# Deep learning algorithms for automatic part inspection in automotive manufacturing

Jon Toledo[1*], Symone G. S. Alcalá[2†] and Alvaro J. Lopez-Lopez[1†]

[1]Institute for Research in Technology, ICAI School of Engineering, Comillas Pontifical University, C/ Alberto Aguilera 25, Madrid, 28015, Madrid, Spain.
[2]Faculty of Sciences and Technology, Federal University of Goiás, Estrada Municipal - Quadra e Área Lote 04, Aparecida de Goiânia, 74971-451, Goiás, Brazil.

*Corresponding author(s). E-mail(s): jontoledo2000@gmail.com;
Contributing authors: symone@ufg.br; allopez@comillas.edu;
[†]These authors contributed equally to this work.

**Abstract**

Automotive manufacturing industries work at the lowest error margin in the produced parts, and even a small defect can impact their production and success. Post-production quality inspection by humans is expensive and time-consuming. Therefore, industries need to propose automatic and cost-effective quality inspection systems. Deep learning (DL) algorithms have been successfully applied in computer vision systems for defect detection in manufacturing applications, improving accuracy and robustness by learning features from large image datasets. However, one concern in DL-based inspection systems is the class imbalance problem caused by the difficulty in obtaining sufficient defective image samples. This work presents a deep learning-based computer vision system for the detection of defects in battery lid holes. Three architectures are compared and evaluated: YOLOv11, ResNet18, and EfficientNet-B0. After data set preparation and enhancement, the proposed models achieved accuracies greater than 97% and high recall values, demonstrating their feasibility for integration into real-time industrial quality control processes.

**Keywords:** Computer vision, Deep learning, Automated inspection, Defect detection, YOLO, Automotive industry

## 1 Introduction

The manufacturing industries are placed in a competitive environment where agility, quality, and efficiency impact their success [1]. They work at the lowest error margin in the produced components, and even a small defect or variance can corrupt the whole production [2]. The post-production quality inspection by humans can prevent defective parts from arriving to the customers. However, many defects are hidden from human eyes, and human inspection is expensive and time-consuming. Thus, industries need to replace traditional inspection approaches with automatic and cost-effective systems [3].

Computer vision system enables computers to collect and analyze visual data from images and videos using artificial intelligence (AI) algorithms, mimicking human vision capabilities [4]. It allows automated product quality inspection, reducing human errors and resources, increasing efficiency, and ensuring quality. The core of this system is the AI model employed to learn from data and perform predictions. Recently, deep learning (DL) models have emerged as a robust AI approach, outperforming traditional AI models. DL approaches can learn complex image representations and identify defects by employing artificial neural networks and learning algorithms trained on large image datasets [5].

Several DL algorithms have been successfully employed for defect detection in various manufacturing applications, including textile [6], additive manufacturing [7], steel [8], and automotive [9–14]. Defect detection using DL can enhance accuracy and robustness by learning features from large image datasets. In literature, many DL architectures and models have emerged for defect inspections, such as YOLO [9, 10], SqueezeNet [11], EfficientNet [12], ResNet [13], and LeNet [14]. For example, in [13], a ResNet model achieved 99% of accuracy in detecting the presence or absence of screws in an automotive part. Despite DL-based inspection systems outperforming traditional AI models, one concern is the class imbalance problem caused by the difficulty of obtaining sufficient defective samples [15]. To address this problem, data augmentation techniques can be employed [16, 17].

In this work, we propose a deep learning-based computer vision system for the detection of defects in battery lid holes. The system was conceived to automate the entire inspection workflow, from image acquisition to model training and evaluation. Data were collected using six strategically positioned cameras along the production line, ensuring complete coverage of each component. A proprietary automatic segmentation algorithm was then applied to detect and crop each hole individually, generating a homogeneous dataset for weld defect detection. Since the dataset was naturally imbalanced, with most holes being defect-free, data augmentation techniques were applied to increase variability and robustness. After labeling the samples for both classification and detection tasks, the dataset was divided into training, validation, and test sets, and several deep learning models were trained, including YOLOv11 for detection and ResNet18 and EfficientNet-B0 for classification. The trained models achieved accuracies above 97% and high recall values, demonstrating their feasibility for integration into real-time industrial quality control.

