[1] **Theodoros Tziolas**

[2] **Konstantinos Papageorgiou**

[3] **Theodosios Theodosiou**

[4] **Sebastian Pantoja**

[5] **Nikolaos Dimitriou**

[6] **Elpiniki I. Papageorgiou**

Employing Deep Learning for Defect Detection in Antenna Assembly

Journal of Electrical Systems

Abstract: - Assembly processes involve disparate materials that possess dissimilar resiliencies and therefore are prone to generating defective products. Manually performed quality inspection of such products is a time-consuming and susceptible to errors process. The emerging computer vision techniques in smart manufacturing can alleviate the need for thorough manually performed quality control. Object detection techniques provide crucial localization abilities, thus helping the operators further validate the identified defect with ease. In this work, several state-of-the-art object detection models are assessed in a real industrial imagery dataset and with the use of transfer learning. EfficientDet D2 is proposed for the identification and the localization of antenna defects that are generated during the assembly process. To further enhance the dataset, heavy on-the-fly data augmentation is employed along with synthetic samples generated with the use of image processing software. The proposed approach utilizing EfficientDet D2 can increase the Average Precision from 0.90 (at IoU 0.5) to 0.97 (at IoU 0.3). The overall performance is further evaluated by applying the F1-Score at each confidence score. For conducting the experiments, the TensorFlow object detection API is employed.

Keywords: Deep Learning, Defect Detection, EfficientDet, Smart Manufacturing.

I. INTRODUCTION

Assembly operations are vital for the manufacturing process [1]. The assembly of materials and parts with different characteristics for the fabrication of the final product is typically one of the most common processes in manufacturing lines. In the case of antenna manufacturing, metallic source reflectors are jointed with plastic housing to form the final product. Despite the fact that modern manufacturing equipment handles such materials thoroughly, incorrect assemblies are highly anticipated due to various reasons. The complexity of the process and the overall quality of each distinct element are among the reasons that would be responsible for defective outcomes. On that basis, prime quality control is vital for ensuring the manufacturing of high-quality products.

Manual product inspection in an exclusive manner, is an obsolete strategy, as it induces many limitations, especially in large scale productions. To this, computer vision techniques have emerged as a way to handle massive amount of information in contemporary manufacturing equipment, in a faster and much more accurate way, compared to manual inspection performed by humans. Quality inspection based on computer vision can be described as a classification approach which distinguishes compliant from non-compliant to prescribed specifications products. As an enhancement of the classification approach, object detection (OD) is the process of estimating the location of the defect using a bounding box. This defect detection enhancement clearly poses an advantage when working in line with the operator for decision making, as it offers an explanation on what was identified as a defect.

Nevertheless, defect detection is differentiated from typical OD schemes, in which efforts are mainly focused on increasing the mean Average Precision (mAP). Although this metric is essential for OD contests, it doesn't provide insights about the False Positives (FP), and thus its application in a real industrial case entails limitations. In

[1] University of Thessaly, Gaiopolis Campus 41500, Larisa, Greece. ttziolas@uth.gr

[2] University of Thessaly, Gaiopolis Campus 41500, Larisa, Greece. konpapageorgiou@uth.gr

[3] University of Thessaly, Gaiopolis Campus 41500, Larisa, Greece. dozius@uth.gr

[4] Televés S.A.U,15706 Santiago de Compostela, Spain. spantoja@televes.com

[5] Centre for Research and Technology Hellas, Information Technologies Institute, 57001 Thermi-Thessaloniki, Greece. nikdim@iti.gr

[6] University of Thessaly, Gaiopolis Campus 41500, Larisa, Greece. elpinikipapageorgiou@uth.gr

addition, defects classes can be of several shapes, or sparse along the product, in comparison to OD standard datasets. Therefore, the overlapping area between ground truth and prediction boxes, or the Intersection over Union (IoU) threshold should also be examined. To this, OD needs to be customized based upon the needs of the industrial task.

The literature involves two categories for Convolutional Neural Network (CNN) object detectors [2]:

1) Single stage. These detectors can perform end-to-end object detection in a single shot. They are mostly used for real-time applications where fast inference is crucial. Such models are the YOLO [3], EfficientDet [4], CenterNet [5], etc.

2) Two-stage. These algorithms decouple the bounding boxes from the detections. In the first stage, the model proposes regions. Then, a dedicated network classifies these Regions of Interest (ROI). They are preferred when high accuracy is of high importance at the expense of computational effort, as they are considered slower but more accurate than the single-stage detectors [6]. Faster R-CNN [7] is a popular model that belongs in this group.

Each category has its own pros and cons, yet all require a massive amount of data for efficient learning. However, real industrial data are often scarce or imbalanced [8]. This poses a challenge in the development of robust solutions with deep learning, and hence transfer learning and data augmentation techniques have been employed in literature. TensorFlow Object Detection (TFOD) offers an Application Programming Interface (API) [9] which contains state-of-the-art pre-trained models. These models were trained on the Common Objects in Context (COCO) dataset [10] and are presented in TFOD with their metrics in mAP and inference speed. In addition to the provided models, TFOD offers flexibility in the customization of both the processing and the training parameters. Based on the presented metrics, an examination of the most promising models is performed in this work for the task of defect detection following the assembly of antennas. The EfficientDet models meet the criteria that are essential in real-time industrial applications, as they combine relatively high mAP with small processing latency.

