A Comparative Study of Rule-based Image Processing Techniques and Deep Learning Algorithms in Automated Optical Inspection

Michael DC. Trinidad
Diego Jose L. Cabalza
Ian Carlo F. Ecal

Carmona Automation and Mechatronics, Test Operations
onsemi, Golden Mile Business Park, SEZ Governor's Drive, Maduya, Carmona, Cavite, Philippines
Michael.Trinidad@onsemi.com, DiegoJose.Cabalza@onsemi.com, Ian.Ecal@onsemi.com

ABSTRACT

In the competitive field of semiconductor manufacturing, ensuring the timely delivery of high-quality products to customers is paramount. Leveraging recent advancements in imaging technology and artificial intelligence, Automated Optical Inspection (AOI) systems are now increasingly adopted to replace slow, manual inspection methods.

A typical AOI setup can cost around $20,000 USD which includes computers, cameras, lighting systems and computer vision software applications. In pursuit of cost-effectiveness, onsemi Carmona made a significant investment in the development of an in-house computer vision system. This system plays a vital role in transforming traditional manual inspection into a sophisticated automated optical inspection process by deploying both rule-based image processing techniques and deep learning models.

This paper delves into the early challenges and recent improvements in developing computer vision systems for defect detection. It also highlights the drawbacks of conventional image processing techniques compared to the deep learning approach.

The paper used mold package defect detection as a case study demonstrating the superiority of deep learning models. Notably, YOLOv8 (You Only Look Once version 8) emerged as a powerful solution with 99.28% accuracy. This outperformed conventional rule-based image processing techniques that were used initially.

1.0 INTRODUCTION

The mold assembly process has long been associated with package defects, which can arise from various factors such as human-induced causes and machine issues. Common defects include voids, package chipping, scratches, foreign material, and cracks. In this context, mold manual vision inspection becomes an ideal starting point for automation.

The in-house computer vision system, developed by the team, utilizes an Ultra-HD 4K camera integrated with locally fabricated handling equipment. However, its performance on mold samples falls short of the company's standards. Figure 1 shows an MSA trial result achieving only 86.3% accuracy with false alarm rate of 11.74% and miss rate of 36.36%. These suboptimal performance metrics pose risks both to customers in terms of quality and to the manufacturer due to potential yield loss. Maintaining high accuracy in computer vision systems is crucial, especially when dealing with quality control and manufacturing processes. Balancing false alarms and misses is a delicate task and improvements in the software can lead to better outcomes for both customers and the company.

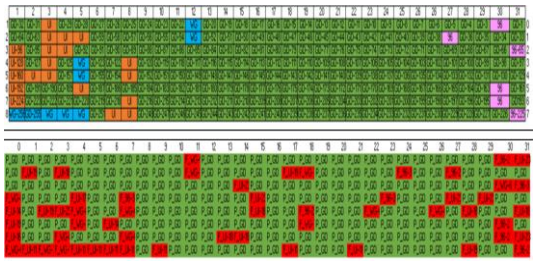

Figure 1. Measurement Systems Analysis (MSA) Reject Map vs Rule-based Output Map. MSA Reject Map is at the top while the Rule-Based Output Map is at the bottom. The performance metrics of the Rule-based method, employing set rules with image processing techniques, had an accuracy of 86.3%, a miss rate of 36.36%, and a false alarm rate of 11.74%.

1.1 Rule-based Image Processing Techniques and Limitations

Rule-based image processing techniques are among the earliest methods used in computer vision systems. This traditional approach aims to classify objects within an image for various purposes. Essentially, developers implement rules and algorithms to process images. These rules often involve logical conditions expressed through code. The acceptance or rejection of specific features in an image is typically based on measures such as thresholds, object counts, standard deviation and area in terms of pixels.

