Available online at www.eccomasproceedia.org
Eccomas Proceedia SMART (2023) 1196-1208

ECCOMAS
Proceedia

X ECCOMAS Thematic Conference on Smart Structures and Materials
SMART 2023
D.A. Saravanos, A. Benjeddou, N. Chrysochoidis and T. Theodosiou (Eds)

AUTOMATED DEFECT DETECTION IN BATTERY LINE ASSEMBLY VIA DEEP LEARNING ANALYSIS

Anastasios Tzelepakis*, Lampros Leontaris* Nikolaos Dimitriou*, Evangelia Koukidou†, Dimitrios Bollas†, Aristoklis Karamanidis† and Dimitrios Tzovaras*

*Information Technologies Institute
Centre for Research and Technology Hellas
Thermi, 57001 Thessaloniki, Greece

† Sunlight Group Energy Storage Systems
Kifisiá, 14564 Attica, Greece

Abstract. Recent technological achievements in computer vision and machine learning have provided promising solutions in industrial quality control. As automated solutions are hard to integrate in the manufacturing process, a common practice during battery line assembly involves the manual investigation of the battery parts which is both inefficient and time consuming. We focus on a challenging production stage in the assembly line where human inspection is not feasible and the appeared defects can only be inspected in the later stages of production. For this purpose, we propose an in-situ system that automates the quality control process and performs defect diagnosis by accurately identifying anomalies in the current production stage. The implemented system aims to monitor the production line and visualize defective occurrences in battery assembly line, by utilizing deep neural networks (DNNs) and examining the defects on real production samples, collected with a machine vision system. In order to decide the optimal configuration for the specific task, we perform a cross-evaluation of various state-of-the-art (SOTA) DNN architectures, specialized in object detection. Furthermore, we explore copy-paste data augmentation mechanisms to generate additional training data from a small number of defective samples. The system is initially validated on the localization of defects in industrial samples, using mean average precision (mAP) as a metric for the performance evaluation and then validated on the classification of defective and non-defective samples, using F-score, precision and recall as evaluation metrics.

Key words: Deep learning (DL), Smart Manufacturing, Defect object detection, Battery manufacturing industry, Production line monitoring

Peer-review under responsibility of the organizing committee of SMART 2023.
doi: 10.7712/150123.9868.444540

1 INTRODUCTION

The increasing concern over the environmental crisis has led to a growing need for eco-friendly energy storage options. The development of reliable and efficient systems has become imperative in this regard. The introduction of artificial intelligence (AI) technologies into battery production lines has the potential to significantly reduce the occurrence of errors during the assembly process. Intelligent quality control systems are being developed in battery manufacturing and assembly lines to enhance the reliability the products and to drive investment and purchase interest towards the use of electric vehicles [1], as well as renewable energy sources [2]. Therefore, integrated solutions [3] that include Internet of Things (IoT) networks for real-time data observation and acquisition, AI systems that utilize sensor data to determine the battery's state at each stage of the assembly process, and an intelligent warning system that consistently notifies the technical staff of potential anomalies can contribute to the creation of a zero-defect industry.

The manufacturing process of batteries involves assembling individual components through several stages, which constitutes the majority of the process in a continuous production line. Each stage introduces varying degrees and types of wear and inaccuracy, necessitating separate research into each stage. This study focuses on the process of wrapping an insulating sheath, called a separator, which provides physical and electrical isolation between opposite electrodes [4]. Common defects that occur during this stage include holes, abrasions, and wrinkles on the insulating casing's surface [5]. Furthermore, misplacements of the separator around the electrodes and on top of the production line are prevalent. Batteries with the wrong orientation on the production line and elements protruding from the separator that should naturally wrap them are frequent occurrences. The separator is the primary safety component of the battery cell and thus, it is critical to detect defects accurately and on time. Furthermore, production costs from raw material waste, material replacement and logistic expenses are also important. Therefore, accurate fault detection and localization are crucial in reducing the aforementioned issues and producing defect-free battery separators.

