Applying YOLOv10 Combined with Image Tiling for Defect Detection in CMOS Image Sensors

Abstract: To identify surface defects on CMOS image sensors using industrial cameras, this study addresses the challenges posed by high-resolution industrial camera images. To reduce the loss of image features caused by resizing, we utilized the state-of-the-art object detection algorithm, YOLOv10. We compared the commonly used image scaling method (which conserves training and inference resources) with our proposed method of combining YOLOv10 with image tiling. Both methods were trained and deployed on Jetson Orin NX 8GB for inference comparison. This improvement significantly enhanced the training results, with precision, recall, and mAP@0.5 increasing from 0.79, 0.68, and 0.75 to 0.87, 0.82, and 0.88, respectively. The inference results also showed an increase in mAP@0.5 from 0.75 to 0.88, demonstrating that the algorithm can accurately detect small-scale objects in high-resolution images.

Keywords: YOLOv10; Object Detection; Image Tiling; Small Object; Edge Computing

1. INTRODUCTION

In modern industries, particularly in semiconductor manufacturing, high-resolution cameras are commonly used for image capture, followed by defect detection using Automated Optical Inspection (AOI) [1] or Artificial Intelligence (AI) techniques. These industrial cameras typically boast high resolutions[2]. In traditional methods of object detection, to reduce computational load, it is often necessary to resize images before processing to enhance processing efficiency. While this approach can improve processing speed, it may also lead to the loss of fine details of smaller objects, thereby affecting the ultimate detection accuracy.

We identified defects on CMOS image sensors (CIS)[3] using high-resolution images captured by industrial cameras as input. We integrated the YOLOv10[4] model with our proposed image tiling[5] technique and compared the training and inference results on the same computer specifications and NVIDIA Jetson Orin NX 8GB.

Through comparisons with the original method of scaling images for prediction, we demonstrated the superiority of this approach in detecting small objects. This method not only improves the detection accuracy of small objects but also maintains performance for larger objects.

2. EXPERIMENTS

In this chapter, we will discuss the architecture, dataset, preprocessing methods, and hyperparameter tuning used in the training process, divided into the following subsections: Architecture, Preprocessing, Datasets, Model Training, and Inference.

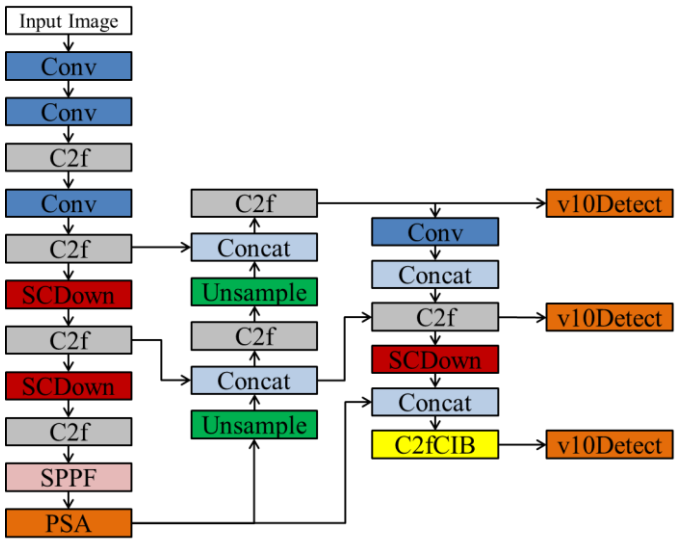


Fig. 1 YOLOv10 Architecture

2.1 Architecture

YOLOv10 is designed to accurately predict the classes and locations of objects in images with low latency. It is widely used in various practical applications, including autonomous driving[6], robotic navigation[7], and object tracking[8]. The architecture integrates several novel improvements, achieving outstanding detection accuracy while maintaining high speed and efficiency. The YOLOv10 architecture diagram is shown in Figure 1.

2.1.1 NMS-Free Training

During training, YOLO models[9] typically utilize TAL[10] to assign multiple positive samples to each instance. This one-to-many assignment approach generates rich supervisory signals, which is beneficial for optimization and achieving superior performance. However, it requires YOLO to rely on NMS post-processing, resulting in suboptimal inference efficiency during deployment. YOLOv10 addresses this by proposing the use of consistent dual assignment, eliminating the need for NMS[11], thereby reducing inference latency, As shown in Figure 2.