The main contribution of this research work is the investigation of OD models from the EfficientDet family (D0 - D4) and the application of the most efficient one for defect detection in real-time production process. Moreover, the antenna assembly generates defects that affect the shape (housing imperfections), the surface (small cracks/breaks) and the overall product (cracks/breaks/missing parts). The accurate detection of these defects with the proposed approach can greatly improve the production efficiency and the overall quality of the products. In addition, the investigated models were further compared to both state-of-the-art single-stage and two-stage detectors such as the YOLOv7, the Faster R-CNN and models that were proposed in the literature, to further assess the strengths and weaknesses of each model.

The rest of the paper is organized as follows. The next section introduces the previous work and the basic concepts of OD. In the third section, the details of the methodology are further explained. The results of the applied methodology are presented and discussed in the fourth section. Finally, the work is summarized in the final section.

II. Literature and Basic Concepts

A. Related work

Deep learning OD has been widely used in smart manufacturing with promising results [11]. However, our proposed approach has not yet been adequately explored in literature. This ascertainment is based on the results provided after a rigorous search conducted by the authors. More specifically, a meticulous search utilizing the following combination of keywords “antenna AND (assembly OR manufacturing) AND defect detection” was performed in research databases such as Scopus and Google Scholar. Although these search engines gave back a plethora of results concerning articles in the field of antenna sensor development for metal quality inspection, these results were not relevant to the scope of the current research attempt. In fact, the only pertinent prior work that was found belongs to the authors of this manuscript. This prior work along with a literature review regarding defect detection methodologies in surfaces and/or materials with the use of EfficientDet is presented herein to further assess the abilities and the generalization of these models in the defect detection task.

Lite versions of EfficientDets that were developed with the TensorFlow Lite were examined in [12]. In this initial work of the authors, the efforts were focused on lowering the computational needs of the EfficientDet algorithms in order to be used within edge devices. TensorFlow Lite was used to compress and memory-optimize the original EfficientDet models. The derived models, even though capable of being deployed and run in edge devices, produced modest detection metrics with the proposed model EfficientDet D2 Lite, which achieved 73.68% AP at 0.5 IoU.

In another prior work of the authors [13], the advantages of CenterNets were assessed for the same task. In detail, CenterNet models with different backbone CNNs were examined. The CenterNet ResNet50 version achieved the highest AP of 0.94 at IoU 0.3. To further prove the effectiveness of the proposed OD approach, an accuracy comparison was performed with a binary classification approach. The work of [14], investigated the application of binary classification in this task further. Their novel method incorporated Fuzzy Cognitive Maps, state-of-the-art CNN models and transfer learning. This method enhanced the accuracy of the CNNs and the transparency in the decision-making. However, as evidenced by the findings in [13] this binary classification method ultimately underperforms compared to the proposed OD approach.

By utilizing a similar dataset that consisted of antenna images from laboratory measurements, the work of [15] proposed a two-phase training strategy to boost the performance of baseline classifiers while maintaining low inference times. They employed ResNets in their experiments and each acquired image was split into multiple image patches. Subsequently, when a defective patch was recognized by the classifier, its coordinates were used to estimate the location of the defect. Their methodology achieved 0.91 AP. Finally, they presented an IoT framework using Blockchain deployed in Private Ethereum.

Regarding the employment of EfficientDets in different domains of defect detection, literature results demonstrate that these models are applicable to ultrasonic material inspection [16], fabric [17], packaging [18] and PCB [19] defect detection. In the work of [16], the D0, D1 and D2 versions of EfficientDet family were examined. These EfficientDets were compared to RetinaNets with different backbone ResNets CNNs and to YOLOv3. The D0 version was proved to be the most efficient in detecting defects in steel blocks with 89.6% mAP which is a 9% increase in the mAP achieved by YOLOv3.

In [17], only the lightest EfficientDet version was examined, the D0, as the authors focused on developing an edge-computing solution with the use of a NVIDIA Jetson TX2 device. Their approach achieved both high mAP (>90%) and low latency in the examined datasets as the proposed scheme proved to be 2.5 faster than a cloud-based approach. EfficientDet D0 accuracy was compared to single-stage OD models such as YOLOv3, RetinaNet and Single Shot Detector (SSD) proving the efficiency of D0.

The work of [18] leveraged transfer learning and incorporated a channel attention network with the Mish activation function into a modified EfficientDet architecture. This approach achieved a mean Average Precision (mAP) of 99% in detecting defects in packaging bottles, which is a 2% improvement in the current state-of-the-art which consists of single-stage models such as YOLOv3/v4, and SSD.

In [19], the authors employed the EfficientDet D1 architecture for PCB defect detection. They utilized an open dataset augmented with data augmentation techniques. Furthermore, they compared their method to established OD models, including Faster R-CNN and RetinaNet. Their findings suggest that EfficientDet D1 offers a compelling balance between accuracy and speed.

In addition to the scarcity of research directly addressing antenna manufacturing defect detection, a further limitation identified in the existing literature is the restricted use of OD models for comparative analysis. While several OD models with diverse properties have been proposed [2] [11], many prior studies restrict their evaluation to a limited selection of models within the context of their tasks. Our work aims to provide a more comprehensive assessment by comparing the proposed approach with a wider variety of OD models.

