A Light-Weight Defect Detection System for Edge Computing

Hsiang-Ting Huang, Tzu-Yi Chiu, and Chia-Yu Lin
Department of Computer Science and Engineering, Yuan Ze University, Taiwan.
Email: sallylin0121@saturn.yzu.edu.tw

2022 IEEE International Conference on Consumer Electronics - Taiwan | 978-1-6654-7050-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCE-TAIWAN55306.2022.9868995

***Abstract*—Recently, many factories have utilized AI to help Automatic Optical Inspection (AOI) machines accurately detect defects. They usually deploy AI models on the clouds and submit the data to the clouds for inference. However, transmission delay increases the response time of the AI model. If AI can differentiate defects on the local edge devices, the production efficiency can be significantly improved. In this paper, we propose a light-weight defect detection system that utilizes pruning techniques to compress the model and can accurately detect defects at a faster speed. Besides, we compare the performance of pruned and unpruned models on Kneron KL520 AI dongle and NVIDIA Jetson Nano to verify the superior ability of pruning to accelerate inference. The accuracy of the pruned model in the proposed system can reach 97.7% on Kneron KL520 AI dongle. The inference speed is 28.2 frames per second, 1.6 times faster than the unpruned model. Also, compared to NVIDIA Jetson Nano, the inference speed on Kneron KL520 AI dongle is two times faster. This result shows the better performance of Kneron KL520 AI dongle than NVIDIA Jetson Nano on inference. In summary, the proposed system can significantly improve the efficiency of production lines and avoid the information security risks brought by cloud computing.**

I. Introduction

Electronics manufacturing is one of Taiwan's mainstream industries. In recent years, component miniaturization has led to the development of AOI equipment with AI. Since AI needs powerful computation resources, AI models are usually deployed on the clouds. Data are sent to clouds for model training and inference. However, the transmission rates and response time may be unacceptable. If AI model can be deployed on the local edge devices, the production efficiency can be significantly improved.

In this paper, we propose a light-weight defect detection system to calculate data on the edge devices and then send a small number of results back to other devices. We first implement the "image preprocessing module" to perform size modification and normalization to reduce the computational burden, then unify the color channels of data from different production lines. In the "data augmentation module", we augment data to solve the uneven amount of data in each class. Then we design a "pruning module" to speed up the inference speed using the L2-norm criterion and Least Absolute Shrinkage and Selection Operator (LASSO) regularization. With this method, the model can fit into the edge devices. Besides, we verify the superior ability of pruning on inference speed, and compare the model's performance on Kneron KL520 AI dongle and NVIDIA Jetson Nano. Our experiment shows that the pruned model deployed on different edge devices is at least 1.6 times faster than the unpruned model in inference speed. Therefore, the system can accurately classify defects, greatly improve the efficiency of production lines and avoid the information security risks brought by cloud computing.

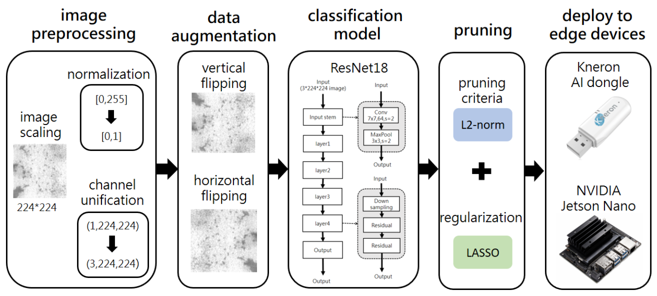

Fig. 1. System framework

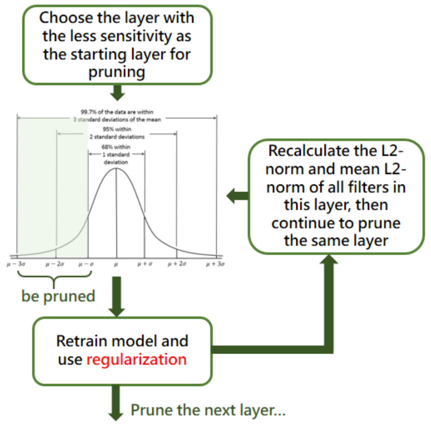
Choose the layer with the less sensitivity as the starting layer for pruning
be pruned
Recalculate the L2-norm and mean L2-norm of all filters in this layer, then continue to prune the same layer
Retrain model and use regularization
Prune the next layer...

Fig. 2. The concept of pruning process in the system

II. Research Methods

Our system framework is shown in Fig 1. The "image preprocessing module" is composed of three parts. The first part is image scaling, which modifies the size of the input image to 224*224. The second part is to convert the pixel values to the range of 0-1. The last step is to unify the RGB channels into (3, 224, 224) format. In the "data augmentation module," we utilize vertical flipping and horizontal flipping to augment data. In the training process, we adopt ResNet18 and

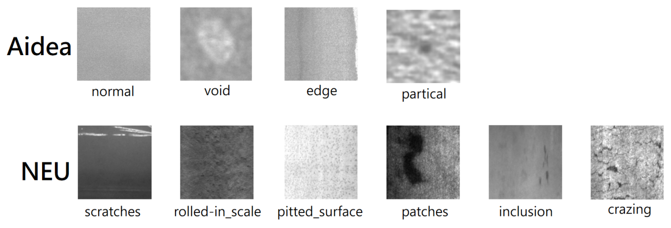

Fig. 3. Datasets from Aidea(upper) and NEU(bottom)

TABLE I
ORIGIN AND PRUNING BY RESNET18.