Machine vision can be an effective method for automatically detecting battery separator defects. However, implementing machine vision systems in manufacturing processes is challenging due to the diverse materials, structures, and properties of battery separators. Deriving image processing parameters for separators with dissimilar optical properties requires manual adaptation and expert knowledge, resulting in the use of costly service engineers. To overcome these challenges, researchers have proposed optimized machine vision system designs and a methodical approach for faster parameter fitting, using deep neural networks.

In this study:

- we apply copy paste technique for data augmentation of defective samples from battery production line.
- we benchmark five state-of-the-art object detection algorithms on the task of defect detection.
- we investigate the best approach, in both performance and inference time, to automate

defect detection in battery assembly lines.

2 METHODOLOGY

This section provides a comprehensive account of the implementation procedure, which focuses on the proposed neural network models, namely Faster R-CNN, RetinaNet, Mask R-CNN, YOLOv7, and YOLOv8. An analysis of the architecture of each model is presented to shed light on its design and functionality. Moreover, a thorough description of the constructed dataset for the study is given, followed by an explanation of the metrics utilized to evaluate the models' performance.

2.1 PROPOSED NETWORKS

In recent years, Deep Learning (DL) approaches have revolutionized the field of defect detection [6] [7], with methods based on convolutional neural networks (CNNs) achieving state-of-the-art performance. These methods typically consist of a backbone network that extracts features from the input image and a detection head that performs object detection using these features.

Object detection refers to the process of identifying and locating objects in images or videos and is critical in computer vision for various applications such as surveillance, self-driving cars, and medical imaging. Object detection methods are categorized into two types: single-stage and two-stage methods. Single-stage methods directly predict object class labels and bounding boxes in a single step and are faster than two-stage methods. However, they can suffer from lower accuracy due to processing large numbers of object proposals. Two-stage methods generate object proposals and refine them to provide accurate bounding boxes and class labels. These methods can be slower but achieve higher accuracy, making them suitable for applications where precision is critical. Examples of single-stage methods include YOLO [8] and SSD [9], while R-CNN [10] and its variants such as Fast R-CNN [11] and Faster R-CNN [12] are examples of two-stage methods.

In the subsections, below, we describe five object detection algorithms for evaluating defect detection on a dataset containing defective images of battery separators.

2.1.1 Faster-RCNN

The Faster R-CNN [12] is a highly accurate object detection algorithm that uses a two-stage framework to detect objects in images. It is based on the Region-Based Convolutional Neural Network (R-CNN) [10] architecture and consists of a Region Proposal Network (RPN) [12] and a Fast R-CNN [11] detection network. The algorithm employs a Feature Pyramid Network (FPN) [13] to generate a feature map that captures object details at multiple scales and resolutions. The RPN generates object proposals by predicting objectness scores and bounding box offsets for each anchor box at multiple scales and aspect ratios. The proposals are then passed to the Fast R-CNN detection network, which extracts features from the proposed regions and uses them to classify and refine the bounding box coordinates of the objects. The algorithm is trained using a

multi-task loss function that jointly optimizes the RPN and the Fast R-CNN network. The loss function comprises two components, a classification loss and a bounding box regression loss:

$$L = L_{cls} + L_{box} \tag{1}$$

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2}$$

where, p_i is the predicted probability of anchor i being a positive sample, p_i^* is the binary groundtruth label specifying whether anchor i is a positive sample, t_i are the predicted four coordinate parameters, t_i^* are the ground truth coordinates, N_{cls} is the normalization term for the classification task, N_{box} is the normalization term for the regression of the bounding box coordinates, λ is a parameter used to balance the weights of the classification loss L_{cls} and the bounding box regression L_{box} and, finally, L_{reg} is the smoothed $L1$ loss.