B. Problem Formulation

The formal expression of the defect detection approach is described in the following. Given a sample set of N defected antennas depicted as 2D images, then for each image x_i , $x_i : i \in \mathbb{N}^*$, a mapping function is pursued with supervised learning to provide an output y_i.

$$y_i = \{(c_1, b_1, s_1), \ldots, (c_j, b_M, s_M)\} \quad (1)$$

where $j \in \mathbb{N}^*$ is the number of defect classes; b is the bounding box representing the two corner points in the height and width of the image [h_1, w_1, h_2, w_2]; $s \in (0,1]$ is the confidence score of the prediction, and $M \in \mathbb{N}^*$ is the number of defects in this sample.

Learning is performed by minimizing the combination of a classification and a localization loss:

$$\text{Total loss} = (w_1 \times \text{classification loss}) + (w_2 \times \text{localization loss}) \quad (2)$$

where w_1, w_2 are the assigned weights for biased penalization. The Jaccard Index or the IoU between prediction bounding boxes and ground truth is used to assess the True Positives (TP), the False Positives (FP) and the False Negatives (FN) predictions, and to define the Precision, Recall and F1-Score classification metrics. The IoU, the Precision, the Recall and the F1-Score are defined as:

$$\text{IoU} = \text{Area of Overlap} / \text{Area of Union} \quad (3)$$

$$\text{Precision} = TP / (TP + FP) \quad (4)$$

$$\text{Recall} = TP / (TP + FN) \quad (5)$$

$$\text{F1-Score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (6)$$

The Average Precision (AP) is employed as primary metric, with AP ∈ [0, 1]. AP is calculated as the Area Under Curve (AUC) of the Precision × Recall curve. For competition metrics such as COCO, the AP is the mean across 10 incrementing IoU values {0.5, 0.95} to reward localization across all classes. In this work, COCO metrics were primarily used to evaluate the training.

C. EfficientDet

EfficientDet [4] (D0-D7) is a family of single stage OD models targeting efficiency in terms of speed and accuracy. It utilizes several optimization and backbone architectures, like the EfficientNet [20] and the bi-directional Feature Pyramid Network (bi-FPN), to achieve state-of-the-art AP while being up to 9x smaller than detectors previously explored. It is also an approach with considerably less computational costs compared to detectors of prior works. EfficientNet is employed as a backbone feature extraction network, since it uniformly scales each dimension with a fixed set of scaling factors. The bi-directional Feature Pyramid Network (bi-FPN) is also employed for feature mapping. In short, the network receives multiple levels of features from the backbone as input, and outputs a list of fused features that represent the most important characteristics of an image. The exploitation of BiFPN improves the accuracy by 4%, while reducing the computational cost by 50%, as compared to alternative architectures, such as top-down Feature Pyramid Network (FPN) [21] as well as alternative FPNs such as the NAS-FPN [22]. Further details on the architecture of the EfficientDet are provided in [4].

III. METHODOLOGY

This section outlines the methodology employed to develop the proposed defect detection scheme utilizing transfer learning (Fig. 1). We begin by detailing the dataset and the data acquisition process. Subsequently, the data processing steps undertaken to enhance their suitability for the task are described. Finally, the configuration of the employed OD models is elucidated. Each of these steps is comprehensively addressed in the following subsections.

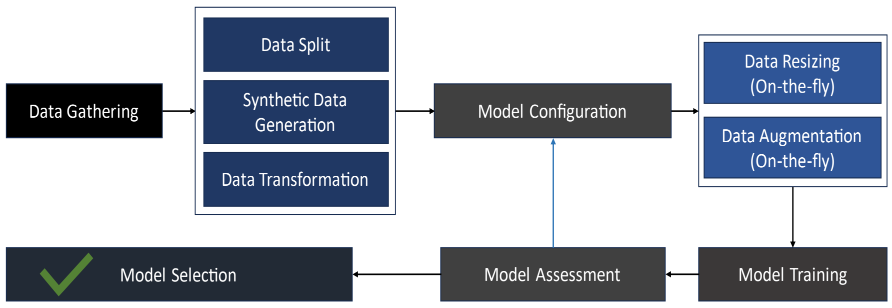

Fig. 1. The proposed methodology for antenna manufacturing defect detection

A. Dataset

The employed dataset contains images directly acquired from production and laboratory measurements, using 2D area scan cameras. The laboratory measurements were performed under ideal conditions, i.e., uniform background color, diffuse top light etc. The cameras used are a FLIR Blackfly S BFS-PGE-200S6C and a Baumer VCXG-241C. The output resolution of the sensors in width and height are 4401×2898 for the FLIR and 4096 × 3000 for the Baumer sensor, respectively. The dataset acquisition was focused on the defected samples that were scarce. Overall, the acquired dataset contains 161 defected samples and 25 healthy ones. Furthermore, the majority of the defected samples comprise more than one defects, with an average number of 2.5 per image, which further increases the variety and the total number of defects.

The defects were manually annotated with the use of LabelImg software [23] to produce the required xml files in the PASCAL VOC [24] format for the TFOD models and in a simple txt format for the YOLOv7. The defect labels assigned by experts belong in three groups: a) plastic break, b) metal break and c) housing imperfection. These are depicted in Fig. 2.

