15

Discovering and Classifying Digital and Wooden Industries Products' Defects at the Edge by a Yolo/ResNet-based Approach and Beyond

Robin Faro[1], Alessandro Strano[1], and Francesco Cancelliere[2]

[1]Deepsensing SRL, Italy
[2]University of Catania, Italy

Abstract

The paper aims to present an AI-based Automated Optical Inspection (AOI) software for both digital and wooden industries developed within the EdgeAI project. Current approaches rely on centralized solutions, where the computation is performed inside the inspection machine itself. Instead, we present algorithms that work at the edge to give rise to competitive solutions to existing ones. In particular, we experiment with two different tasks of defect identification: detecting the defect position within a wooden panel by using YOLO, in which a 96% accuracy is reached. Secondly, concerning the digital industry, we perform a two-step classification between defective and non-defective microchips and then between four possible defect classes in their surface exploiting a ResNet network and obtaining a 97% accuracy. We also exploit explainability tools to understand which parts of the images caused the model's decision. After developing the AI models we port them to two less power-consuming edge devices, Nvidia Orin Nano, and Nvidia Orin AGX, observing unchanged performance.

Keywords: automated optical inspection, edge computing, key performance indicators, deep learning, computer vision, PCBA defect detection, wooden product defect detection, NVIDIA Jetson ORIN.

15.1 Introduction

The paper aims to present an AI-based AOI software developed within the EdgeAI project for defect detection of the PCBAs used in the digital industry to be implemented at the edge with the main expected outcome of being a viable and cost-effective solution for the inspection of many industrial products, not only digital boards. In particular, these algorithms are at the core of the AOI solution shown in Figure 15.1 where visual testing is done at the edge and learning in the cloud. This solution, as illustrated in [1], can reduce purchase and power consumption costs without increasing latency. Indeed, in this architecture, learning is done on the cloud but several tests suitable for highlighting groups of defects can be performed in parallel by competing boards thus decreasing latency.

A solution similar to the one adopted in the EdgeAI project has been proposed by Advantech where the NVIDIA board is replaced by the MIC-72 but without discussing the algorithms used in practice for defect detection. On the contrary, several algorithms have been proposed in the literature to manage the abovementioned problems using GPUs. From the literature, we found that these are mainly optimized versions of the DL-powered YOLO algorithm [2]. The mAP (mean Average Precision) of the latest proposed algorithms goes beyond 99%, i.e. 99.17% in [3], 99.5% in [4] , and 99.71%

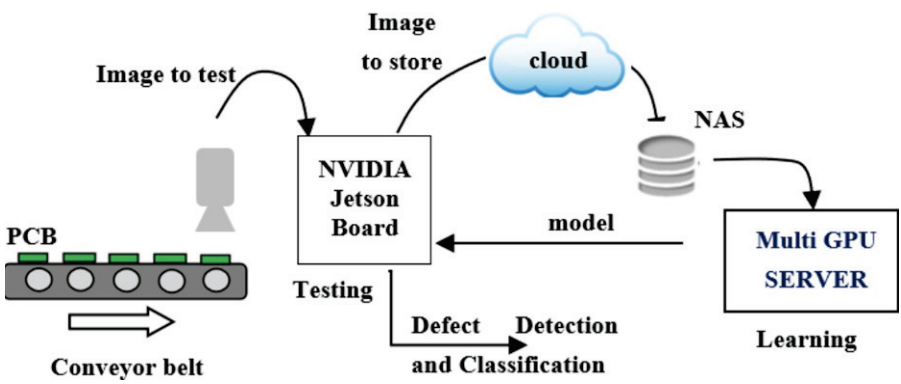


Figure 15.1 An AOI solution consisting of an edge board for testing and a GPU server for learning.

in [5]. However, this comparison is only indicative since the mentioned precision values were not achieved using the same data set. Also, it is not shown the performance degradation passing from a solution implemented on GPUs to real testing done using edge devices.

The feasibility of implementing YOLO-based algorithms for edge tests has been recently shown in literature, for example, in the YOLO implementation on NVIDIA Jetson TX2 illustrated in [6] which is characterized by satisfactory precision performance, that is, mAP0.5 = 98%. This is a relevant starting point for our implementations aiming at solving two open problems:

- To what extent very accurate algorithms can be implemented for edge tests using more performant NVIDIA boards, such as JETSON ORIN, for the quality control of PCBAs to be used in applications where the constraint of near-zero defects is required
- How to implement such algorithms on less expensive boards such as STM32

We decided to conduct experiments on wood and digital industries products which are also cases of study in the context of EdgeAI project. We start by performing wood defect detection on a publicly available dataset. Secondly, considering the absence of a significant dataset on defective PCBAs (which are the main focus of the project), we conducted preliminary experiments on advanced packaging microchips. Finally, we export the models into edge devices comparing the performance of the ported models in terms of latency, power consumption, and accuracy.