	Unpruned model	
Device	***Kneron KL520 AI dongle***	***NVIDIA Jetson Nano***
Accuracy	97.7%	98.2%
Inference speed	17.8	7.7

	Pruned model	
Device	***Kneron KL520 AI dongle***	***NVIDIA Jetson Nano***
Accuracy	97.7%	98.7%
Inference speed	28.2	13.8

add LASSO regularization [1] to increase the sparsity, which is the percentage of zero weight in filters, to facilitate pruning and avoid the risk of overfitting. In the "pruning module," we iteratively prune the filter in some convolution layers. The pruning process starts from the less sensitive layer and then pruning its neighboring layers progressively. In order to select redundant filters, we use the L2-norm [2] to measure the importance of each filter. We prune the filter whose L2-norm value is less than one standard deviation from the mean L2-norm value of the same layer and retrain with LASSO regularization for 10 epochs after pruning one time. Finally, we deploy the pruned and unpruned models to Kneron KL520 AI dongle and NVIDIA Jetson Nano for making a comparison.

III. EXPERIMENT

In our experiment, we deploy the pruned and unpruned models to Kneron KL520 AI dongle equipped with KL520 (NPU) and NVIDIA Jetson Nano equipped with 128-core Maxwell (GPU) for making a speed comparison. We use Aidea dataset [3] and NEU dataset [4] to evaluate the proposed system. Fig 3 shows four defect types of Aidea AOI data and six types of hot-rolled strip defects of NEU dataset. We use LASSO regularization to increase model sparsity and take L2-norm as a pruning criterion.

Table I shows the inference results of the original and pruned models deployed to Kneron KL520 AI dongle and NVIDIA Jetson Nano, respectively. Since the model must be quantized before deploying to Kneron KL520 AI dongle, the accuracy decreases slightly. Although the accuracy on KL520 is lower than using NVIDIA Jetson Nano, the inference speed is way faster. This result shows that the KL520 outperforms the 128-core Maxwell in accelerating neural networks. Table I also shows the pruned models deployed on different edge devices are at least 1.6 times faster than the unpruned models in inference speed. The accuracy can even be higher than the unpruned models by retraining. These results confirm the superior ability of pruning on the model's inference speed and accuracy improvement.

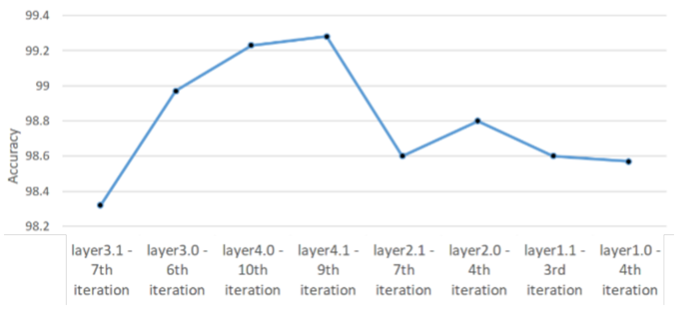

Fig. 4. Retraining accuracy from the last iteration of each layer

Furthermore, we observe some phenomena in the experiment described below. Fig 4 shows the accuracy after pruning and retraining at the last iteration of each convolution layer. We start pruning from the less sensitive layer and then prune its neighboring layers progressively. Initially, the accuracy after retraining shows an upward trend with increased pruned neurons. However, the retraining accuracy gradually decreases after multiple iterations. We consider the possible reason for this phenomenon: the remaining neurons are getting fewer and fewer after every pruning iteration, and the importance of the information contained in each neuron will gradually increase. Therefore, if we continue pruning, it may result in a decrease in the model's accuracy.

IV. CONCLUSION

In this paper, we proposed a light-weight defect detection system for edge computing, which compressed the classification model and improved the inference speed of the model. The accuracy of the pruned model was 97.7% and 98.7%, and inference speeds were 28.2 and 13.8 frames per second when the model was deployed on KL520 AI dongle and NVIDIA Jetson Nano, respectively. We found that the pruned model deployed on different edge devices was at least 1.6 times faster than the unpruned model in inference speed.

From experiments, the proposed system successfully reduced the complexity of the model and make accurate inferences on hardware devices with limited computing power. Therefore, our system can not only intensively improve the efficiency of defect detection in production lines but also keep the data safe on edge sides.

REFERENCES

[1] Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in neural information processing systems 28 (2015).*
[2] Li, Hao, et al. "Pruning filters for efficient convnets." *arXiv preprint arXiv:1608.08710 (2016).*
[3] Aidea AOI dataset : https://aidea-web.tw/topic/6354136b-3301-4306-aa5e-07fd07e1838a
[4] NEU surface detect database : http://faculty.neu.edu.cn/songkechen/zh-CN/zhym/263269/list/index.htm