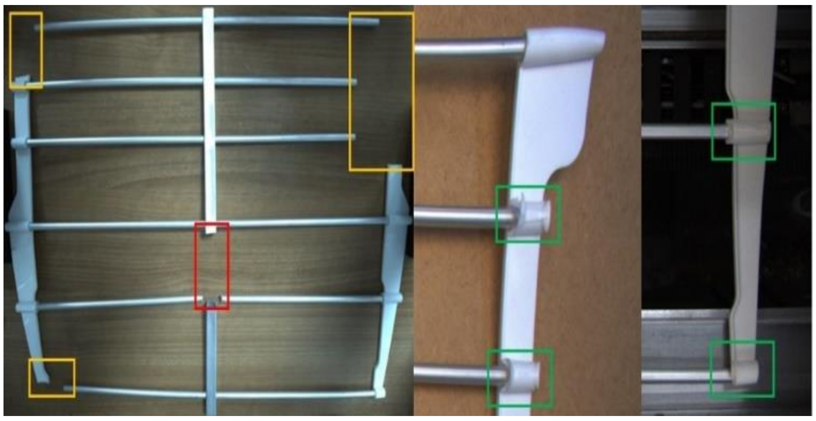

Fig. 2. Three classes of defects are considered. With red color the metal break, with yellow color the plastic breaks and with green color the housing imperfections

B. Preprocessing

The adopted data preprocessing strategy encompasses both offline and online methods. Offline methods, including data splitting, synthetic data generation, and data transformation, are applied prior to training. Online methods, defined during model configuration, involve data resizing and augmentation and are executed at each training iteration. For the sake of clarity and organization, all preprocessing methods are presented in this subsection.

1) Data Split

Of the defected samples, 80% (131 images) was kept for training and the remaining 20% (30 images) was used for testing. Subsequently, a small portion of these training images (20%, 24 images) was used as a separate dataset to validate the training procedure. Regarding the healthy samples (25 images), since they cannot be used in training, they were merged into the testing defectives to further examine the sensitivity in FPs. It is worth noting that data split was performed prior to synthetic data generation, to ensure that testing data were unique and hidden from the training stage.

2) Synthetic Data Generation

To further enrich the training and validation datasets, an image processing technique was used to mix and deform areas. This technique was primarily selected for the generation of rare defect types, such as breaks. By combining, inserting, or moving pixel areas from one image location to another, realistic defects were systematically produced. An example of this procedure can be seen in Fig. 3. However, only 24 new images (16 for training and 8 for validation) were created with this approach so as to prevent potential biases. The synthetic images were added to the originals and the overall number of images includes 123 for training, 32 for validation and 55 for testing.

3) Data Transformation

Finally, images and labels were transformed into tfrecord files, a simple format for storing a sequence of binary records. This binary format offers advantages in storage memory, pipelines, batch processing etc. when working with TensorFlow[25].

4) *Dimensionality Reduction*

As each model has its own requirements for input dimension and aspect ratio, the resize to the fixed aspect ratio of 1 was performed on the fly by zero-padding for the larger examined models (D3, D4, Faster R-CNN), and with bilinear interpolation for the rest (YOLOv7, D0, D1, D2). Zero-padding offers less training time without affecting accuracy [26], whereas bilinear interpolation was chosen to avoid shrinking the input information in small networks.

5) *Data Augmentation*

In addition, heavy data augmentation (DA) was performed on the fly with random adjustments in the color channels, in noise and in the structure of the images, at each training step. Apart from increasing the number of images that the model is treating, the employed DA also aimed to minimize the differences between the lab and the production datasets by inducing noise in images; thus, making the model robust in noisy backgrounds. More specifically, DA was performed using random horizontal/vertical flip, random crop/scale [0.5, 2.0], random adjust in hue/contrast/brightness/saturation, random RGB to gray, random self-concatenation vertical/horizontal, random jpeg quality, and random patch of gaussian noise. In Fig. 4 some arbitrary examples of the data augmentation outcomes are presented. These examples were accessed through TensorBoard which is a provided tool by the TensorFlow library that enables tracking of the training.

