, pp. 2916-2929, Aug. 2020.

[6] **Y. He, K. Song, Q. Meng and Y. Yan.**: An end-to-end steel surface defect detection approach via fusing multiple hierarchical features, *IEEE Trans. Instrum. Meas*, vol. 69, no. 4, pp. 1493-1504, Apr. 2020.

[7] **Y. Matsubara, D. Callegaro, S. Baidya, M. Levorato and S. Singh.**: Head Network Distillation: Splitting Distilled Deep Neural Networks for Resource-Constrained Edge Computing Systems, *IEEE Access*, vol. 8, pp. 212 177-212 193, 2020.

[8] **S. Ren, K. He, R. Girshick and J. Sun.**: Faster R-CNN: Towards real-time object detection with region proposal networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137-1149, Jun. 2016.

[9] **K. He, X. Zhang, S. Ren and J. Sun.**: Deep residual learning for image recognition, *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 770-778, Jun. 2016.

Wang, Li.: Modeling Industrial ADMET Data with Multitask Networks, *arXiv.org*, 2016.