15.2 Related Works

Wood Defect Detection

The idea of an AI-based detection of wood superficial imperfections was introduced several years ago. For example, [7] tried to catch the presence of a defect in Pinus lumber. The dataset was small and composed of 400 training images and 100 test images, two different learning approaches were used: Neural Networks and Support Vector Machines. Even if these algorithms are pretty simple, interesting results were achieved, with a 97% accuracy. In [8] this problem was faced by using a cascade of Adaboost classifiers. First, he tried to detect the presence of blue stain, then of decay, and finally of cracks. The experiment was conducted on a limited collection of about 100 images, which presented 300 examples of defects and the best result obtained was a 12% error rate. Subsequently, [9] decided to exploit a VGG16 model for this

task, obtaining really high performance on a 6 common defects dataset. They also performed data augmentation on the original 1200 available pictures and trained a Mix-FCN model that achieved a 91% pixel accuracy.

A direct application of the above-mentioned YOLO architecture was instead done by [10], who aimed to distinguish among 4 different typologies of defects, obtaining an 88% mAP. Furthermore, [11] proposed a modified version of YOLOv7 for wood floor small defect detection which reaches a 94% mAP. Similarly, [12] showed that a backbone-modified version of YOLOv7 reached an mAP of 81% on our same public dataset.

In summary, wood defect detection is a big challenge considering the various shapes, positions, and possible combinations of the existing industrial defects, which could worsen the performance of a model trained to catch only a part of them. Hence our goal is to exploit state-of-the-art error detection methods in order to increase the amount of defects that our model is able to spot, without penalizing reliability and above all inference time, which is a crucial parameter in industrial world applications.

Chip Surface Defect Detection

The detection of surface defects in chips is crucial for ensuring high-quality production in the semiconductor industry. Historically, methods for detecting these defects relied on image processing techniques. For example,

[13] used a combination of grayscale transformation, mathematical morphology, and pattern recognition to detect defects on printed circuit boards, achieving high detection speed. Similarly, [14] employed filtering methods with Support Vector Machines to classify defects. These methods are fast but often struggle with generalization across different defect types due to their reliance on manually set parameters.

Deep learning has become increasingly popular in recent years and to address this limitation its advancement has introduced more robust methods for defect detection. Most modern techniques are divided into three categories: classification, segmentation, and object detection. For instance, [15] utilized an improved Spatial Pyramid Pooling Network (SPPNet) for defect classification, while [16] applied a 3D convolutional neural network (CNN) to classify defects on wafers. Object detection networks, such as Faster R-CNN and YOLO, are increasingly favored due to their speed and accuracy. For example, [17] used Faster R-CNN to detect multiple types of defects in steel and concrete structures, while [18] enhanced YOLOv3 with a Group Pyramid Pooling module for rapid detection of PCB surface defects.

Moreover, other deep learning-based techniques such as the SSD (Single Shot MultiBox Detector) and its variants have been explored for defect detection. [19] proposed an SSD-based method that utilized shallow features to detect smaller defects, although it faced challenges in detecting finer details. Later improvements such as DSSD [20] (Deconvolutional Single Shot Detector) achieved better detection accuracy by incorporating deconvolution layers to add more contextual information, although at the cost of increased processing time.

To tackle the problem of small object detection, [21] proposed using YOLOv3 with multi-scale feature maps, but it showed limited effectiveness for small defects due to insufficient deep feature extraction. An improvement came with the introduction of YOLOv4, which balanced speed and accuracy better than previous models. However, small object detection remained a challenge. In response, [22] developed SO-YOLO, a modified version of YOLOv4, aimed at detecting small-scale chip defects by improving shallow feature fusion. This approach achieved superior performance with an 86% mAP, surpassing YOLOv4 and even YOLOv5.

The continuous evolution of deep learning methods has also seen the integration of attention mechanisms to improve defect detection. For example, [23] proposed a weakly supervised detection framework to predict the location and probability of defects using a small dataset, achieving 99.5% accuracy.

These approaches underscore the ongoing challenges in defect detection, particularly for small objects, and highlight the importance of balancing accuracy with inference speed in real-world industrial environments.

Deployment of AI Models at the Edge

Deploying AI models at the edge has become an increasingly popular solution due to its ability to bring computation closer to data sources, reducing latency and reliance on centralized infrastructure.

In edge computing, AI models are deployed on local, resource-constrained devices, such as edge boards or embedded systems like Nvidia Orin Nano or Orin AGX. These devices are capable of executing complex deep learning models directly at the point of data collection, enabling faster response times and reducing the load on cloud services. This approach is especially beneficial in industrial settings where real-time defect detection is crucial, such as quality control in manufacturing, where any delay in detecting defects can result in significant costs.

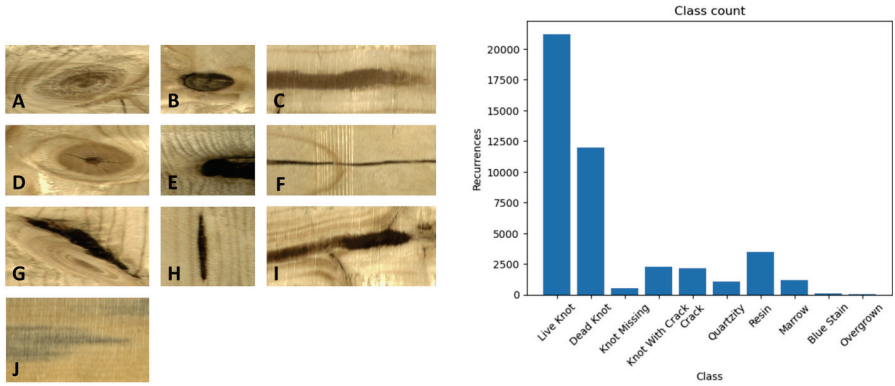
One of the key challenges in deploying AI models at the edge is the limitation of computational power and energy resources. Models need to be optimized to balance inference accuracy with speed while keeping resource consumption in check. Techniques such as model quantization, pruning, and using more efficient architectures like YOLO for object detection and ResNet for classification have proven effective in ensuring that edge devices maintain high performance without compromising on precision. The deployment of these models allows for effective parallel testing of multiple samples, leading to decreased latency compared to centralized processing, where communication delays and network reliability can introduce bottlenecks. Moreover, edge deployment supports data privacy and security by processing data locally, which is often a requirement in industrial settings.