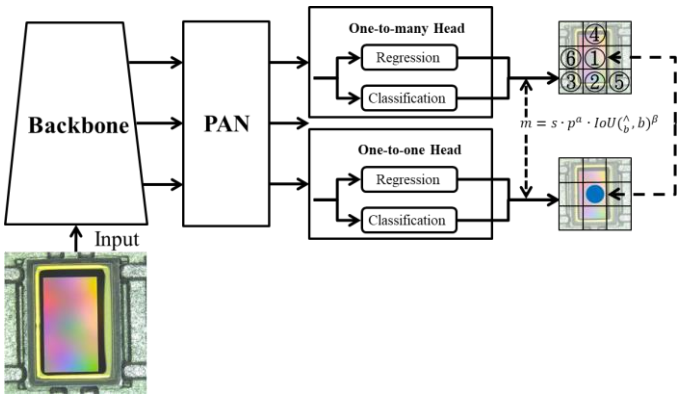


Fig. 2 Consistent dual assignments for NMS-free training

2.1.2 Holistic Model Design

The YOLO model architecture presents significant challenges in balancing efficiency-accuracy trade-offs[12]. Although previous YOLO models have undergone various design strategies, a comprehensive examination of the components within YOLO has been lacking. YOLOv10 thoroughly optimizes the efficiency and accuracy of different parts, featuring a lightweight classification head, spatial-channel decoupled downsampling, and order-guided block design.

2.1.3 Enhanced Model Capabilities

Integrates large-kernel convolutions and partial self-attention[13] modules to enhance performance without incurring significant computational overhead.

Therefore, this experiment will utilize YOLOv10 as the algorithm for detecting defects in CMOS Image Sensors.

2.2 Preprocessing

Due to the large size of images captured by industrial cameras and the need to identify areas located in the center, we first cropped the images outward from the center, creating new images with dimensions of 1,080 x 1,080, as shown in Figure 3, where the boxes indicate small defects on the sensor. Considering the inconsistent placement angles of the objects to be tested, we used Torchvision for data augmentation[14] to enhance the model's robustness.

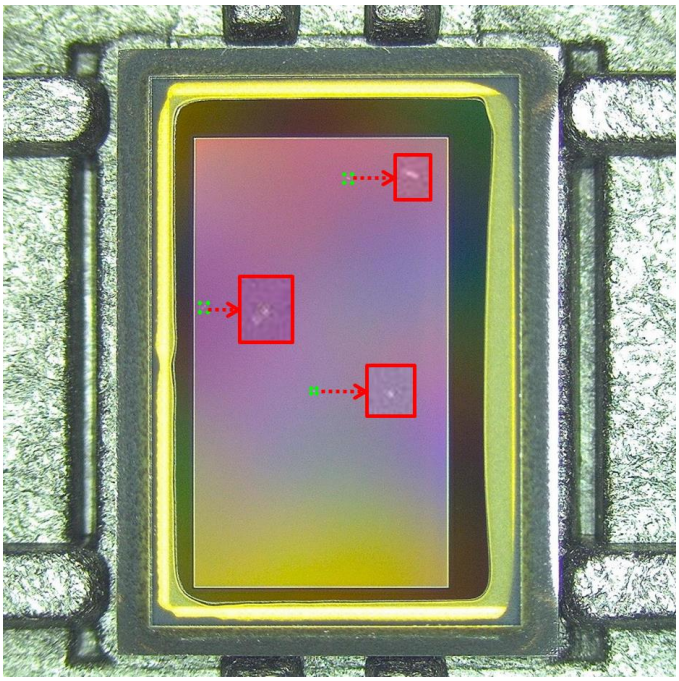


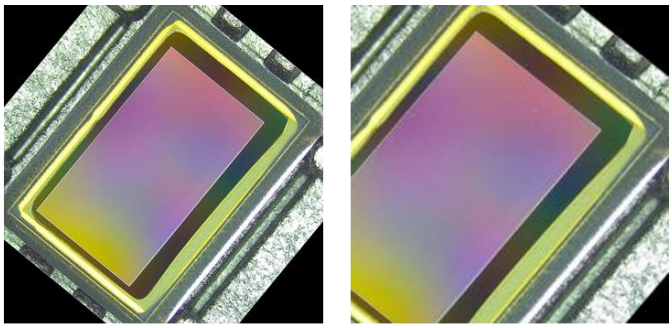
Fig. 3 Experiment A: Traditional Training Method