2.1.2 RetinaNet

RetinaNet [14] is a leading-edge one-stage object detection algorithm that utilizes a Feature Pyramid Network (FPN) [13] to enable the detection of objects of different sizes and scales in images. The RetinaNet architecture utilizes an FPN to produce a feature pyramid that includes features from multiple scales. It also employs a Feature Fusion module to combine features from different scales to improve the accuracy of object detection. In the detection subnetwork, RetinaNet predicts objectness scores and bounding box regression offsets for each anchor box at multiple scales and aspect ratios.

$$CE(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise} \end{cases} \tag{3}$$

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases} \tag{4}$$

$$CE(p_t) = -\log(p_t) \tag{5}$$

To tackle the class imbalance problem, RetinaNet introduces a focal loss function that modifies the standard cross-entropy loss by down-weighting the contribution of well-classified examples and increasing the contribution of misclassified examples.

$$CE(p_t) = -\alpha_t \log(p_t) \tag{6}$$

$$FL(p_t) = -(1-p_t)^\gamma \log(p_t) \tag{7}$$

$$FL(p_t) = -\alpha_t (1-p_t)^\gamma \log(p_t) \tag{8}$$

Here, $y \in \pm 1$ specifies the ground truth class and $p \in [0, 1]$ is the model's estimated probability for the class with label $y = 1$, $a \in [0, 1]$ is a weighting factor, which the value a for class 1 and $1 - a$ for the class -1. This weighting factor handles the class imbalance in the dataset. $(1 - p_t)^\gamma$ is a modulating factor to the cross entropy loss, with tunable *focusing* parameter $\gamma \geq 0$.

2.1.3 Mask R-CNN

The Mask R-CNN [15] algorithm is an advanced object detection and instance segmentation method that improves upon the R-CNN [10] architecture by adding a third branch to predict object masks, in addition to object bounding boxes and objectness scores. The architecture incorporates a Feature Pyramid Network (FPN) [13] that generates feature maps at different scales and uses a Region of Interest (RoI) Align layer [15] to enrich them with spatial information. The network generates region proposals using a Region Proposal Network (RPN) [12] that predicts objectness scores and bounding box offsets for every anchor box at multiple scales and aspect ratios. The RoI Align layer then extracts features from the feature maps corresponding to each proposal, and the three branches predict the objectness score, bounding box offset, and object mask. Finally, the object mask branch generates a binary mask for each RoI by applying a sigmoid function to the output of a fully convolutional network. Mask R-CNN is trained using a multi-task loss function that combines the classification loss, the bounding box regression loss, and the mask loss:

$$L = L_{cls} + L_{box} + L_{mask} \tag{9}$$

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \log \widehat{y}_{ij}^{k} + (1 - y_{ij}) \log(1 - \widehat{y}_{ij}^{k}) \right] \tag{10}$$

Here L_{cls} and L_{box} are identical to the ones used in Faster R-CNN, y_{ij} is the label of a cell (i, j) in the groundtruth mask for the region of size $m \times m$, $\widehat{y}_{ij}^{k}$ is the predicted value of the same cell in the mask learned for the groundtruth class k.