15.3 Spotting Defects in Wood Industry Products

15.3.1 Defect Detection Dataset

First, we try to identify the exact position of a defect within an image of a wooden panel. To do so, we exploit a publically available dataset containing a total of 20275 images: 1992 images of sawn timbers without any defects and 18283 timber images with one or more surface defects. On average, there are 2.2 defects per image, while only 6.7% of images contain more than three defects. The highest occurrence of defects is 16 defects per image. The dataset includes the following wood defects:

1. **Live Knot:** A portion of a tree branch incorporated into the trunk, appearing as a circular or oval area of darker wood. Live knots are solid and firmly attached but may cause irregular grain patterns.
2. **Dead Knot:** Similar to a live knot, but from a non-living tree. Dead knots result from a branch that died and fell off, leaving a void filled with resin or bark, often differing in color or texture from the surrounding wood.
3. **Knot Missing:** The absence of a knot where one would typically be expected, leads to a more uniform wood appearance.
4. **Knot With Crack:** A knot that contains a crack or split within it.
5. **Crack:** A separation or break in the fiber structure of the wood.
6. **Quartzity:** The presence of quartz or silica deposits in the wood, appearing as small, translucent or whitish mineral inclusions.
7. **Resin:** The presence of sticky or resinous substances, occurring naturally or due to injury or stress, often appearing as pockets or streaks of amber-colored substance.



(a) Examples of wood defects (b) Distribution of defects in the dataset

Figure 15.2 Visual representation of wood defects and their distribution.

8. **Marrow:** Soft, spongy tissue found in the central portion of the tree trunk, lighter in color and softer than the surrounding wood.
9. **Blue Stain:** A bluish discoloration caused by fungal or bacterial growth, leaving pigments that produce a blue or grayish tint in the wood.
10. **Overgrown:** Abnormal growth of wood fibers, resulting in irregular or distorted patterns, often due to hormonal imbalances or stress on the tree.

Some examples of the defects are shown in Figure 15.2 a while their distribution can be observed in Figure 15.2 b. It's important to underline the considerable unbalance between the classes, where the first two are dominant with respect to the other eight. Considering that our main concern is to spot defective elements rather than understanding their nature and that this class imbalance, together with the innate similarity between defects, makes it hard to distinguish among all of them, we decide to combine all of them under a general 'Defect' class.

15.3.2 Experiments and Results

The object detection model we decided to exploit is YOLO[24]. This model, whose name stands for You Only Look Once, was revolutionary in the field of object detection since it was able (starting from its first release) to output both ROIs (Regions of Interests) and their classification after just one forward pass of the input image through the network. In our case, we finetuned one of the last available releases of YOLO, which is YOLOv8, in its medium version

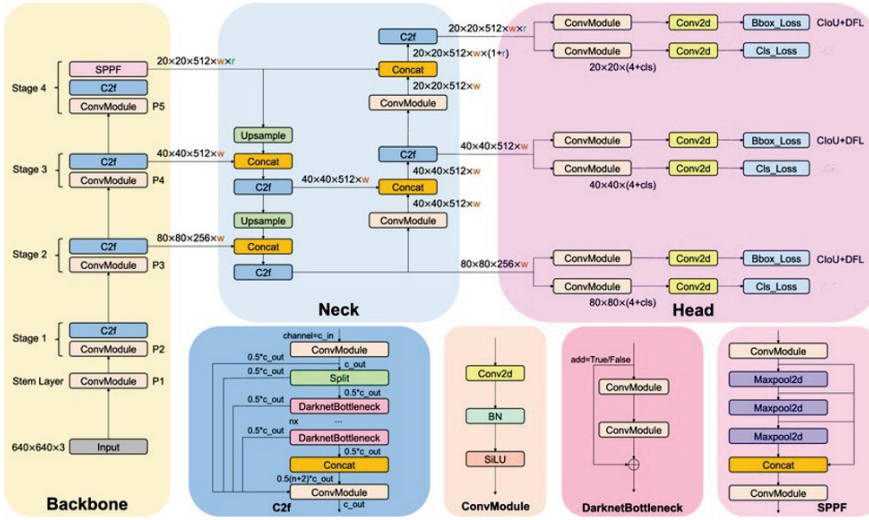


Figure 15.3 YOLOv8 architecture.