2.2.1 Random Rotation

Randomly rotates the input images at a given probability to accommodate defects of different shapes and orientations, as shown in Figure 4(a).

2.2.2 Random Resize Crop

At a given probability, random cropping will be applied to a randomly assigned size, followed by resizing to the specified dimensions, as shown in Figure 4(b).



(a) (b)

Fig.3 (a)Random Rotation (b)Random Resize Crop

2.3 Datasets

The dataset used in this experiment consists of CMOS images captured by an industrial camera with a resolution of 1,920×1,080 pixels, annotated for white spots. There are a total of 2,169 images in the dataset, with 1,518 images in the training set, 433 images in the validation set, and 218 images in the test set. Then, two datasets are created:

2.3.1 Experiment A: Image Scaling

Scale all input images to 640x640.

2.3.2 Experiment B: Image Tiling

Randomly select an annotated object and extend outwards from its center to create a 640×640 pixel image. If other annotated objects are not within the image, repeat this process using a different annotated object as the center and extend outwards. If the extended area exceeds the original image dimensions, fill the extra space with black pixels. As shown in Figure 5.

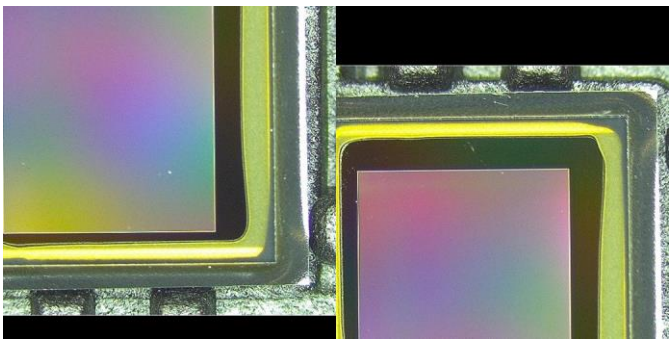


Fig. 5 enter-Based Cropping Method

2.4 Model Training

Considering the requirement for fast recognition speed while maintaining relatively high accuracy in practical development, we opted for the YOLOv10 Medium model. Both experimental groups were trained using the same hyperparameters, as shown in Table 1, to ensure comparable results. All models were trained on an NVIDIA GeForce RTX 4080 SUPER GPU with 10,240 CUDA cores.

Table 1 Training Hyperparameters Configuration

Hyperparameter	Value
Epochs	500
Batch Size	32
Workers	0.01
Learning Rate	640
Weight Decay	5×10^{-4}
Optimizer	SGD
Momentum	0.937

2.5 Inference

We will divide the inference part into two experiments for comparison:

2.5.1 Experiment A: Image Scaling

Resize the input image to the same size as the training images, which is 640×640.

2.5.2 Experiment B: Image Tiling

When dividing the input image into multiple tiles, formulas (1) and (2) can be used to determine the number of horizontal and vertical tiles.

$$tiles_x = [\frac{\mathcal{W}}{step}] \quad (1)$$

$$tiles_y = [\frac{\mathcal{H}}{step}] \quad (2)$$

Where $\mathcal{W}$ and $\mathcal{H}$ represent the width and height of the input image, respectively, and $step$ is the distance moved during each tiling step. This step distance is calculated as the tile size minus the overlap area. The overall inference process is shown in Figure 6.