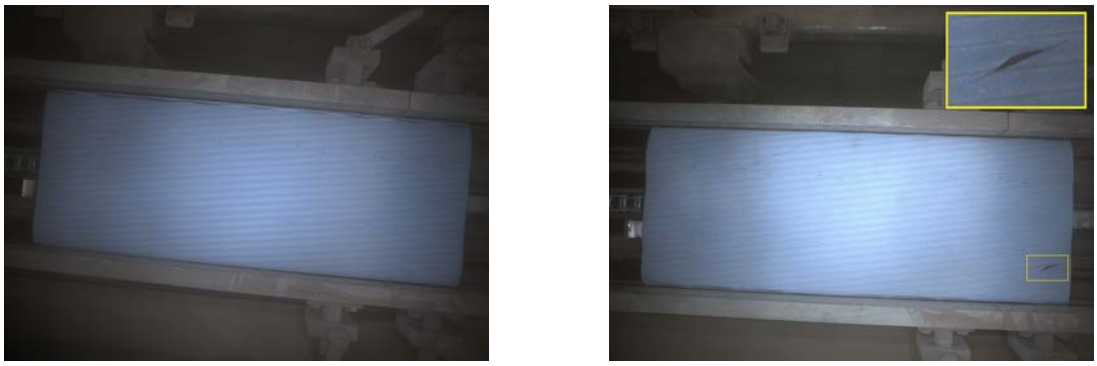
2.1.4 YOLOv7 & YOLOv8

The YOLO [8] object detection algorithm is an efficient real-time system that treats object detection as a regression problem. YOLO uses a single neural network to simultaneously predict class probabilities and bounding box coordinates of multiple objects in a given image. The efficiency of the convolutional layers is crucial for achieving fast inference speeds. To improve learning power, YOLOv7 [16] minimizes the gradient distance by selecting a final layer aggregation method, which includes an extended version of the ELAN computational block called E-ELAN [16]. A new technique for scaling the network depth and width simultaneously while concatenating layers together is proposed, which maintains optimal model architecture while allowing for scalable model size. Re-parameterization techniques aim to enhance the robustness of the model by averaging a set of model weights. Module-level re-parameterization techniques, which target specific components of the network with distinct re-parameterization strategies, can improve the generalization and robustness of object detection models. The addition of an auxiliary head at an intermediate location can be beneficial in improving the performance of the model, and experiments were conducted to explore different levels of supervision for the auxiliary head in YOLOv7. YOLOv8 [17] is a highly recognized model for image segmentation

tasks due to its exceptional speed, accuracy, flexibility, and enhanced developer experience improvements. Its advanced architecture, pre-trained models, and unified framework for training models have made it a preferred solution for real-time applications including object detection, instance segmentation, and image classification. These attributes make YOLOv8 a state-of-the-art model for image segmentation tasks.

2.2 DATASET

Acquiring defective image samples from the production line, where the error rate is typically small, is a challenging task. Industries with a low error rate often have robust quality control procedures in place, which prevent defective products from reaching the end consumer. Additionally, it is time consuming and costly to collect defective samples from production, while creating a large and diverse dataset is critical for training AI models. In the examined production stage, various types of defects can appear including: perforations, ruptures, imprinted markings, and paint residue, indicating faults in the separator material. Additionally, instances of misaligned seam placement, exposition of the positive plate, wrinkling at the separator border, and improper orientation of electrodes, commonly referred to as lag, were also noted. An indicative example of a normal and a defective sample is depicted in Figure 1, while the various defect types are presented in Figure 2.

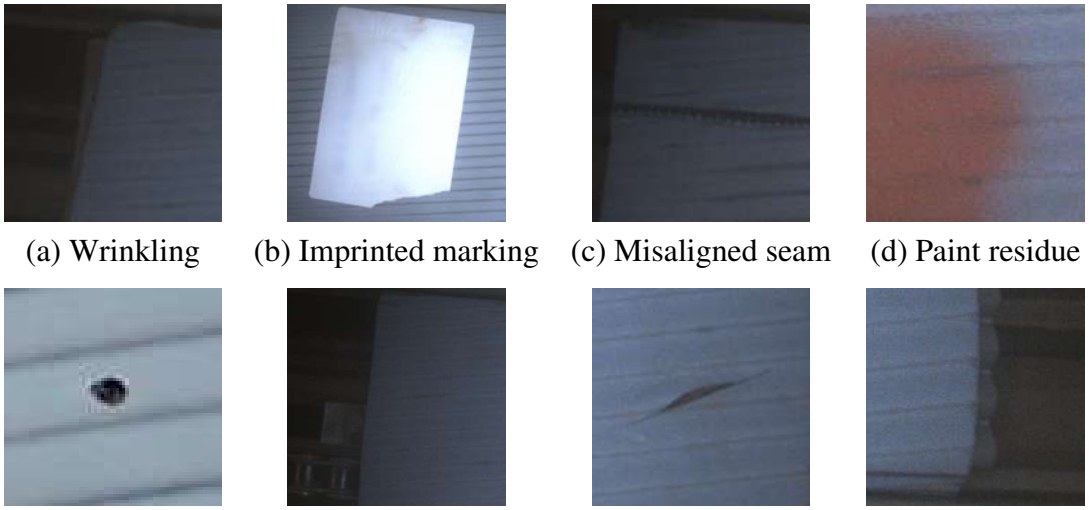


(a) Normal separator **(b)** Defective separator

Figure 1: Example of a normal (a) and a defective (b) separator, with the latter containing ruptures on its surface.