1.1.1 Thresholding

In the initial stages of recipe creation, colored images captured by the camera are converted into grayscale. For binary, pixel intensities below the threshold value become 0

(black), and those above it become 255 (white). The opposite is true for binary inverse thresholding. Figure 2 shows the binary and binary thresholding processes with a defined threshold value.

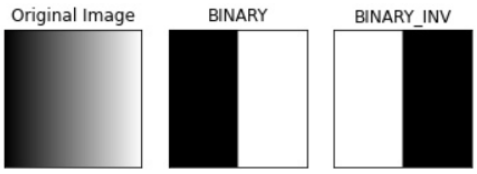

Figure 2. Binary and Binary Inverse Thresholding. The first image displays a grayscale image with varying pixel intensities, which can be converted into numeric values. The second image represents binary thresholding. In this process, intensity values below a specified threshold become black, while those above it become white. The third image, on the right, is the opposite of binary thresholding, referred to as binary inverse thresholding.

Figure 3 shows an example of using binary thresholding to isolate the void defect from an image as part of the recipe creation process. The light-colored void within the image becomes a separate object, while the remaining portion of the package serves as the background.

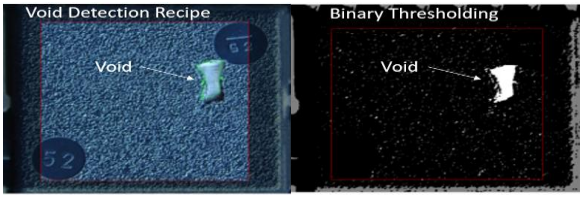

Figure 3. Recipe Test Creation for Void Detection using a sample image. Voids refer to the presence of porous-like empty cavities on the molded package surface. These are visually manifested as pinholes, pits, spherical holes, or crater-like features.

Figure 4 shows the void detection test result of a sample unit. While the original image is converted to binary thresholding, the isolated void defect is not easily seen compared to the test recipe. This causes its pixel values to fall beneath the recipe's threshold limit. Consequently, the defect is undetected due to the recipe's inability to accurately segment the image.

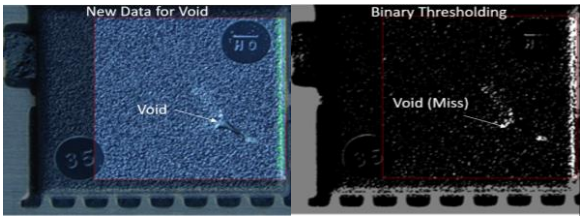

Figure 4. Void Defect Detection Test Result in Sample Unit. A void defect seen as a pit hole appears to have the same color as the rest of the captured unit, showing a missed detection despite correct defect classification.

To address the issue of defects having varying threshold values for segmentation purposes, the recipe is enhanced with additional tests to accommodate different scenarios or defect variations. Typically, variations in color necessitate multiple tests, which in turn add extra cycle time to the inspection process.

1.1.2 Fixed Region of Interest versus Variable Indexing

One of first requirements in building an automated optical inspection recipe is to set the ROI (Region of Interest). ROI defines the borders on which the operations of the image processing application take place. Figure 5 shows a test recipe with set ROI for detection of voids within the area. Areas outside of the ROI are not considered during the analysis of the image.

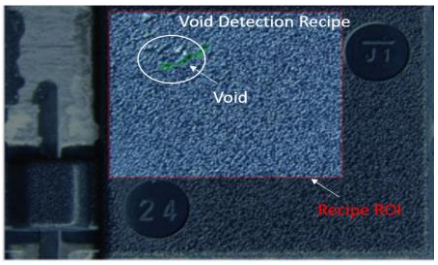

Figure 5. Region of Interest (ROI) for a Void Defect Detection Recipe. The figure shows a recipe for detecting voids using a rectangular border, serving as the Region of Interest (ROI). The rectangular box area is where the analysis is expected to occur.