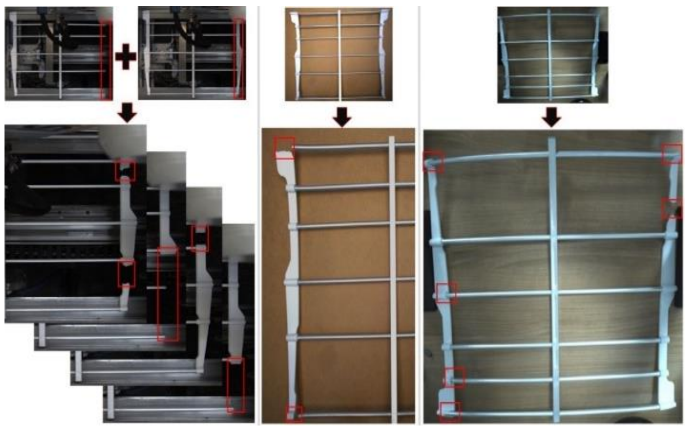

Fig. 3. Examples of synthetic defects

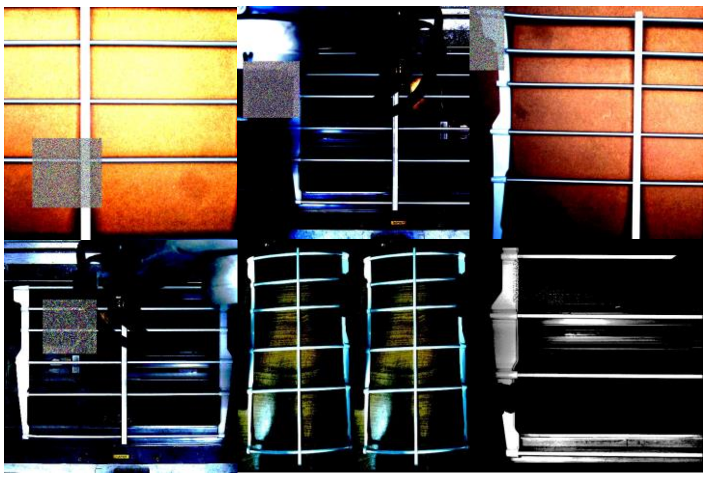

Fig. 4. Examples of random data augmentation processing outcomes

C. *Model Configuration*

TFOD API allows the configuration of the models via the respective pipeline file provided with the pre-trained model [9]. The configuration is accomplished through the use of Protocol Buffers which is based on structured data.

As the addressed problem includes three types of defects, the classification problem consists of three classes. In addition, the total maximum number of boxes was decreased from 100 (COCO dataset requirement) to 20 due to the fact that 20 defects are more than what is anticipated to exist in a single image of the examined dataset. Furthermore, the decrease of the total predicted bounding boxes results in faster postprocessing when Non-Max-Suppression (NMS) [27] is used. The main goal of model performance in the provided application was the classification accuracy rather than the localization accuracy.

The batch size was the first parameter to be adjusted for the EfficientDets to allow efficient, accurate and robust performance. The batch size was set to values {1, 1, 2, 4, 8} for each one of the examined versions {D4, D3, D2, D1, D0}, respectively, as higher sizes resulted in GPU memory failures. Due to the low batch sizes, the modification of the optimizer learning rate and training steps (i.e., gradient updates) was also essential. Since EfficientDets were trained on COCO with a batch size of 128, the same learning rate with smaller batch sizes resulted in training failure.

Following the original implementation of EfficientDets, the cosine decay learning rate and the momentum optimizer were used. The learning rate base was decreased from 0.08 to 0.004 while the warm-up learning rate was decreased from 0.001 to 0.0002. The maximum number of training steps during experiments was set to 120k. Regarding anchors, aspect ratios of 0.25 and 4.0 were added to the primary sizes {0.5, 1.0, 2.0}. For regularization purposes, a dropout with 0.3 probability was inserted into the box predictor. In addition, a higher weight (×1.2) was given to the sigmoid focal classification loss [28] instead of the Huber's (smooth L1) localization loss [29], to further penalize misclassifications. The focal loss is a modified version of the cross-entropy loss that performs well with class imbalance. Finally, transfer learning was performed in TFOD by fine-tuning all layers.

The NMS [27] algorithm was employed as a post-processing step to discard overlapping predictions of bounding boxes that belong to the same class. The algorithm calculates the IoU between predictions of the same class; if the IoU is higher than a predefined threshold only the prediction that has the highest confidence threshold is returned. Overlapping of defects was not anticipated in this task, therefore IoU was given the value of 0.3. Thus, on the occasion that defects overlap, even partially, the predictions of defects are considered that indicate the same defect.

IV. Results and Discussion

All the experiments were performed in a workstation equipped with an Intel Core i9-11900KF @ 3.5GHz CPU, 16 GB RAM and NVIDIA GeForce RTX 3080 Ti with 12GB of GDDR6X memory, running Windows 10 Professional. The software was implemented in the Python language with the TFOD API (TensorFlow v.2.9.1).

The experimentation process involved the initialization and the training of several separate model instances to acquire the proper hyperparameters and the best weights for each model. The minimum IoU value in OD metrics of 0.5 was compared with a smaller IoU threshold of 0.3 to emphasize the classification potentials of the model by allowing the localization to be less precise. The EfficientDet models were further compared to state-of-the-art single-stage and two-stage detectors. For the former category, the lightweight CenterNets of [13] and the YOLOv7-x were examined which were designed for real-time processing. For the latter, the well-known Faster R-CNN model with the Inception ResNet [30] as a backbone network was exploited. Regarding YOLOv7-x, the PyTorch official edition was utilized. The anchors in YOLOv7 were initialized based upon the K-means algorithm as in the original implementation, whereas the EfficientDets and the Faster R-CNN shared the same anchor parameters. In addition, the Faster R-CNN was modified to have 100 region proposals to be more time-efficient without sacrificing accuracy, as suggested in [9].

The best-performed models along with their values on the following performance metrics are depicted in Table 1. F1-score was calculated for each confidence score, and its highest values for each IoU are presented below. The minimum confidence score was set to 10%, meaning that below this score all the predictions are considered FPs and are omitted. The latency is calculated for both GPU and CPU processing, considering the potential hardware limitations in an industrial implementation. As far as the average inference speed estimation is concerned, it should be noted that the time needed for the first image is ignored due to internal resources initialization. Moreover, time per training step is calculated as a metric for the training duration which depends on the batch size, the architecture of the network and the preprocessing parameters that are performed on the fly.