In our work, we address the problem of data scarcity, applying a copy-paste data augmentation procedure to generate sufficient amount of data for DL training. By using techniques such as image cropping, rotation, and scaling, it is possible to create a large and diverse dataset from a small number of samples [18]. This can be particularly useful for training AI models to detect and classify defects, as it allows the model to learn from a variety of different examples and become more robust to variations in the appearance of defects.

Using a machine vision camera, Imaging Source DFK 38GX304 12Mpixels, installed in the production line, we obtained color images of size 3000×4096 from 15 defective samples and



(a) Wrinkling (b) Imprinted marking (c) Misaligned seam (d) Paint residue

(e) Perforation (f) Lag (g) Rupture (h) Exposed plate

Figure 2: Different types of defects.

2000 normal samples. The defective samples represented all the defective categories that could appear in the production. Through copy-paste augmentation, we created a dataset comprising 4000 defective samples with dimensions, containing a total of 5990 defect instances. In Table 1 we see in detail the instances per category, included in train and test set. Each generated image includes one or two defect instances. We meticulously considered the approximate positioning of each defect type on the battery separator's surface. We achieved this by extracting a grid of all possible locations where a particular defect can occur and then pasting the cropped defect at the designated point. An indicative example is illustrated in Figure 3. We see, that given a pair of defective and normal sample, we use accurately annotated defective regions of the defective samples and copy-paste them to the normal samples. We use 80% of the total data for training and the remaining 20% for testing, resulting in 3200 training images, and 800 testing images.

Table 1: Number of instances per defect type in training and test set.

Defect type	Wr[a]	Im[b]	Ms[c]	Pr[d]	Per[e]	Lag	Rup[f]	Exp[g]	Total
Training set	1445	502	471	520	522	439	446	435	4780
Test set	357	120	125	118	113	126	126	125	1210

[a] Wrinkling, [b] Imprinted marking, [c] Misaligned seam, [d] Paint residue, [e] Perforation, [f] Rupture, [g] Exposed plate