Figure 6 shows a test result of a sample unit with the positional shift of the ROI. This shift is attributed to minute variations in mechanical shifting of the handler or the lead frame itself, even those as small as a few microns. This can result in a failure in detection since void falls outside the ROI.

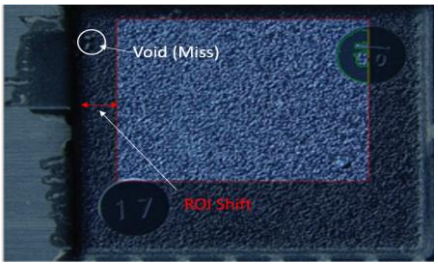

Figure 6. Region of Interest (ROI) for a Void Defect Detection Recipe. The figure illustrates a positional shift of the ROI. The void, situated in the upper left section, falls outside the ROI, leading to a failure in detection.

1.1.2 Filtering

In creating recipes for automated optical inspection, it is important to consider the varied sizes of defects. Some defects falling under the same category can come in varied sizes, color, and shapes. Voids for example can look like a large crater and some can look like a small pin hole. It is advisable when using a rule-based image processing technique to create multiple inspection tests to account for these variations. One technique that can detect the smallest

defect size is filter coverage. Figure 7 shows the application of filter coverage, reducing noise by filtering out supposed defects too small to meet the actual rejection criteria.

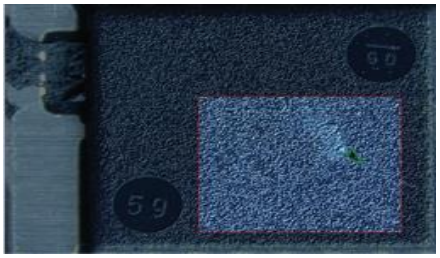

Figure 7. Filter coverage for a Void Defect Detection Test Recipe. The figure shows a small void defect, where a specified filter coverage limit value enables the recognition of a clump of black pixels as voids within this recipe.

Figure 8 shows a test result of a sample unit with the application of filter coverage. Numerous clumps of dark pixels are displayed which fall within the filter coverage limit value. This results in these clumps not filtered as noise. This circumstance can trigger a false alarm, suggesting the existence of any void where there is none.

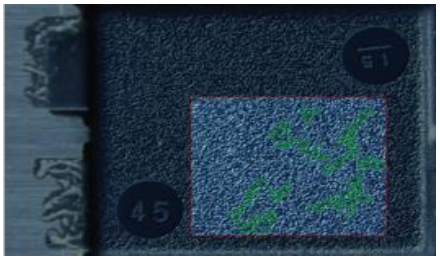

Figure 8. Filter coverage for a Void Defect Detection Test Recipe. The figure displays a new image featuring numerous clumps of dark pixels. This circumstance can trigger a false alarm, suggesting the existence of any void where there is none.

2.0 REVIEW OF RELATED WORK OR LITERATURE

Saberironaghi et al. [1] reviewed the use of deep learning techniques, such as R-CNN, ResNet50, and ShuffleNetV2 convolutional neural networks, for defect detection on industrial products. Examples of defect detection include corrosion detection and metal defect detection. It also acknowledges the common challenges in defect detection such as unbalanced sample identification, limited sample size of defects, and real-time processing.

Dehaerne et al. [2] optimized the use of YOLOv7 for Semiconductor Defect Detection. A dataset of SEM (scanning electron microscopy) images was used for model training as they produce high-resolution images for inspecting defects at the nanometer scale. Defect instances of the image dataset include line collapse, gap, p-gap, bridge, or microbridge defects. The base YOLOv7 model achieved a mean average precision (mAP) of 79%. This was then improved to 86.8% by adjusting certain hyperparameters during model training.

Xiao et al. [3] proposed a Deep Learning-based defect detection algorithm for Printed Circuit Boards (PCBs) based on CDI-YOLO. The network structure of CDI-YOLO is seen as an improvement of YOLOv7, a previous version of YOLOv8. The results of their methodology produced a mean average precision (mAP) of 98.7% on a sample PCB defect dataset with a detection speed of 128 frames per second.