The specific goal of this work is to develop an automated system capable of accurately detecting weld defects in battery lid holes. The system must reliably identify and discard defective parts without human intervention, while remaining robust to variations in illumination, defect types, and surface differences between components. Furthermore, it must operate at a speed compatible with real industrial production lines, ensuring that inspection is performed without slowing down the workflow. This problem highlights the need for a tailored computer vision system capable of combining effective preprocessing, annotation, and deep learning techniques to classify parts as either compliant or defective under realistic production conditions.

This work is organized as follows. Section 2 presents the main related works. Section 3 describes the employed materials and methods. Section 4 presents the experimental results and discussion. Section 5 concludes the paper.

## 2 Related works

With the advances in DL algorithms, researchers are increasingly exploring DL capabilities to learn and analyze data images in automotive manufacturing applications automatically. Wang et al. [9] propose an adhesive defect detection approach for automotive applications based on YOLOv8. It incorporates an attention mechanism, called skip squeeze and excitation, into YOLOv8 and replaces the original IoU loss function with the WIoU loss function to improve the detection performance for small adhesive defects. The proposed approach achieved higher accuracy and speed than the standard YOLOv8.

In the automotive industry, a concern is the quality of the welding spots in the produced parts. Since manual inspection is usually inefficient and error-prone, in literature, several computer vision works for automatic spot welding quality can be found in [10–12, 18]. For example, Dai et al. [10] present a modified YOLOv3 model for small spot welds to detect their positions and qualities. In addition, data augmentation techniques (flipping, color jittering, shift, and crop) are proposed to increase the dataset. The results demonstrated that Faster R-CNN slightly outperforms the modified YOLOv3 in detection performance but at a higher time cost. Other studies are devoted to inspecting welding defects in power batteries [11, 12] since a safety vent welded on the battery can prevent unpredictable explosions.

Hachem et al. [13] propose an automatic part quality control in an automotive company to replace a visual inspection method performed by an operator. The proposed system, composed of a ResNet-50 model, detects the presence or absence of screws in a component with 99% of accuracy. Muresan et al. [14] present a convolutional neural network based on the LeNet-5 architecture to identify well-placed, badly placed, and missing bushings in an automotive component. The proposed model achieved between 98% and 99% accuracy under different scenarios that is different light conditions and camera's positions. Thus, taking into account the emerging success of DL in defect detection, this paper proposes a DL algorithm to inspect an automotive part.

## 3 Materials and methods

This section presents the problem description and the main steps of the proposed methodology.

### 3.1 Problem description

This work addresses a real-world industrial case of an international company specializing in designing, developing, and manufacturing metal automotive components. The company provides parts to several automobile manufacturers in the world. In particular, it aims to improve its computer vision system that automatically detects failures or missing welded washers on small holes in battery covers and, thus, discards the defective ones. The main objective is to avoid defective covers arriving to the customers.

Currently, the company operates with a computer vision system for inspecting battery covers that employs a grayscale value pyramid algorithm, developed by an external software provider. For each battery cover, the actual system receives six images from six cameras (Figure 1); then, the algorithm inspects each hole and classifies it as OK (i.e., without defect) or NOK (i.e., with defect). Figure 2 shows examples of OK and NOK holes. If a defect is detected by the algorithm, an operator must inspect the cover to confirm the failure. Thus, if the algorithm misclassifies a cover as NOK, an operator spends time reviewing good covers; nevertheless, if it misclassifies a defective cover as OK, a customer may receive a faulty part. Thus, the company wants to develop a new algorithm to reduce cases of false NOK without losing the system's ability to detect defective covers.

### 3.2 Data acquisition

The images used were collected from an assembly line of the company in Spain. They correspond to a cover battery from one model car. To acquire images, a cover battery is held perpendicular to the ground; then, an arm robot photographs it using six cameras. Each camera captures an image of an area of the cover, and each image contains between 5-8 visible holes (see Figure 1). The cameras overlap some areas, so repeated holes can be found within these six images. In addition, the captured images may have different lighting conditions, mainly because the parts' material composition may vary by batch and reflect different lighting settings.