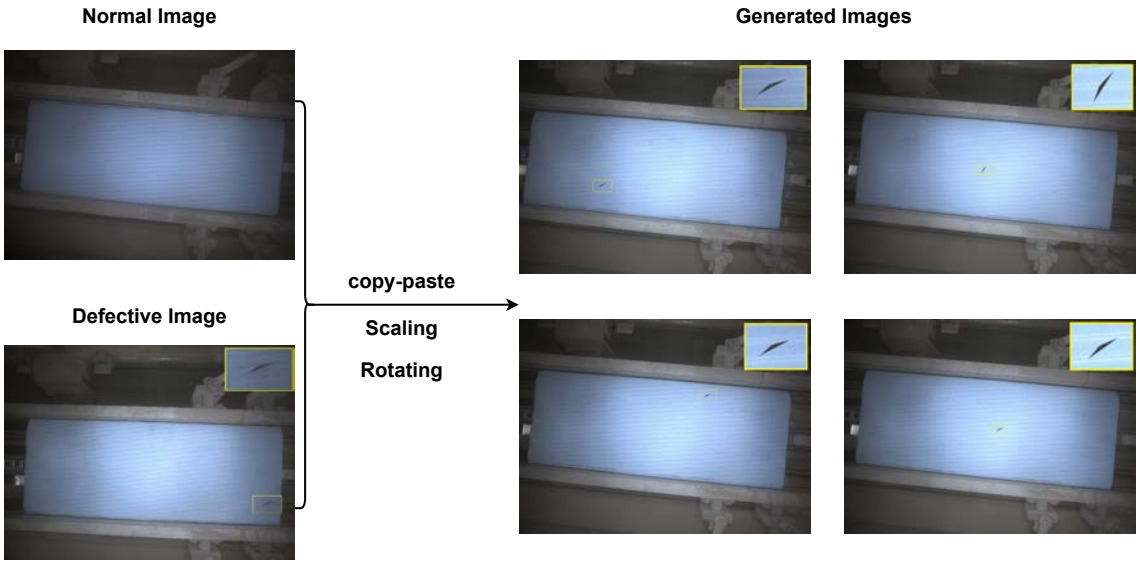


Figure 3: Copy-paste augmentation method.

2.3 METRICS

In every deep learning model, a suitable metric is required to impartially assess the model's performance and capabilities, as well as its limitations. In the context of classification tasks, metrics are commonly determined by: True positives (TP), False positives (FP), False negatives (FN) and True negatives (TN) [19]. The precision, recall and F-score metrics quantify the proportion of correctly classified samples relative to the total number of samples, whereas the Intersection over Union (IoU) compares the overlap between the predicted mask and the ground truth label.

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$Fscore = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{13}$$

$$IoU = \frac{TP}{TP + FN + FP} \tag{14}$$

Mean Average Precision (mAP) is a performance metric widely used in object detection tasks. It quantifies the precision and localization accuracy of an algorithm in detecting objects within an image and is computed by the mean of the average precision (AP) for all classes. AP is calculated by determining the area under the precision-recall curve for a given class. The

utilization of mAP as a metric enables the comprehensive evaluation of an algorithm's capability to detect and locate objects of different sizes and categories, yielding a single scalar value for overall performance assessment. The formula for computing mAP is given by Equation 15

$$mAP = \frac{1}{N} \sum_{i}^{N} AP_i, \tag{15}$$

where N is the number of classes, and AP_i is the AP for class i. Since Precision is dependent on IoU, mAP[0.5:0.95] is commonly used [19] to consider the average for a number of IoU thresholds.

3 EXPERIMENTAL RESULTS

For all the experiments we used the same environment, using GPU: NVIDIA GeForce GTX 1080 Ti 11GB, CPU: Intel i7-8700K, RAM: 64 GB, software environment: Python 3.9, Pytorch backend. During the training, for all the models the same process was followed which is to train each model until convergence. During evaluation, for each of the trained models, an image is provided as input and the model generates the coordinates of the bounding boxes, the predicted class and the confidence score for each detected instance as an output.

The experimental results are presented in Table 2, where we report both mAP and inference time for each model. We see that both YOLO architectures outperform the other approaches, while YOLOv8 achieves the best mAP of 92.20%, in comparison to the other networks. Secondly, we notice that YOLOv7 achieves slightly lower mAP. The YOLOv8 network has an average inference time of 0.007 seconds, equivalent to a detection speed of 142 frames per second. This performance level satisfies the real-time inspection requirements of production lines. For the rest of the approaches, YOLOv7 is approximately 2 times slower than YOLOv8, while the other are 8-10 times slower. The average precision (AP) reflects the YOLOv8's ability to accurately detect and localize the majority of the defect types, achieving an AP that exceeds 0.950 for these specific classes. Conversely, it is evident that YOLOv8 does not perform well for small-scale defect types, such as perforations and ruptures. For these particular defect types, YOLOv8 achieves its worst performance, with AP scores of 0.619 and 0.840, respectively.

The classification of battery separators as defective or not is a critical task for operators in the battery production line. In this regard, we utilize the outputs of the trained object detection models to classify an image containing a battery separator as defective if the object detection model outputs a detection of a defect with a confidence score exceeding a predefined threshold of 0.5. This threshold enables a flexible and customizable classification process, which can be adapted based on the desired level of risk tolerance. To evaluate the performance of the proposed methodology, we enhanced the test dataset with 1600 normal images to realistically represent the production line. We employed precision, recall, and f1-score as evaluation metrics to assess the classification performance of the model.