Trinidad et al. [4] conducted a previous case study where a Deep Learning model was trained by another Convolutional Neural Network called ResNet50. The model is used to identify Personal Protective Equipment (PPE) worn by the operator before accessing bake ovens. The case study achieved an impressive accuracy of 99.6% and successfully differentiated between proper and improper wearing of PPE.

3.0 METHODOLOGY

3.1 Data Collection

Deep learning for object detection requires large quantities of data for the model to perform well. Figure 9 shows the in-house developed handling equipment, with two bar lights attached as side lightings to illuminate each unit surface during image capturing. Side lightings are effective in highlighting surface contours and particularly effective in detecting defects such as cracks, scratches, and pits. With the equipment, the team initially collected images from 15 strips of TSSOP devices containing 256 units for each strip. The handler is equipped with an Ultra HD 4K camera that can capture an image with an original size of 1920 X 1080 pixels. An auto-crop feature was employed to reduce the image size to 958 X 1077 pixels, effectively removing regions that are not included in the inspection process.

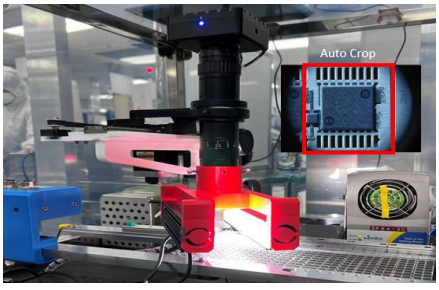

Figure 9. AOI Machine Camera Setup with Integrated Lighting for Image Capture. The figure demonstrates the application of an Ultra HD 4K camera, equipped with two bar lights for side package illumination. It also provides a preview of the image captured from a single unit under each lighting condition.

3.2 Annotation

In object detection tasks, annotating images with descriptive labels is crucial for training models. LabelMe, an open-source graphical annotation tool, is widely used for this purpose. Defects within the images can be annotated using various forms such as lines, points, and rectangles. These defects are then labeled according to their respective categories. Figure 10 shows an example of annotating a void defect from an image.

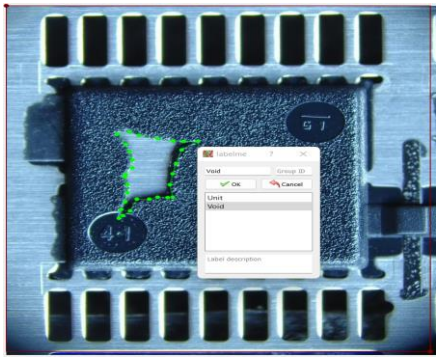

Figure 10. Sample Annotation of Defects in Image. The figure shows an annotation task carried out on a unit with a void defect using LabelMe.

3.3 Conversion from JSON to YOLO

After using LabelMe for object detection annotation task, the initial file format of the labels is in JSON (JavaScript Object Notation). To train a YOLOv8 model, the JSON file needs to be converted to a text format. Figure 11 shows a sample label file of a converted JSON file. This conversion can be achieved by running a "labelmetoYolo" script.

Figure 11. Sample Label File for an Image in JSON Format. The figure presents a sample label file of an image, structured in JSON format. The file's contents show the rejects identified within the image.

3.4 Model Training

YOLOv8, as it is called on the Ultralytics website, represents the latest iteration of the renowned real-time object detection and image segmentation model. Leveraging state-of-the-art advancements in deep learning and computer vision, YOLOv8 is designed for high-performance tasks. Figure 12 shows an in-house application for YOLOv8 also developed to streamline model training tasks. Initially, the data is divided into two sets: training and validation. To simplify the process, the team focused on binary classification, specifically identifying units as either 'good' or having 'void defects.' The data split serves the purpose of training the model and evaluating its performance. After training for 64 epochs, the model retains the best weights for optimal results.