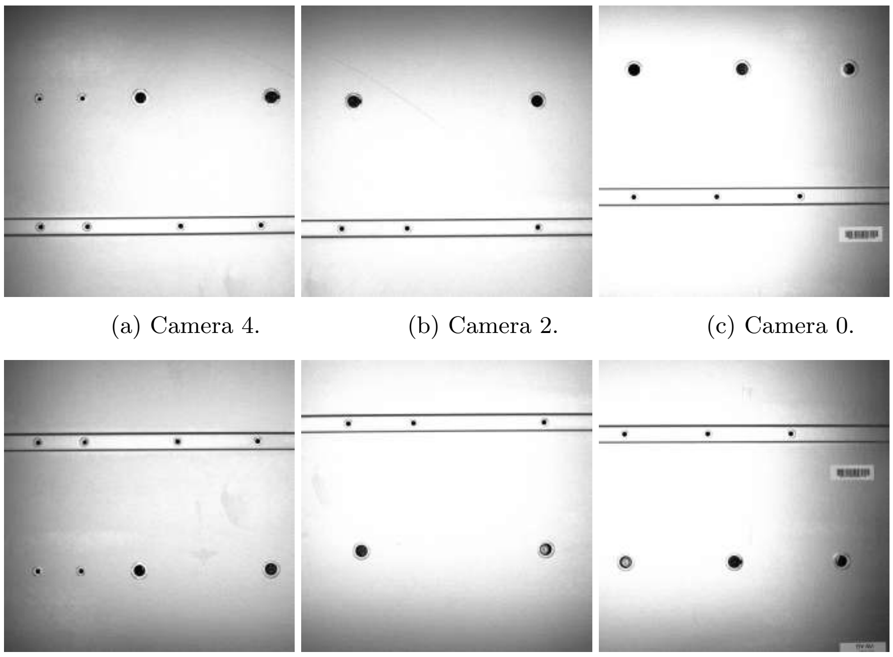
The data acquisition process has generated an initial set of 378 images (2856 $\times$ 2848 resolution) of 63 covers, being that 54 covers are non-defective and 9 are defective (i.e., a cover with at least one defective hole). As there are fewer samples of defective holes, data augmentation is employed to increase the data samples.

### 3.3 Data preprocessing

In this step, images are transformed so that they can be effectively processed by the learning model. In particular, this step converts the initial set of images with multiple holes to an augmented set of cropped images with one hole. The following substeps were performed to automatically locate, crop, and label each hole, as well as create new images.

#### 3.3.1 Cropping

To locate and crop a hole from an image, the initial set of images (in RGB mode) was converted into grayscale images using the OpenCV API. To do this, the pixels of the grayscale images can assume values between 0 (black) and 255 (white).



(a) Camera 4. (b) Camera 2. (c) Camera 0.

(d) Camera 5. (e) Camera 3. (f) Camera 1.

**Fig. 1**: Battery cover images captured by the system's cameras.

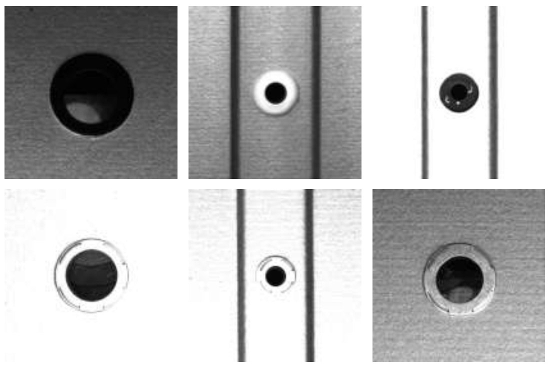


**Fig. 2**: Examples of defective (top) and non-defective (bottom) holes.

Then, binary images were obtained using a simple threshold. This method assigns 1 to pixels equal to or greater than a threshold and 0 to pixels less than a threshold. After checking different random threshold values, the best value was found to be 60 for the given images. Using this binarization procedure, the darkest image areas (such as the holes) become black, and the lighter areas become white, achieving maximum contrast.

After binarization, contour detection techniques were applied to locate the potential holes. Contours with an area smaller than 500 pixels were discarded, while the remaining ones were approximated by rectangular bounding boxes. To reduce false positives, only contours with a nearly square aspect ratio and within the expected spatial region of the holes were considered. For each valid contour, the centroid was computed as the reference point of the hole. Close centroids were merged using a Euclidean distance criterion (threshold = 300 pixels) to avoid duplicate detections. Finally, the centroids were ordered according to their position (left–right and top–bottom) to match the layout of the holes as seen by the cameras. Around each centroid, a region of $400 \times 400$ pixels was cropped, resulting in 1,080 (OK) and 125 (NOK) images corresponding to non-defective and defective holes, respectively.

### 3.3.2 Data augmentation

The previous step revealed fewer images of defective (NOK) holes than non-defective (OK) holes. Thus, the database has a class imbalance problem. This occurs when there is a disproportionate ratio of instances. In class imbalance databases, a learning model tends to be more biased toward the majority class, leading to inaccurate classification of the minority class and poor classification accuracy [19].