The examined models of Table 1 can be divided in two categories based on their speed/accuracy trade-off for further assessment. The first category involves the lightweight models such as the CenterNets, the YOLOv7-x and the EfficientDets D0-D1. It is observed that the 13ms GPU latency of the YOLOv7-x indicates real-time 60 Frame Per Second (FPS) detection. However, the YOLOv7-x underperforms in defect detection abilities, as the low AP and F1-score sets limitations on industrial implementation. Regarding the D0 and D1 models, their metrics are mediocre for this task. In contrast, the best trade-off for this category is offered by the CenterNets from previous work, as they achieved the highest combination of F1-Score and AP among the lightweight models, and the fastest CPU inference. For the second category and the heavier models, it is evident that EfficientDet D2 has an advantageous performance. The AP of 0.97 indicates high detection capabilities, whereas F1-score implies model robustness against FPs. Furthermore, the time required for processing is less than a second even for CPU running, which means that the model can be implemented in the examined production process without any hardware limitations. Even though EfficientDet D3 and D4 are advanced versions, they failed to surpass D2, in terms of performance. Similarly, the two-stage Faster R-CNN detector despite achieving the second highest detection metrics, requires considerable processing time especially for CPU hardware. To this end, the CenterNet MobileNetV2, the CenterNet ResNet50, and the EfficientDet D2 are the most promising models for the requirements of this task. However, EfficientDet D2 is preferred as it exceeds over CenterNets in AP and F1-score, with a substantial difference, though.

Table 1 Performance metrics of the examined models. With bold the best metrics.

Model	***AP @ 0.5 IoU***	***F1-Score @ 0.5 IoU***	***AP @ 0.3 IoU***	***F1-Score @ 0.3 IoU***	***Latency GPU (ms)***	***Latency CPU (s)***	***Time per step (s)***	***Batch size***
CenterNet MobileNetV2 [13]	0.80	0.86	0.93	0.93	26	**0.17**	3.1	16
CenterNet ResNet50 [13]	0.88	0.89	0.94	0.94	28	0.19	3.0	8
YOLOv7-x	0.81	0.80	0.82	0.81	**13**	0.64	2.4	8
Faster R-CNN	0.85	0.89	0.94	0.96	112	1.70	1.3	2
EfficientDet D0	0.60	0.74	0.84	0.86	57	0.19	0.7	8
EfficientDet D1	0.78	0.80	0.89	0.90	65	0.47	0.9	4
EfficientDet D2	**0.90**	**0.93**	**0.97**	**0.98**	98	0.65	0.6	2
EfficientDet D3	0.84	0.89	0.92	0.96	128	1.16	0.6	1
EfficientDet D4	0.84	0.87	0.91	0.94	191	1.80	0.8	1

An insight of the training process is given in Fig. 5 which illustrates losses, in both training and validation. It is observed that the training curve of the EfficientDet D2 is fluctuated compared to the validation curve and the CenterNet MobileNetV2 training curve that are considerably smoother. This happens due to the low batch size of 2 that was used in training and resulted in noisy learning. In addition, it is observed that both classification and total loss present high similarity with respect to the higher weight that was passed to the classification loss.

A critical issue that needs to be elaborately investigated is the determination of the confidence threshold to filter out FPs. FPs with high confidence thresholds indicate models' sensitivity to misclassifications. On that basis, AP needs to be calculated and further assessed through the Precision×Recall curve so that the proper threshold for filtering FPs is revealed. This can be attained by ranking the model's total detections in a descending order of their confidence scores achieved in the test dataset, thus revealing where the highest metrics occurred. Each detection is classified either as TP or FP based on the selected IoU threshold, and then the accumulated TP (Acc TP) and FP (Acc FP) for each score are calculated. However, AP, as previously stated, cannot provide adequate insights regarding the FP performance. For this reason, F1-Score is also calculated for each score. Instances of confidence scores in descending order for the EfficientDet D2 are depicted in Table 2. In more detail, the optimum combination of the best AP and F1-score occurs in the 85th detection for the EfficientDet D2. Thus, the confidence score of this prediction can be defined as a threshold for filtering detections. Moreover, the proper confidence threshold for filtering EfficientDet D2 detections can be pursued in between 0.25 and 0.19, where the former minimizes the FPs whereas the latter increases the detection ability.

Table 2 Order where the highest metrics (in bold) were appeared in the test dataset predictions for the EfficientDet D2.

Nr. Detection	***Confidence score***	***IoU***	***TP***	***FP***	***Acc TP***	***Acc FP***	***Precision***	***Recall***	***AP***	***F1-score***
83	0.27	0.6	✓	-	83	0	1.0	0.95	0.95	0.98
84	0.24	0	-	✓	83	1	0.99	0.95	0.95	0.97
85	**0.19**	0.7	✓	-	84	1	0.99	0.97	**0.97**	**0.98**