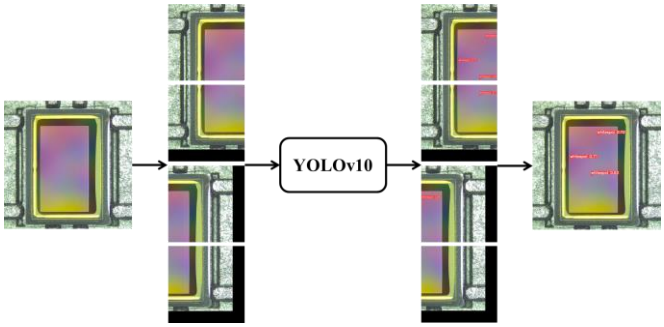


Fig. 6 Image Tiling Inference Process

3 RESULTS AND DISCUSSION

In this chapter, we discuss performance metrics, the size of the trained model, and the results of training and inference on the test set.

3.1 Performance Metrics

This section will introduce the evaluation metrics[15] and model size.

3.1.1 Mean Average Precision

Mean Average Precision (mAP) is a widely recognized performance metric for object detection models, as shown in formula (3). The Average Precision (AP) for each class "k" is determined by calculating the area under the precision-recall curve. mAP provides a comprehensive score by integrating recall, precision, and Intersection over Union (IoU), thereby offering an unbiased performance measurement.

$$\text{mAP} = \frac{1}{n}\sum_{k=1}^{k=n} \mathcal{AP}_k \tag{3}$$

The Average Precision (AP) for each class "k" is determined by calculating the area under the precision-recall curve. mAP provides a comprehensive score, integrating recall, precision, and Intersection over Union (IoU), thereby offering an unbiased performance measurement.

3.1.2 Size of the Trained Model

This study involves deploying the model on NVIDIA Jetson Orin NX 8GB. Due to the limitations in onboard memory and hardware storage capacity, larger models require more computation, resulting in lower efficiency. Therefore, it is crucial to keep the model size as small as possible. For this reason, we selected the YOLOv10 Nano model.

Table 2 presents the YOLOv10 Nano model specifications.

Table 2 YOLOv10 Nano Model Specifications

Model	YOLOv10-N
Test Size	640
#Params	2.3M
FLOPs	6.7G
AP^{val}	38.5%
Latency	1.84ms

3.2 Training Results

In this study, Experiments A and B were conducted using the same computer specifications for training. Here are the respective results for each experiment.

3.2.1 Experiment A: Image Scaling

Table 3 presents the results obtained through training using the Image Scaling method. The model achieved a precision of 0.79, a recall of 0.68, a mAP of 76% at an IoU of 0.5, and a mAP of 35% at an IoU of 0.95. Figure 7 shows the confusion matrix of the trained model.

Table 3 The training results of Experiment A

Model	YOLOv10-N
mAP@0.5	0.765
mAP@0.95	0.356

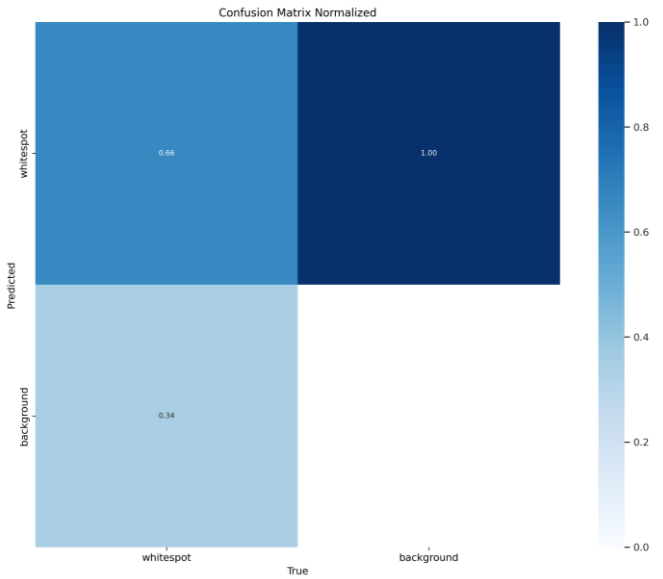


Fig. 7 The confusion matrix for Experiment A

3.2.2 Experiment B: Image Tiling

Table 4 presents the results obtained from training using the Image Tiling method. The model achieved a precision of 0.87, a recall of 0.82, a mAP of 89% at an IoU of 0.5, and a mAP of 45% at an IoU of 0.95. Figure 8 shows the confusion matrix for the trained model.