This problem can be overcome by data augmentation techniques that synthetically create new data samples to address imbalance classes [16]. In literature, several image data augmentation techniques are available, such as classical approaches (e.g., flipping, rotation, translation, filters, and noise) and deep learning approaches (e.g., generative adversarial networks and neural style transfer) [17].

Among them, classical image data augmentation techniques were applied to increase the number of the minority class (NOK) in this work. Using rotation and filter (brightness and contrast) techniques, 623 new NOK images were generated. After these procedures, 1,832 image data points were obtained, being that 1,084 images are from the OK class and 748 images are from the NOK class.

### 3.3.3 Labeling

After, the images were manually labeled using the LabelImg tool [20]. It is a popular image annotation tool for creating labeled datasets so that DL models can be easily trained. LabelImg allows labeling by drawing bounding boxes and, thus, generating annotations in a text document with the number of the annotated class followed by the coordinates of the bounding box. In this case, the 1,694 images were annotated as OK or NOK, and then the result was saved into a .txt file.

### 3.3.4 Data splitting

The dataset was divided into training, validation, and test subsets to evaluate the performance of the proposed models. A total of 1,283 images were used for training (759 OK and 524 NOK), while 325 images (325 OK and 224 NOK) were reserved for validation. For the test set, the original number of defective covers (9 NOK) was augmented to balance the classes, resulting in a total of 341 test hole images. This splitting strategy ensured that model training and hyperparameter tuning were performed on independent sets, while the final evaluation relied on unseen data.

## 3.4 Model description and setup

The YOLOv11 model, one of the most recent architectures in the YOLO family, was employed for object detection. The training process was initialized from the pre-trained nano version (yolo11n.pt) on the COCO dataset, allowing the transfer of optimized weights for general detection tasks and their adaptation to the specific problem through fine-tuning. The input image size was fixed at 640 $\times$ 640 pixels, applying padding when necessary to preserve the original aspect ratio. Training was performed for 100 epochs, with performance evaluated on the validation set at the end of each epoch.

The default YOLOv11 configuration was adopted for the AdamW optimizer, with an initial learning rate of $1 \times 10^{-3}$ and a cosine learning rate scheduler, which progressively decreases the learning rate as training advances. To improve generalization and mitigate overfitting, data augmentation techniques integrated in the YOLO framework were employed, including slight rotations, brightness and contrast variations, horizontal flips, and random scaling.

For the binary classification stage of each hole, two architectures were employed: ResNet18 and EfficientNet-B0. Both models were initialized with weights pre-trained on ImageNet and subsequently fine-tuned through transfer learning for the specific task. In both cases, the final fully connected layer was replaced by a single logit output (dimension = 1), enabling direct optimization of a binary loss. The loss function employed was BCEWithLogitsLoss, which integrates the sigmoid activation into the loss term itself, providing improved numerical stability compared to applying a sigmoid followed by a conventional Binary Cross-Entropy.

The model inputs were resized to 224 $\times$ 224 pixels and normalized using the standard ImageNet statistics, ensuring compatibility with the initial convolutional layers of both architectures and a homogeneous data flow. The dataset was organized using the ImageFolder structure (train/ and val/ subfolders per class), which facilitated data loading through PyTorch DataLoaders and random shuffling within mini-batches.

The optimization procedure was identical for both models to ensure a fair comparison. Specifically, the Adam optimizer was employed with an initial learning rate of $1 \times 10^{-3}$, a batch size of 32 (adjusted according to GPU memory), and a training schedule between 10 and 50 epochs depending on validation convergence. No layers were explicitly frozen; therefore, full fine-tuning was performed, allowing both the early convolutional blocks and the newly added classification layer to adapt to the visual characteristics of the holes and their defects. During validation, model outputs were passed through a sigmoid and thresholded at 0.5 to obtain the predicted label; this threshold could be adjusted later depending on the desired precision–recall trade-off for real industrial deployment.

At each epoch, the training loss was monitored, and in validation, classification metrics were computed, including precision, recall, F1-score, and accuracy. After training, the best state of each model was saved for subsequent evaluation in Section 4.

## 3.5 Model evaluation

Accuracy, precision, recall, F1-score, and inference time were employed to evaluate the models. Accuracy measures the model performance by calculating the proportion between predictions made by a model and the actual values. Thus, it indicates the model's ability to assign labels to the classes correctly. Mathematically, *accuracy* is defined as [7]:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}, \qquad (1)$$

where $TP$ is the number of true positive samples, $TN$ is the number of true negative samples, $FP$ is the number of false positive samples, and $FN$ is the number of false negative samples. In this work, positive and negative classes are represented by OK and NOK, respectively. Accuracy values are between 0 and 1, where the closer to 1, the better the classification.