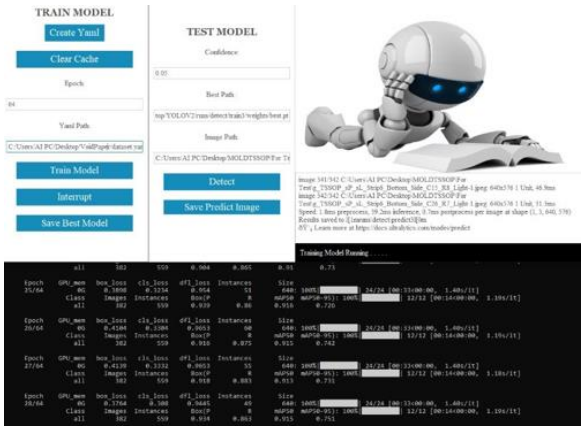

Figure 12. YOLOv8 Model Training using a Custom UI Application. The figure shows the model training process carried out in YOLOv8 using a custom-built user interface specifically designed for training deep learning models.

When model training is finished, its output will include a generated best model along with the statistical results of the process. Figure 13 shows a selection of images also displayed in the output trained with YOLOv8.

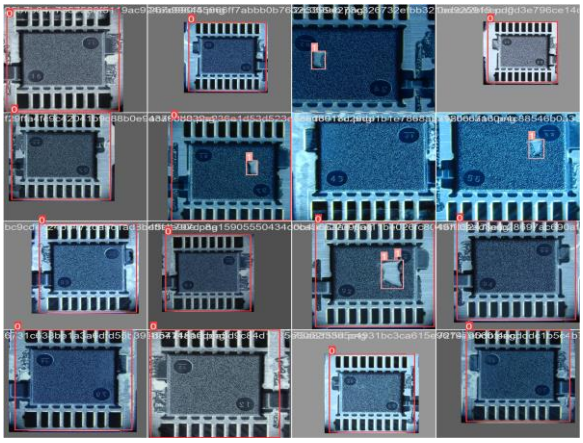

Figure 13. YOLOv8 Model Training with an Image Dataset. The figure displays a selection of images that have been trained using the YOLOv8 model.

4.0 RESULTS AND DISCUSSION

4.1 Accuracy of Model Training

The overall loss components in YOLOv8 are the box_loss (Box Loss), cls_loss (Classification Loss) and dfl_loss (Dual Focal Loss). Figure 14 shows these three loss components in a downward trend. Box loss is the measured error between the bounding box and the ground truth bounding box coordinates. Classification Loss is the classification accuracy where the object class is correctly predicted. Lastly, the Dual Focal Loss addresses the problem of class imbalance. It adjusts the loss function to give more importance to less frequent classes.

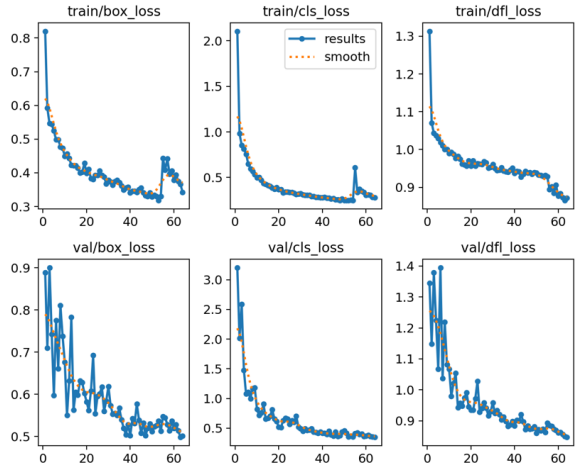

Figure 14. Loss Metrics Over Training Epochs for Both Training and Validation Sets. The figure shows the progression of box_loss, cls_loss, and dfl_loss for both the training and validation sets over increasing number of epochs. A clear downward trend is observed, indicating that the model's accuracy is improving during the training process.