Table 4 The training results of Experiment B

Model	YOLOv10-N
mAP@0.5	0.894
mAP@0.95	0.457

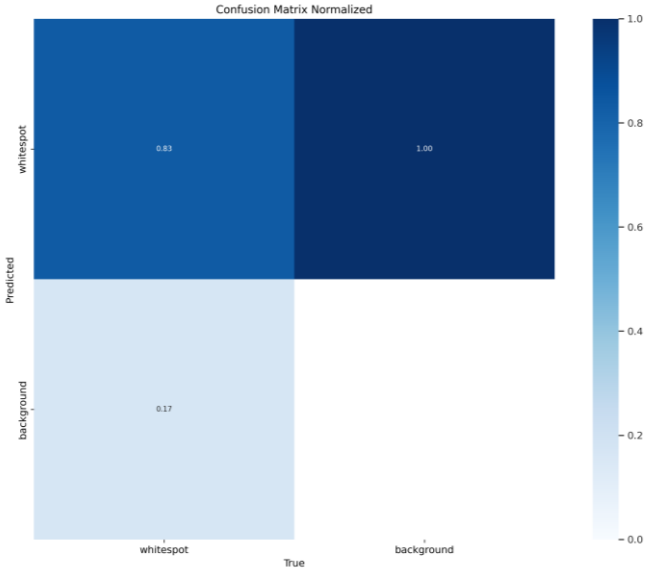


Fig. 8 The confusion matrix for Experiment B

3.3 Training Results

In this study, the inference for both Experiment A and Experiment B was deployed on Jetson Orin NX 8GB. Here are the results for each experiment.

3.3.1 Experiment A: Image Scaling

By performing inference validation on the test set images, the input images were resized to 640x640 for inference, and the results are shown in Figure 9. The inference results show that the mAP is 75% at an IoU of 0.5 and 34% at an IoU of 0.95, with an average processing time per image of 5.7 x 10^{-3} seconds. Table 5 presents the inference data results.

Fig. 9 The inference result of Experiment A

Table 5 The inference data results of Experiment A

Model	YOLOv10-N
mAP@0.5	0.756
mAP@0.95	0.346
Processing Time	5.7 x 10^{-3} (sec)

3.3.2 Experiment B: Image Tiling

By performing inference validation on the test set images, the original input images were cropped using image tiling techniques to ensure each input image is 640x640, and then inference was carried out as shown in Figure 10. The final merged result is shown in Figure 11. The inference results show that the mAP is 88% at an IoU of 0.5 and 42% at an IoU of 0.95, with an average processing time per image of 1.25 x 10-1 seconds. Table 6 presents the inference data results.