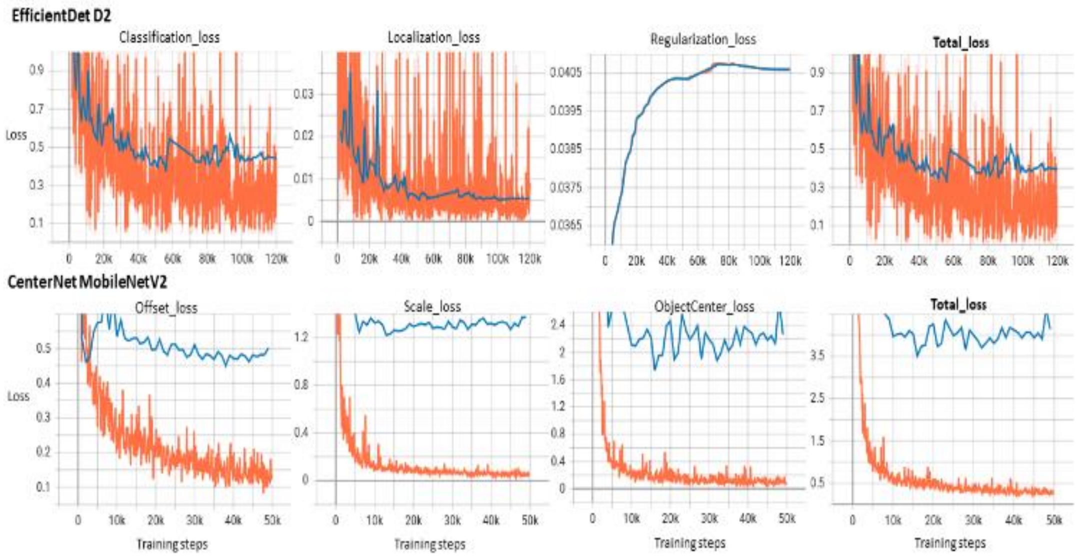

Fig. 5 The losses of the EfficientDet D2 and the CenterNet MobileNetV2 of [13]. Training loss in orange and Validation loss in blue.

Regarding the necessity for a lower IoU value (at 0.3) resulting in accurate defect detection, as presented in Table 1, this can be further explained by visualizing TP detections of the experiments with such low IoU values (Fig. 6).

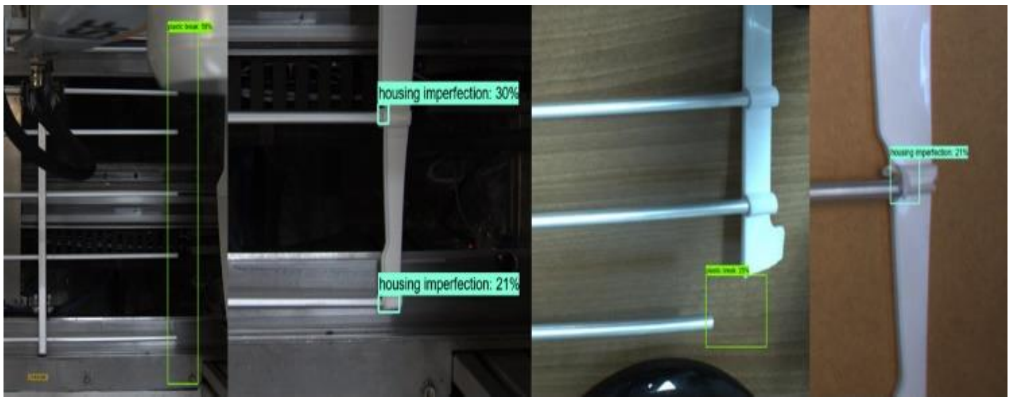

Fig. 6 The qualitative results of detections that have IoU <0.5. In the first and the third images break detections are presented, whereas housing imperfections are presented in the second and the last.

As for the breaks, the model struggled to correctly locate such defects since there is no consistency in their shape and size. This also happens in the case of housing imperfections. Experts annotated the defects at the points where either the cap was deformed by rough edges or tiny cracks occurred from poor insertion. In most of the cases, these deformations mostly occurred at the point of insertion, whereas on some occasions such as the one in the last image, both edges of the cap were deformed. Thus, the whole cap was annotated as defected. However, only the point with the most noticeable deformation provided a more robust output of the model, thus resulting in lower IoU.

V. Summary

This work examined the process of defect detection following the assembly of antennas in a real industrial dataset, implementing an efficient methodology based on state-of-the-art object detection models. The proposed approach involved synthetic samples as well as heavy data augmentation to further enrich the limited samples. Subsequently, state-of-the-art models were assessed with transfer learning to alleviate the scarcity of this dataset and to explore and determine proper detectors of high accuracy and low latency for the task of defect detection. EfficientDet D2 is proposed in this study as a particularly efficient classification model, which can achieve an AP of up to 97%, when lowering the IoU threshold to 0.3. The near-real-time inference execution speed of 98 ms in GPU and 650 ms in CPU indicates satisfying compliance with the requirements of the production times. To highlight the efficiency of the proposed methodology and assess its performance, the proposed model was examined side by side with other EfficientDet versions and other state-of-the-art models such as CenterNet variants, the Faster R-CNN and the YOLOv7.

Future work will be oriented toward the inclusion and testing of more defected samples. Therefore, possible limitations of the current approach regarding data scarcity will be eliminated. Furthermore, in the development stage, data transmission latency as well as the overall integration of the scheme on cloud and edge devices will be further assessed. This assessment may uncover potential limitations in the proposed model, particularly regarding inference execution speed and the time required for near-real-time monitoring.

Acknowledgement

This work has supported by EU Project OPTIMAI (H2020-NMBP-TR-IND-2020-singlestage, Topic: DT-FOF-11-2020, GA 958264). The authors acknowledge this support.