(around 26 million parameters). In general YOLO architecture (fully shown in Figure 15.3)is made up of three main blocks:

- Backbone → This is a CNN whose main role is feature extraction. It is composed of several stages which progressively reduce the spatial resolution while increasing the number of channels. Each of them relies on the so-called C2f block, which is in turn based on the Darknet Bottleneck, made up of 2 convolution blocks and a skip connection.
- Neck → This block aggregates information from different stages in order to improve the capacity of the model to recognize objects of different sizes. It is also based on the C2f block.
- Head → This component processes the aggregated features coming from the neck passing them through convolution blocks responsible for the final predictions of both bounding box and classes

We trained the model for 100 epochs and in terms of results, a 95% precision and a 94% recall are obtained, while the mAP_{50} reached a 96% value. Some of the predictions on the validation set are shown in Figure 15.4 while the progress of the abovementioned metrics during the training, as well as the loss functions on train and validation, is shown in Figure 15.5

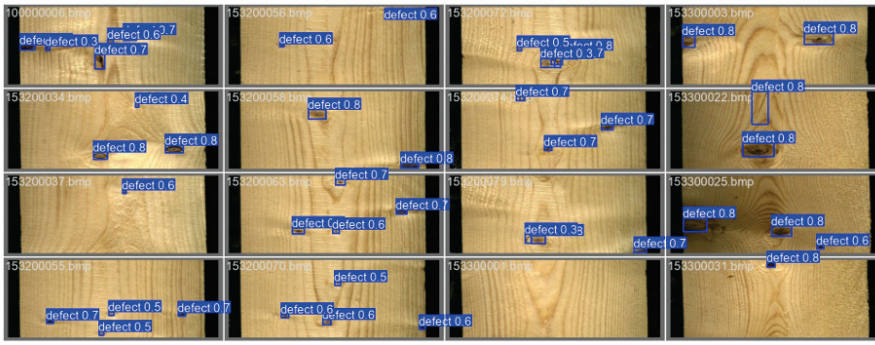
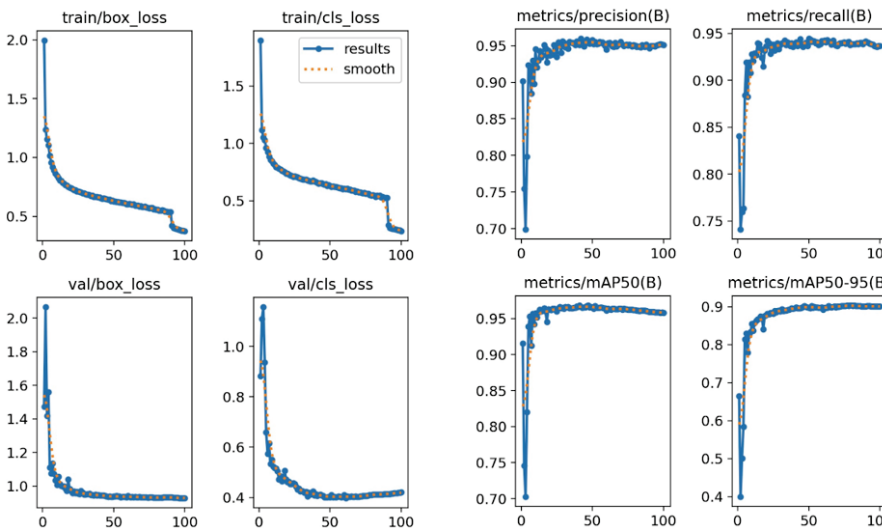


Figure 15.4 Example of prediction on validation set.



(a) Losses on train and validation sets (b) Precision, Recall and Mean Average Precison

Figure 15.5 Training metrics of YOLOv8 model.

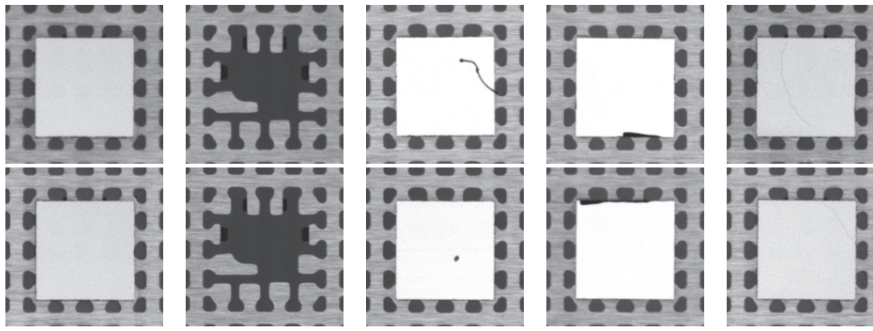
15.4 Spotting Defects in Digital Industry Products

In this section, we focus on the identification and classification of chip surface defects. Given the challenges associated with acquiring a comprehensive dataset, we employed the one introduced by Wang et al. [25], which, despite its modest size, provides valuable insights into the defect detection process.