The findings presented in 3 indicate that the Mask R-CNN model demonstrates a slightly superior performance to the YOLOv8 architecture for the classification of battery separator

Table 2: Experimental results for object detection. For the total number of classes, mAP@[IoU=0.5:0.95] and inference time are calculated.

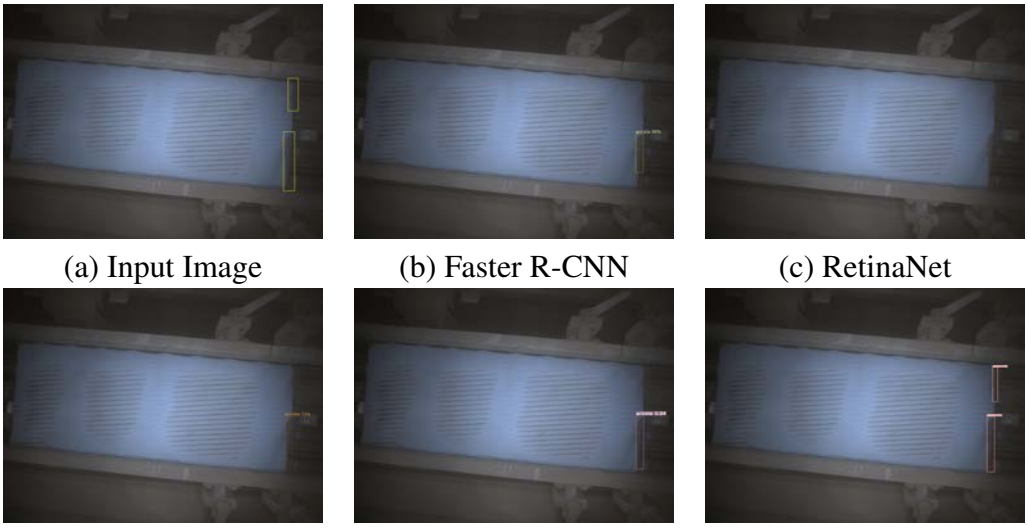
	Faster R-CNN	RetinaNet	Mask R-CNN	YOLOv7	YOLOv8
Defect type	Average Precision				
Wrinkling	0.760	0.694	0.782	0.884	**0.962**
Imprinted marking	0.960	0.990	0.970	0.984	**0.991**
Misaligned seam	0.744	0.598	0.845	0.943	**0.994**
Paint residue	0.936	0.925	0.948	0.986	**0.995**
Perforation	**0.740**	0.661	0.710	0.673	0.619
Lag	0.914	0.880	0.972	0.994	**0.995**
Rupture	0.649	0.709	0.698	0.714	**0.840**
Exposed plate	0.849	0.847	0.908	0.941	**0.976**
mAP	0.819	0.788	0.854	0.889	**0.921**
Inference time	0.067s	0.060s	0.070s	0.017s	**0.007s**

Table 3: Precision, recall and f-score on the evaluation of classifying a battery separator image as defective or normal.

Model	Precision	Recall	F-score
Faster R-CNN	0.887	0.990	0.930
RetinaNet	0.981	0.887	0.929
Mask R-CNN	0.912	0.993	0.944
YOLOv7	0.940	0.932	0.927
YOLOv8	0.956	0.942	0.940

images. Specifically, the Mask R-CNN model achieves an F-score of 0.944, whereas the YOLOv8 model achieves an F-score of 0.940. The other network architectures evaluated in the study exhibit a relatively lower performance, with F-scores around 0.930.

Figure 4 presents a qualitative comparison between the examined approaches, on a defective image from the production. The actual defect of the sample is the presence of two wrinklings, as indicated in Figure 4a. The results reveal that RetinaNet fails to detect any defects on the surface of the battery separator. Conversely, Faster R-CNN, Mask R-CNN and YOLOv7 models manage to identify only one of the two existing wrinklings. Lastly, YOLOv8 architecture accurately identifies the defects and produce the bounding boxes and the confidence scores for the predicted defect class. YOLOv8 generates a confidence score of 0.62 and 0.91, respectively for the two defects. Erroneous detections were observed mostly on the separator's periphery, coinciding with areas of suboptimal system illumination [20].