The optimal model is chosen based on the highest mAP (Mean Average Precision) score. Figure 15 shows the mAP score reflecting how effectively the model localizes and classifies objects.

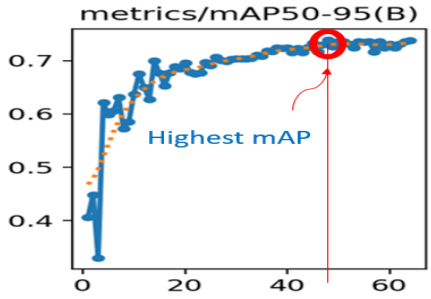

Figure 15. Peak Model Performance Accuracy Score. The figure shows the peak mAP value of 0.74 achieved at epoch 48. This peak value signifies the model's optimal performance effectiveness in terms of both localization and classification accuracy.

4.2 Prediction Result

To assess the performance of rule-based image processing versus YOLOv8, the team collected a sample of 276 images. These images contained randomly selected units, both 'good' and those with 'void defects.' Figure 16 shows the result comparison of Rule-based image processing and YOLOv8. Rule-based image processing scored a moderate 77.17% with 14.22% miss rate and around 60.78% in false alarm. It predicted all the units with defects. However, most of it was not actually captured but pointed to a different part of the package. In contrast, YOLOv8 exhibited near zero false alarms and zero miss rate. It correctly predicted the defects and its location in the package.

Method	Accuracy	Miss	False Alarm
Rule-based Image Processing	77.17%	14.22%	60.78%
YOLOv8	99.28%	0%	3.9%

Figure 16. Rule-based method vs. YOLOv8 results. The table shows that the accuracy of YOLOv8 greatly surpassed that of rule-based image processing performance.

Figure 17 shows sample images of units classified as rejects using Rule-based image processing. However, many of these detections are inaccurately positioned, not aligning with the actual locations of the defects.

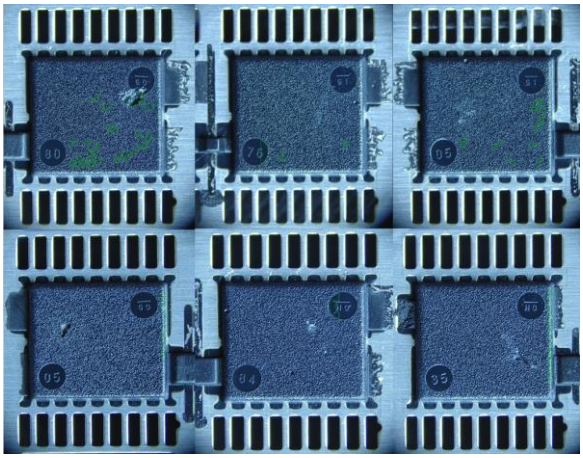

Figure 17. Sample images of Units classified as Voids with Low Positional Accuracy using Rule-Based image processing. The figure shows that the rule-based method classified many units as rejects but with inaccurate positioning.

Figure 18 shows sample images of units misclassified as misses using Rule-based image processing. The void defects seen in the units were not captured by the recipe in the rule-based method, contributing to the miss rate.

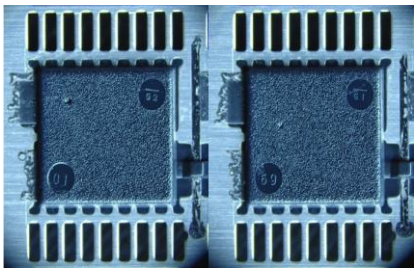

Figure 18. Sample images of Misclassified Units using Rule-Based method. The figure shows examples of misclassified units that exhibit actual void defects but were not captured using the rule-based image processing method (miss-rate).

Figure 19 shows sample images of units also misclassified as false alarms using Rule-based image processing. The units were declared as having void defects despite not exhibiting such defects.