15.4.1 Defect Detection and Classification Dataset

The selected dataset includes a total of 2763 images of chip surfaces, where 2000 images do not contain any defects, and 763 images present one among four possible defects. We implemented a two-step classification approach, aimed to streamline the industrial process and enhance the efficiency of quality control. The initial phase involves a binary classifier designed to quickly and accurately filter out defect-free chips. This step serves as a critical screening process, ensuring that only chips identified as non-defective continue through the subsequent stages of the industrial process. By doing so, we effectively reduce the computational load and focus further inspection efforts only on potentially problematic chips. This approach is particularly valuable in high-throughput manufacturing environments, where minimizing delays and optimizing resource allocation are crucial. For the chips flagged as defective in the first step, a more granular analysis is then conducted in the second step. This phase involves classifying the specific type of defect present on the chip surface. The detailed classification not only aids in determining the exact nature of the defect but also provides essential insights that can be used for root cause analysis. In particular, the dataset contains the following defect classes:

- NO_DIE (6b) represents a unique defect scenario where the actual error lies in the absence of the soldered chip: the chip is missing on the substrate. The absence of a chip can be attributed to manufacturing faults, such as misalignment during the chip placement process or incorrect soldering. These errors can occur due to equipment malfunction, human error, or process inconsistencies.
- DIE_INK (6c) includes chips with internal ink stains: these defects can arise from ink deposition errors or contamination during the manufacturing process. Detecting and classifying these internal defects is essential for ensuring the integrity and reliability of the chip's functionality.
- DIE_BROKEN (6d) involves chips with visible breaks or fractures along their edges. These breaks can occur during the manufacturing process or as a result of external factors. Detecting and accurately localizing these broken edges is crucial for quality control and identifying potential manufacturing issues.



(a) DEFECT_FREE (b) NO_DIE (c) DIE_INK (d) DIE_BROKEN (e) DIE_CRACK

Figure 15.6 Defect-free chip and four common chip surface defects

Table 15.1 Distribution of the dataset

DEFECT_FREE	NO_DIE	DIE_INK	DIE_BROKEN	DIE_CRACK
2000	100	135	42	486

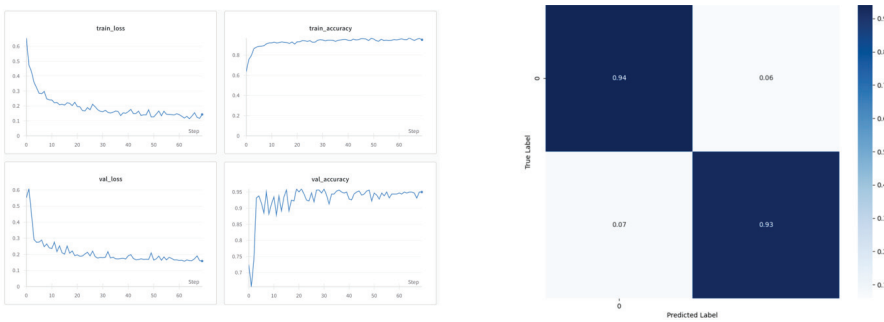
- DIE_CRACK (6e) represents chips with cracks occurring internally within the chip structure. These cracks can stem from stress during fabrication, handling, or environmental factors. Detecting and characterizing these cracks aids in identifying structural weaknesses and preventing potential chip failures.

The distribution of the different classes in the dataset is the following (1):

15.4.2 Experiments and Results

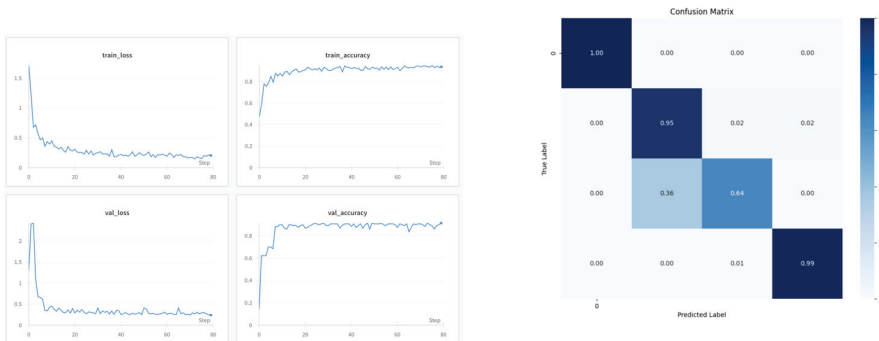
The model we used to perform classification is ResNet [26], in particular the version 18 layers deep (namely ResNet-18). It introduced the innovative concept of “skip connection", which connects the activations of a given layer to further layers by skipping some intermediate layer, thus alleviating the issue of vanishing gradient. The binary classifier was trained for 70 epochs and obtained a 94.5% accuracy and 94% recall, precision, and F1 score. The progress of accuracy and loss function on train and validation is shown in Figure 15.7 a

As regards the second phase classifier, it was trained for 80 epochs obtaining 97% accuracy, 87% recall, and 89% F1 score, as shown in Figure 15.8 a



(a) Loss and accuracy on train and validation sets (b) Confusion Matrix: 0 Defect Free - 1 Defective

Figure 15.7 Metrics' trends during training and test result.