References

[1] D. Ben-Arieh, R. Ranjan Kumar, and M. K. Tiwari, “Analysis of assembly operations’ difficulty using enhanced expert high-level colored fuzzy Petri net model,” *Robotics and Computer-Integrated Manufacturing*, vol. 20, no. 5, pp. 385–403, Oct. 2004, doi: 10.1016/j.rcim.2004.03.002.

[2] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, “A survey of modern deep learning based object detection models,” *Digital Signal Processing*, vol. 126, p. 103514, Jun. 2022, doi: 10.1016/j.dsp.2022.103514.

[3] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.” arXiv, 2022. doi: 10.48550/ARXIV.2207.02696.

[4] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and Efficient Object Detection.” arXiv, Jul. 27, 2020. Accessed: Dec. 28, 2022. [Online]. Available: http://arxiv.org/abs/1911.09070

[5] X. Zhou, D. Wang, and P. Krähenbühl, “Objects as Points.” arXiv, Apr. 25, 2019. Accessed: Dec. 28, 2022. [Online]. Available: http://arxiv.org/abs/1904.07850

[6] A. Lohia, K. D. Kadam, R. R. Joshi, and A. M. Bongale, “Bibliometric analysis of one-stage and two-stage object detection,” *Libr. Philos. Pract*, vol. 4910, p. 34, 2021.

[7] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.

[8] T. Tziolas *et al.*, “Wafer Map Defect Pattern Recognition using Imbalanced Datasets,” in *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 2022, pp. 1–8. doi: 10.1109/IISA56318.2022.9904402.

[9] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[10] T.-Y. Lin *et al.*, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[11] H. M. Ahmad and A. Rahimi, “Deep learning methods for object detection in smart manufacturing: A survey,” *Journal of Manufacturing Systems*, vol. 64, pp. 181–196, Jul. 2022, doi: 10.1016/j.jmsy.2022.06.011.

[12] A. Feleki *et al.*, “EFFICIENTDET APPLICATION FOR DETECTION OF INCORRECT ASSEMBLIES IN THE ANTENNA MANUFACTURING PROCESS,” in *10th ECCOMAS Thematic Conference on Smart Structures and Materials*, Patras, Greece: Dept. of Mechanical Engineering & Aeronautics University of Patras, 2023, pp. 1270–1278. doi: 10.7712/150123.9874.444639.

[13] T. Theodosiou *et al.*, “CENTERNET-BASED MODELS FOR THE DETECTION OF DEFECTS IN AN INDUSTRIAL ANTENNA ASSEMBLY PROCESS,” in *10th ECCOMAS Thematic Conference on Smart Structures and Materials*, Patras, Greece: Dept. of Mechanical Engineering & Aeronautics University of Patras, 2023, pp. 1209–1220. doi:

10.7712/150123.
9869.444634.

[14] T. Tziolas *et al.*, “Deep Fuzzy Cognitive Maps for Defect Inspection in Antenna Assembly,” *Procedia Computer Science*, vol. 232, pp. 97–106, 2024.

[15] L. Leontaris *et al.*, “A blockchain-enabled deep residual architecture for accountable, in-situ quality control in industry 4.0 with minimal latency,” *Computers in Industry*, vol. 149, p. 103919, 2023.

[16] D. Medak, L. Posilović, M. Subašić, M. Budimir, and S. Lončarić, “Automated defect detection from ultrasonic images using deep learning,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 68, no. 10, pp. 3126–3134, 2021.

[17] S. Song, J. Jing, Y. Huang, and M. Shi, “EfficientDet for fabric defect detection based on edge computing,” *Journal of Engineered Fibers and Fabrics*, vol. 16, p. 15589250211008346, 2021.

[18] Z. Sheng and G. Wang, “Fast Method of Detecting Packaging Bottle Defects Based on ECA-EfficientDet,” *Journal of Sensors*, vol. 2022, no. 1, p. 9518910, 2022.

[19] J. Jin *et al.*, “Defect Detection of Printed Circuit Boards Using EfficientDet,” in *2021 IEEE 6th International Conference on Signal and Image Processing (ICSIP)*, 2021, pp. 287–293. doi: 10.1109/ICSIP52628.2021.9688801.

[20] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.

[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection.” arXiv, 2016. doi: 10.48550/ARXIV.1612.03144.

[22] G. Ghiasi, T.-Y. Lin, R. Pang, and Q. V. Le, “NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection.” arXiv, 2019. doi: 10.48550/ARXIV.1904.07392.

[23] Tzutalin, “LabelImg.” 2015. [Online]. Available: https://github.com/tzutalin/labelImg

[24] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[25] “TFRecord and tf.train.Example | TensorFlow Core.” Accessed: Mar. 11, 2024. [Online]. Available: https://www.tensorflow.org/
tutorials/load_data/tfrecord

[26] M. Hashemi, “Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation,” *Journal of Big Data*, vol. 6, no. 1, p. 98, Nov. 2019, doi: 10.1186/s40537-019-0263-7.

[27] A. Neubeck and L. Van Gool, “Efficient non-maximum suppression,” in *18th International Conference on Pattern Recognition (ICPR’06)*, IEEE, 2006, pp. 850–855.

[28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[29] P. J. Huber, “Robust Estimation of a Location Parameter,” *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964, doi: 10.1214/aoms/1177703732.

[30] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-first AAAI conference on artificial intelligence*, 2017.