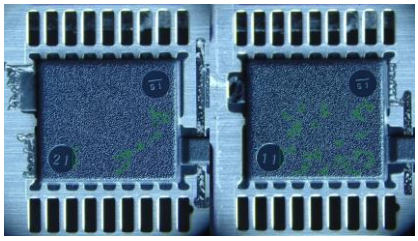

Figure 19. Sample images of Misclassified Units as False Alarm using Rule-Based method. The figure shows examples of misclassified units that were declared as failures using rule-based image processing, despite not exhibiting any actual void defect (false-alarms).

Figure 20 shows sample images of units correctly classified using a YOLOv8 model. The defects were successfully captured with their precise locations in each unit.

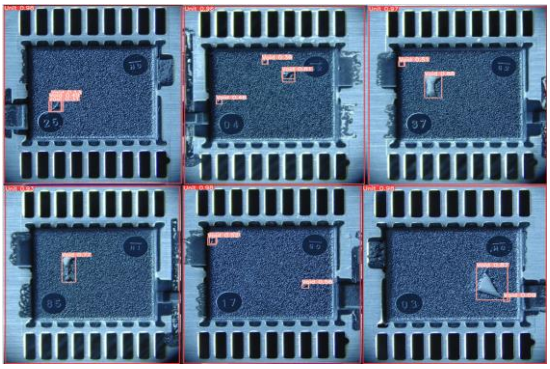

Figure 20. Sample images of accurately classified Units using YOLOv8. The figure demonstrates that the YOLOv8 model accurately classified void defects in their correct locations. (high accuracy).

Figure 21 shows the two misclassified units as false alarms using a YOLOv8 model. A highly minimal false alarm rate is still provided by YOLOv8, ensuring a higher accuracy than the rule-based method.

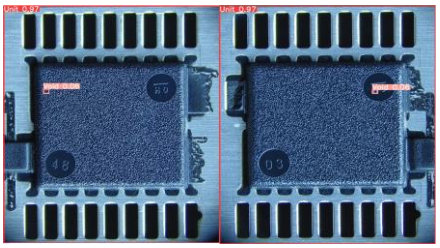

Figure 21. Sample images of Misclassified Units as False Alarm using YOLOv8. The figure shows only 2 misclassified units out of 276 without an actual void defect using YOLOv8 (minimal false-alarm rate).

Figure 22 shows more images gathered with various defects to validate whether YOLOv8 can effectively detect more mold defects. These images were annotated and trained using LabelMe and YOLOv8 respectively. Subsequently, a separate set of test images was analyzed using the trained YOLOv8 model. The results were promising, with the custom-built YOLOv8 model precisely identifying the reject category corresponding to the observed defects. The defects include incomplete fill, scratches, voids, chipping, ejector pin damage and contamination.

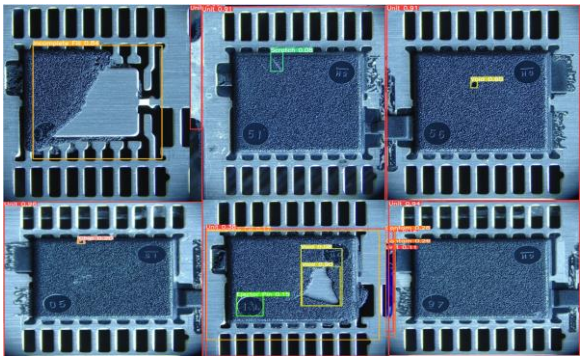

Figure 22. More images of accurately classified Units using YOLOv8. The shows a variety of defects captured by YOLOv8. This was achieved by training on annotated images with multiple defect categories. The effectiveness of YOLOv8 in identifying multiple defect categories is clearly demonstrated in the figure.