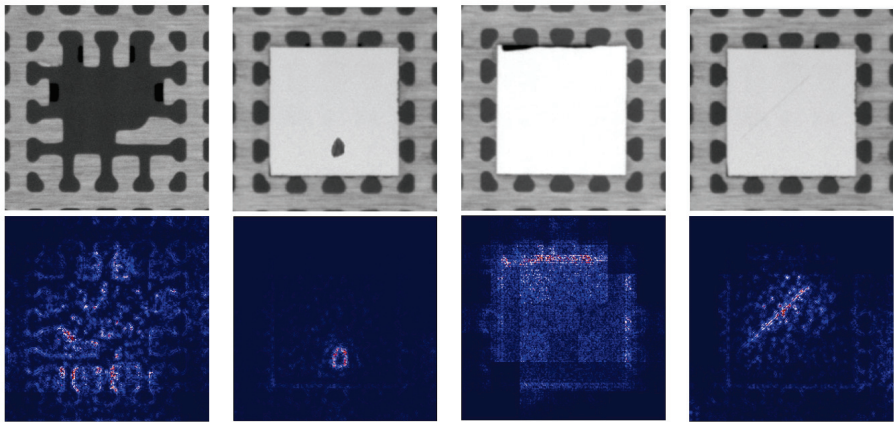
(a) Loss and accuracy on train and validation sets (b) Confusion Matrix: 0 NO_DIE - 1 DIE_INK - 2 DIE_BROKEN - 3 DIE_CRACK

Figure 15.8 Metrics' trends during training and test result

15.4.3 XAI Analysis: insigths into ResNet-18 using Grad-CAM

Explainable AI methods, often referred as XAI, are essential for enhancing the transparency and interpretability of deep learning models by providing insights into their decision-making processes. Grad-CAM (Gradient-weighted Class Activation Mapping) [27] visualizes the regions of an input image that most influence the model's predictions by highlighting the gradients flowing into the convolutional layers, thus allowing us to identify the specific areas of the chip surface on which the model focuses. Below, we provide examples of the Grad-CAM results for each defect class:

- NO_DIE: The activation maps effectively highlight the missing chip region, providing a clear indication of the defect (Figure 15.9 a).



(a) NO_DIE (b) DIE_INK (c) DIE_BROKEN (d) DIE_CRACK

Figure 15.9 Grad-CAM activation maps for different chip surface defect classes.

- DIE_INK: The maps emphasize the areas surrounding ink stains, accurately identifying the relevant regions for defect classification (Figure 15.9 b).
- DIE_BROKEN: It's the least represented class and in fact the maps show some uncertainty in pinpointing the exact broken regions, reflecting challenges observed in the confusion matrix and indicating potential areas for model improvement (Figure 15.9 c).
- DIE_CRACK: The activation maps clearly outline the crack, demonstrating the model's proficiency in localizing and detecting this defect (Figure 15.9 d).

15.5 Porting of the Models on Edge Devices

Porting AI models to edge devices is a critical step in realizing efficient, real-time solutions, especially in industrial settings. The edge devices used in this study include Nvidia Orin Nano and Nvidia Orin AGX, which are specifically chosen for their balance between computational power and energy efficiency. The process of porting models, such as YOLO for defect detection and ResNet for classification, involved converting the models to ONNX (Open Neural Network Exchange) format to ensure they could run efficiently on these resource-constrained devices while maintaining high accuracy. ONNX provides a unified format that allows models to be optimized for different hardware platforms, making them more portable and reducing dependency

on specific frameworks. By converting to ONNX, we were able to leverage hardware acceleration features available on our tested edge devices, in which we observed improved inference speed and reduced latency.

The deployment tests demonstrated that both devices could perform real-time inference, with the Orin AGX showing a higher throughput due to its superior GPU capabilities. However, the Orin Nano, being more cost-effective and energy-efficient, also provided satisfactory performance for applications where slightly lower throughput was acceptable. The deployment experiments showed that the models could achieve near-identical accuracy compared to their performance in a centralized server environment, proving the viability of using edge devices for industrial defect detection.

In addition, an important aspect of edge deployment is ensuring low latency and robustness under varying conditions. The edge devices managed to maintain high accuracy with inference times well within the acceptable range for real-time operations, demonstrating their suitability for on-site, autonomous quality inspection. By processing data locally, the system also ensures data privacy, which is crucial in industries dealing with sensitive information, such as PCB industries that deal with pre-production boards.

Overall, porting the models to edge devices proved successful, providing a viable solution for decentralized, real-time defect detection in industrial applications. The performance metrics, as shown in Tables 15.2 and 15.3, highlight the trade-off between inference speed and power consumption for each device. The Orin AGX, with its higher power consumption, offers better inference times, while the Orin Nano provides a more energy-efficient alternative suitable for less demanding applications. The use of powerful yet efficient edge devices like the Orin Nano and Orin AGX resulted in a system capable of high performance under the constraints typical of edge computing environments.

Table 15.2 Performance of Nvidia Orin Nano for model deployment

Model	Inference Time (ms)	Power Consumption (W)
Chip Defect/No Defect	25.1	11.2
Chip Defect Classification	64.5	13.4
Wood Defect Classification	72.3	14.2

Table 15.3 Performance of Nvidia Orin AGX for model deployment

Model	Inference Time (ms)	Power Consumption (W)
Chip Defect/No Defect	22.0	15.6
Chip Defect Classification	44.4	21.5
Wood Defect Classification	50.4	22.5

15.6 Conclusions and Future Works

In this paper, we presented an AI-based Automated Optical Inspection solution designed for both the digital and wood industries, focusing on defect detection at the edge. Through a series of experiments, we explored the application of YOLOv8 for wood defect detection and ResNet for chip surface defect classification. The results demonstrated that our proposed approach achieved high accuracy, precision, and recall, proving the effectiveness of deep learning models in industrial defect detection tasks. We also presented an explainability analysis of the ResNet prediction, which can help in understanding the region of the image that led to the model's decision.