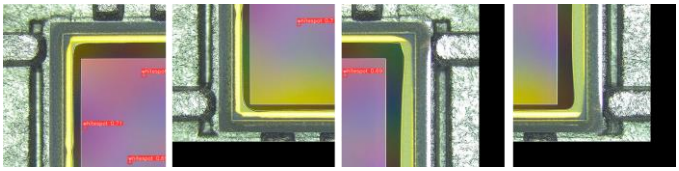


Fig. 10 Individual inference for each tile

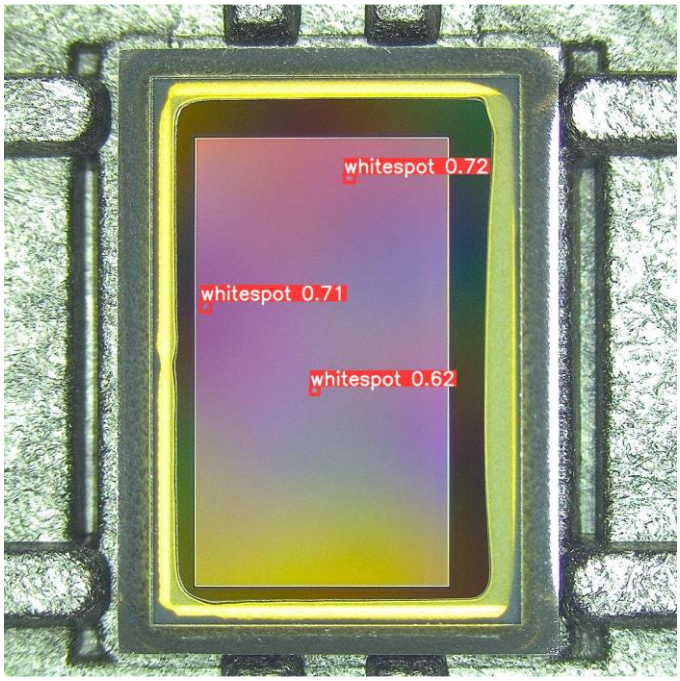


Fig. 11 The Inference Result of Experiment B

Table 6 The inference data results of Experiment B

Model	YOLOv10-N
mAP@0.5	0.887
mAP@0.95	0.424
Processing Time	1.25×10^{-1} (sec)

4. CONCLUSION

In conclusion, this study uses YOLOv10 for object detection and selects the Nano model to meet the requirements of edge devices. We compared the traditional Image Scaling method with our proposed Image Tiling method for training and inference. The results demonstrate that although our YOLOv10 model with Image Tiling has an average increase of 0.12 seconds per image during inference, the mAP for training improved by 28.4%, and the mAP for inference on the test set increased by 22.5%.

These results indicate that the Image Tiling method can significantly enhance the model's detection accuracy. Therefore, despite the increase in inference time, Image Tiling remains a powerful approach for improving object detection accuracy and is worth considering for applications with industrial cameras or other high-resolution scenarios.

REFERENCES

1. Kim, S., Kim, W., Noh, Y. K., & Park, F. C. (2017, May). Transfer learning for automated optical inspection. In 2017 international joint conference on neural networks (IJCNN) (pp. 2517-2524). IEEE.
2. Sudoh, Y. (2019). Optical system for industrial camera that achieves both close minimum object distance and high resolution. Electronic Imaging, 31, 1-5.
3. El Gamal, A., & Eltoukhy, H. (2005). CMOS image sensors. IEEE Circuits and Devices Magazine, 21(3), 6-20.
4. Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). Yolov10: Real-time end-to-end object detection. arXiv preprint arXiv:2405.14458.
5. Ozge Unel, F., Ozkalayci, B. O., & Cigla, C. (2019). The power of tiling for small object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops (pp. 0-0).
6. Bogdoll, D., Nitsche, M., & Zöllner, J. M. (2022). Anomaly detection in autonomous driving: A survey. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 4488-4499).
7. Dos Reis, D. H., Welfer, D., De Souza Leite Cuadros, M. A., & Gamarra, D. F. T. (2019). Mobile robot navigation using an object recognition software with RGBD images and the YOLO algorithm. Applied Artificial Intelligence, 33(14), 1290-1305.
8. Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X., & Wei, Y. (2022, October). Motr: End-to-end multiple-object tracking with transformer. In European Conference on Computer Vision (pp. 659-675). Cham: Springer Nature Switzerland.
9. Xu, S., Wang, X., Lv, W., Chang, Q., Cui, C., Deng, K., ... & Lai, B. (2022). PP-YOLOE: An evolved version of YOLO. arXiv preprint

arXiv:2203.16250.

10. Feng, C., Zhong, Y., Gao, Y., Scott, M. R., & Huang, W. (2021, October). Tood: Task-aligned one-stage object detection. In 2021 IEEE/CVF International Conference on Computer Vision (ICCV) (pp. 3490-3499). IEEE Computer Society.
11. Neubeck, A., & Van Gool, L. (2006, August). Efficient non-maximum suppression. In 18th international conference on pattern recognition (ICPR'06) (Vol. 3, pp. 850-855). IEEE.
12. Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., ... & Chu, X. (2023). Yolov6 v3. 0: A full-scale reloading. arXiv preprint arXiv:2301.05586.
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
14. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48.
15. Padilla, R., Netto, S. L., & Da Silva, E. A. (2020, July). A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP) (pp. 237-242). IEEE.