(a) Input Image (b) Faster R-CNN (c) RetinaNet

(d) Mask R-CNN (e) YOLOv7 (f) YOLOv8

Figure 4: Defect detection results for a defective image sample containing two wrinklings.

4 CONCLUSIONS

In our work, we used a machine vision system to capture images of defective battery separators. To construct a separator defect dataset, we applied copy-paste augmentation to generate eight distinct types of defects. We then evaluated the performance of five object detection algorithms on this dataset. The findings indicate that YOLOv8 outperformed the other algorithms, achieving a mean average precision (mAP) of 92.20% and an inference time of 0.007 seconds per image. We examined the models' classification performance on real production line conditions. Although Mask R-CNN outperformed YOLOv8 in terms of F-score by 0.42%, the latter exhibited a $10\times$ faster inference time and thus is more appropriate for deployment in the current production line.

Based on the results, we demonstrated that the proposed approach has the potential to be deployed in production, indicating to the operator the location and size of the defects accurately. In future work, we plan to investigate the potential of this solution in other production stages, where it is challenging to apply the same augmentation procedure due to the complexity of defects. Finally, we plan to enhance the lighting conditions in the inspection scene in order to alleviate the adverse impact of suboptimal illumination in the performance of the models.

ACKNOWLEDGMENTS

This work has been co-funded by the European Union and the General Secretariat for Research and Innovation, Ministry of Development & Investments, under Project SYSSOREUTIS/T2EDK-03479. This work was also supported by the European Commission through Project OPTIMAI funded by the European Union H2020 programme under Grant 958264. The opinions expressed in this article are those of the authors and do not necessarily reflect the views of the European Commission.

REFERENCES

[1] K. W. Beard, *Linden's handbook of batteries*. McGraw-Hill Education, 2019.

[2] S. S. N. Kassatly, "The lithium-ion battery industry for electric vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.

[3] S. Bounareli, I. Kleitsiotis, L. Leontaris, N. Dimitriou, A. Pilalitou, N. Valmantonis, E. Pachos, K. Votis, and D. Tzovaras, "An integrated system for automated 3d visualization and monitoring of vehicles," *The International Journal of Advanced Manufacturing Technology*, vol. 111, pp. 1797–1809, 2020.

[4] P. Arora and Z. Zhang, "Battery separators," *Chemical reviews*, vol. 104, no. 10, pp. 4419–4462, 2004.

[5] G. Pistoia, "Lithium-ion batteries: advances and applications," 2013.

[6] T. Kotsiopoulos, L. Leontaris, N. Dimitriou, D. Ioannidis, F. Oliveira, J. Sacramento, S. Amanatiadis, G. Karagiannis, K. Votis, D. Tzovaras *et al.*, "Deep multi-sensorial data analysis for production monitoring in hard metal industry," *The International Journal of Advanced Manufacturing Technology*, vol. 115, pp. 823–836, 2021.

[7] A. Evangelidis, N. Dimitriou, L. Leontaris, D. Ioannidis, G. Tinker, and D. Tzovaras, "A deep regression framework towards laboratory accuracy in the shop floor of microelectronics," *IEEE Transactions on Industrial Informatics*, 2022.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[11] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[12] 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.

[13] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[14] 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.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[16] 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 preprint arXiv:2207.02696*, 2022.

[17] G. Jocher, A. Chaurasia, and J. Qiu, "Yolo by ultralytics," 1 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[18] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2918–2928.

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[20] L. Leontaris, N. Dimitriou, D. Ioannidis, K. Votis, D. Tzovaras, and E. Papageorgiou, "An autonomous illumination system for vehicle documentation based on deep reinforcement learning," *IEEE Access*, vol. 9, pp. 75 336–75 348, 2021.