Moreover, by porting the models to edge devices such as the Nvidia Orin Nano and Orin AGX, we successfully validated the feasibility of running these complex models in a resource-constrained environment without compromising performance, enabling real-time defect detection. We showed that the edge deployment maintains unvaried the accuracy of the models while reducing latency and power consumption, which are critical aspects in the industrial setup.

Future works will focus on several aspects to enhance our solution, for example:

- The next step involves integrating our solution into a real-time industrial setup. This will involve testing the models in environments where objects move on conveyor belts, and high-speed cameras capture images of products for defect detection. Such real-world trials will help assess the models' ability to handle real-time constraints, such as varying lighting conditions, motion blur, and differences in defect types and positions. By doing so, we aim to further optimize the system for seamless operation in an actual production line.
- While our models achieved high accuracy, future efforts will focus on enhancing the quality of training data. This involves not only increasing the quantity of data but also ensuring that it represents a wide range of defects across different types of products and environments. High-quality, diverse data is critical for improving the generalization capabilities of the models, reducing false positives and negatives, and adapting to unseen defect types. In particular, augmenting the dataset with hard-to-detect defects and edge cases will be crucial for improving the system's robustness.
- Facing directly the PCB issue by generating a dataset suitable for detecting defects in these boards, either real or synthetic. Successively,

testing both the one-stage and the two-stage solution presented in the paper. In particular, in the second case, we imagine a first model able to detect each PCB internal component (i.e. resistors, capacitors...) and then several “expert" classifiers that will be able to decide whether a component is correctly mounted or not.

- We aim to explore multi-task learning approaches where the models can detect and classify defects from multiple industries simultaneously, improving the efficiency and flexibility of the AOI system.

Acknowledgements

This work was supported by Chips Joint Undertaking EdgeAI project. The project EdgeAI “Edge AI Technologies for Optimised Performance Embedded Processing” is supported by the Chips Joint Undertaking and its members including top-up funding by Austria, Belgium, France, Greece, Italy, Latvia, Netherlands, and Norway under grant agreement No. 101097300.

References

[1] AAEON. *AI@Edge: AI Vision in Automated Optical Inspection*. Available at: http://www.aaeon.com/ru/ai/boxer-6841m-aoi-app-story.2023. http://www.aaeon.com/ru/ai/boxer-6841m-aoi-app-story.

[2] Juan Terven, Diana-Margarita Córdova-Esparza **and** Julio-Alejandro Romero-González. “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS”. **in** *Machine Learning and Knowledge Extraction*: 5.4 (**november** 2023), 1680–1716. issn: 2504-4990. 10.3390/make5040083. http://dx.doi.org/10.3390/make5040083.

[3] JiaYou Lim, JunYi Lim, Vishnu Monn Baskaran **and** Xin Wang. “A deep context learning based PCB defect detection model with anomalous trend alarming system”. **in** *Results in Engineering*: 17 (2023), **page** 100968. issn: 2590-1230. https://doi.org/10.1016/j.rineng.2023.100968. https://www.sciencedirect.com/science/article/pii/S2590123023000956.

[4] Yinchao Du, Jiangpeng Chen, Han Zhou, Xiaoling Yang, Zhongqi Wang, Jie Zhang, Yuechun Shi, Xiangfei Chen **and** Xuezhe Zheng. “An automated optical inspection (AOI) platform for three-dimensional (3D) defects detection on glass micro-optical components (GMOC)”.

in *Optics Communications*: 545, 129736 (**october** 2023), **page** 129736. 10.1016/j.optcom.2023.129736.

[5] Hongjin Zhu, Lina Xing, Honghui Fan **and** Tao Wu. *New PCB Defect Identification and Classification Method Combining MobileNet Algorithm and Improved YOLOv4 Model*. **april** 2022. 10.21203/rs.3.rs-1544671/v1.

[6] Shaojun Song, Junfeng Jing, Yanqing Huang **and** Mingyang Shi. "EfficientDet for fabric defect detection based on edge computing". **in** *Journal of Engineered Fibers and Fabrics*: 16 (**april** 2021), **page** 155892502110083. 10.1177/15589250211008346.

[7] P. Cavalin, L. S. Oliveira, A. L. Koerich **and** A. S. Britto. "Wood Defect Detection using Grayscale Images and an Optimized Feature Set". **in** *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*: 2006, **pages** 3408–3412. 10.1109/IECON.2006.347618.

[8] Selman Jabo. "Machine vision for wood defect detection and classification". **in** *M.S Thesis, Chalmers University of Technology*: (2011).

[9] Ting He, Ying Liu, Chengyi Xu, Xiaolin Zhou, Zhongkang Hu **and** Jianan Fan. "A Fully Convolutional Neural Network for Wood Defect Location and Identification". **in** *IEEE Access*: 7 (2019), **pages** 123453–123462. 10.1109/ACCESS.2019.2937461.