5.0 CONCLUSION

As discussed in Section 1.0, an in-house computer vision system was developed to transform manual inspection into an automated optical inspection process as an initiative towards cost-effectiveness. At first, the system utilized rule-based image processing techniques for mold defect detection. However, it only achieved an accuracy of 86.3% with a false alarm rate of 11.74% and a miss rate of 36.36% during

Measurement Systems Analysis (MSA) Testing. These results fall below company standards.

Section 3 discussed that by employing YOLOv8, it can deliver better results when provided with a sufficient dataset for model training. It also managed to more accurately locate each defect per unit during testing and provide more flexibility in object detection. Using the same image dataset to test the accuracy of the two methods, YOLOv8 provided a 99.28% accuracy with near-zero false alarm rate and zero miss rate. Meanwhile, the rule-based method only provided a 77.17% accuracy with a false alarm rate of 60.78% and a miss rate of 14.22%.

In conclusion, YOLOv8 deep learning models proved to be a superior alternative to rule-based image processing.

6.0 RECOMMENDATIONS

While deep-learning-based methodologies can be used for image classification, it is also resource-intensive. This will require high-performance hardware like CPUs/GPUs for training deep learning models. Large quantities of data such as images are also required for a more optimal model performance.

In manufacturing settings, where manual inspection processes remain prevalent, YOLOv8 offers a cost-effective alternative to traditional AOI (Automated Optical Inspection) systems. Being an open-source model, YOLOv8 provides flexibility and can serve as a valuable secondary inspection tool. However, inspected images can still undergo further analysis offline to enhance defect detection and quality control.

More importantly, collaboration with process engineering is crucial to know better the rejection criteria of all defects for each product to provide an even more accurate result.

7.0 ACKNOWLEDGMENT

The team extends its gratitude to Vic Delos Reyes for the unwavering support in our ongoing efforts to enhance the in-house developed computer vision project. Additionally, we express our appreciation to Peter Awayan and Ana Eugenio for providing support in writing this paper. Remarkably, we have achieved three consecutive years of publishing papers for onsemi.

8.0 REFERENCES

1. Saberironaghi A, Ren J, El-Gindy M. Defect Detection Methods for Industrial Products Using Deep Learning Techniques: A Review. Algorithms. 2023; 16(2):95. https://doi.org/10.3390/a16020095

2. Dehaerne, E., Dey, B., Halder, S. & De Gendt, S. Optimizing YOLOv7 for Semiconductor Defect Detection. (2023). 10.48550/arXiv.2302.09565.

3. Xiao, G., Hou, S. & Zhou, H. PCB defect detection algorithm based on CDI-YOLO. Sci Rep 14, 7351 (2024). https://doi.org/10.1038/s41598-024-57491-3

4. Trinidad, M., Cabalza, D., & Fajardo, R. One-shot PPE Detection for Bake Ovens using ResNet50 Convolutional Neural Network. 2023

9.0 ABOUT THE AUTHORS

Michael DC. Trinidad, an Electronics and Engineering graduate from the Technological University of the Philippines - Taguig, is a seasoned professional in the Semiconductor Manufacturing Industry. Since joining onsemi in 2006, he has held roles such as Mixed Signal Equipment Engineer, contributing to his expertise in semiconductor testing. He played a key role in a significant big data analytics project in 2018 and is currently deeply involved in vision systems and machine learning implementation.

Diego Jose L. Cabalza completed his Bachelor of Science degree in Electronics and Communications Engineering from the University of the Philippines – Diliman in 2022. He currently serves as a manufacturing engineer at onsemi Carmona, where he focuses on projects related to Data Analytics and Machine Learning.

Ian Carlo F. Ecal is a skilled Manufacturing Equipment Engineer with a strong background in Mechanical Design and Tool and Die Engineering Technology. He holds a license as a Registered Mechanical Engineer and completed his education at the Technological University of the Philippines - Manila from 2012 to 2019. His expertise lies in designing and optimizing mechanical systems, ensuring efficient and reliable manufacturing processes.