[10] Wei-Han Lim, Mohammad Babrdel Bonab **and** Kein Huat Chua. "An Optimized Lightweight Model for Real-Time Wood Defects Detection based on YOLOv4-Tiny". **in** *2022 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*: 2022, **pages** 186–191. 10.1109/I2CACIS54679.2022.9815274.

[11] Wenqi Cui, Zhenye Li, Anning Duanmu, Sheng Xue, Yiren Guo, Chao Ni, Tingting Zhu **and** Yajun Zhang. "CCG-YOLOv7: A Wood Defect Detection Model for Small Targets Using Improved YOLOv7". **in** *IEEE Access*: 12 (2024), **pages** 10575–10585. 10.1109/ACCESS.2024.3352445.

[12] Rijun Wang, Yesheng Chen, Fulong Liang, Bo Wang, Xiangwei Mou **and** Guanghao Zhang. "BPN- YOLO: A Novel Method for Wood Defect Detection Based on YOLOv7". **in** *Forests*: 15.7 (2024). issn: 1999-4907. 10.3390/f15071096. https://www.mdpi.com/1999-4907/15/7/1096.

[13] Zhichao Liu **and** Baida Qu. "Machine vision based online detection of PCB defect". **in** *Microprocessors and Microsystems*: 82 (2021), **page** 103807. issn: 0141-9331. https://doi.org/10.1016/j.micpro.2020.103807. https://www.sciencedirect.com/science/article/pii/S0141933120309522.

[14] Cong Li, Hui-Qing Lan, Ya-Nan Sun **and** Jun-Qiang Wang. "Detection algorithm of defects on polyethylene gas pipe using image recognition". **in** *International Journal of Pressure Vessels and Piping*: 191 (2021), **page** 104381. issn: 0308-0161. https://doi.org/10.1016/j.ijpvp.2021.104381. https://www.sciencedirect.com/science/article/pii/S030801612100079X.

[15] Yufeng Shu, Bin Li **and** Hui Lin. "Quality safety monitoring of LED chips using deep learning- based vision inspection methods". **in** *Measurement*: 168 (2021), **page** 108123. issn: 0263-2241. https://doi.org/10.1016/j.measurement.2020.108123. https://www.sciencedirect.com/science/article/pii/S0263224120306618.

[16] Nikolaos Dimitriou, Lampros Leontaris, Thanasis Vafeiadis, Dimosthenis Ioannidis, Tracy Wotherspoon, Gregory Tinker **and** Dimitrios Tzovaras. "A Deep Learning framework for simulation and defect prediction applied in microelectronics". **in** *Simulation Modelling Practice and Theory*: 100 (2020), **page** 102063. issn: 1569-190X. https://doi.org/10.1016/j.simpat.2019.102063. https://www.sciencedirect.com/science/article/pii/S1569190X19301947.

[17] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani **and** Oral Büyüköztürk. "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types". **in** *Computer-Aided Civil and Infrastructure Engineering*: 33.9 (2018), **pages** 731–747.

[18] Sanli Tang, Fan He, Xiaolin Huang **and** Jie Yang. "Online PCB defect detector on a new PCB defect dataset". **in** *arXiv preprint arXiv:1902.06197* : (2019).

[19] Yiting Li, Haisong Huang, Qingsheng Xie, Liguo Yao **and** Qipeng Chen. "Research on a surface defect detection algorithm based on MobileNet-SSD". **in** *Applied Sciences*: 8.9 (2018), **page** 1678.

[20] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi **and** Alexander C Berg. "Dssd: Deconvolutional single shot detector". **in** *arXiv preprint arXiv:1701.06659* : (2017).

[21] Mingjie Liu, Xianhao Wang, Anjian Zhou, Xiuyuan Fu, Yiwei Ma **and** Changhao Piao. "Uav-yolo: Small object detection on unmanned aerial vehicle perspective". **in** *Sensors*: 20.8 (2020), **page** 2238.

[22] Haixin Huang, Xueduo Tang, Feng Wen **and** Xin Jin. "Small object detection method with shallow feature fusion network for chip surface defect detection". **in** *Scientific reports*: 12.1 (2022), **page** 3914.

[23] Liang Xu, Shuai Lv, Yong Deng **and** Xiuxi Li. "A weakly supervised surface defect detection based on convolutional neural network". **in** *IEEE Access*: 8 (2020), **pages** 42285–42296.

[24] Joseph Redmon, Santosh Divvala, Ross Girshick **and** Ali Farhadi. *You Only Look Once: Unified, Real- Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. https://arxiv.org/abs/1506.02640.

[25] Shuo Wang, Hongyu Wang, Fan Yang, Fei Liu **and** Long Zeng. "Attention-based deep learning for chip-surface-defect detection". **in** *The International Journal of Advanced Manufacturing Technology*:121.3 (2022), **pages** 1957–1971. issn: 1433-3015. 10.1007/s00170-022-09425-4. https://doi.org/10.1007/s00170-022-09425-4.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren **and** Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. https://arxiv.org/abs/1512.03385.

[27] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh **and** Dhruv Batra. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". **in** *International Journal of Computer Vision*: 128.2 (2020), **pages** 336–359. issn: 1573-1405. https://doi.org/10.1007/s11263-019-01228-7.