While accuracy provides a global measure, it may not always reflect class-specific behavior, particularly under class imbalance. For this reason, precision, recall, and the F1-score were also used. Precision is the ratio of correctly predicted positive cases with respect to all predicted positive cases:

$$precision = \frac{TP}{TP + FP}. \qquad (2)$$

Recall measures the ability of the model to identify all relevant positive cases, i.e., the ratio of correctly predicted positives with respect to all actual positives:

$$recall = \frac{TP}{TP + FN}. \qquad (3)$$

The F1-score combines both precision and recall into a single metric, expressed as the harmonic mean of the two:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \qquad (4)$$

Finally, inference time was considered as a practical metric of efficiency, measuring the average time required by each model to process one part. This aspect is particularly relevant in industrial inspection, where thousands of parts must be evaluated daily, and computational latency directly affects production throughput.

## 4 Experimental results and discussion

The experiments were conducted on a PC equipped with an AMD Ryzen 7 5800H processor with Radeon Graphics (3.20 GHz), 16 GB of RAM, and an NVIDIA GeForce RTX 3060 Laptop GPU. The operating system employed was Windows 11 Professional 64-bit. The models were developed using PyTorch version 2.5.1 with CUDA 11.8, and the programming language was Python 3.12.7.

Upon moving to the test evaluation, the results provide a deeper understanding of the models' behavior and, in particular, the sources of prediction errors. As shown in Table 1, most errors correspond to hole samples previously marked by the collaborating company in order to reinforce inspection techniques. In general, the models correctly learned to identify the majority of introduced defects; however, certain specific cases still challenged their generalization capacity.

Two recurrent error patterns were identified. First, some holes contained stickers placed on their surface, which often led YOLO to generate duplicate detections and occasionally caused misclassifications. Second, several OK holes were

**Table 1**: Average performance metrics on the test set for the evaluated models.

| Model | Accuracy | Precision | Recall | F1-score | Inference time (s) |
|---|---|---|---|---|---|
| EfficientNet-B0 | 0.924 | 0.889 | 0.786 | 0.832 | 1.601 |
| ResNet18 | 0.924 | 0.889 | 0.786 | 0.832 | 1.507 |
| YOLOv11 | 0.988 | 0.889 | 0.870 | 0.879 | 3.350 |

painted in black, masking their visual appearance and leading multiple models to mispredict them as NOK. These situations are noteworthy because, although stickers and painted holes were also present in the training set, the models did not always generalize correctly in the test set, highlighting the limitations of the architectures when facing certain visual variations.

Another relevant aspect is inference time. Contrary to initial expectations, the classification-based models (EfficientNet-B0 and ResNet18) showed very similar and relatively low execution times, around 1.5–1.6 seconds per part. In contrast, YOLOv11, despite being conceived for real-time detection, exhibited a significantly higher latency, reaching 3.35 seconds, i.e., approximately twice the time of the classification approaches. Although this difference may appear small in absolute terms, it becomes highly relevant in an industrial production scenario, where thousands of parts must be inspected daily and cumulative delays directly affect process efficiency.

Overall, the discussion of test results provides clearer insights than those obtained during validation. As summarized in Table 1, ResNet18 and EfficientNet-B0 achieved nearly identical metrics, with an accuracy of about 92% and an F1-score of 0.83, whereas YOLOv11 reached a superior performance, with an accuracy close to 99% and an F1-score of 0.88. This confirms YOLOv11 as the architecture with the highest generalization ability under adverse test conditions. Nevertheless, its superior predictive quality comes at the cost of nearly doubled inference time compared to the classification models. Therefore, the final choice between detectors and classifiers depends on the trade-off between accuracy and processing speed required in the industrial deployment scenario.

## 5 Conclusion

This work presented a deep learning-based computer vision system for the automatic detection and classification of defects in battery lid holes, aiming to reduce the cost and subjectivity of manual quality inspection in the automotive industry. The methodology integrated both detection (YOLOv11) and classification (ResNet18 and EfficientNet-B0) approaches, trained and validated on a dataset of cropped hole images prepared through image preprocessing and augmentation.

The experimental results confirmed the effectiveness of deep learning in this domain. While the classification models achieved solid performance, with accuracies of around 92% and F1-scores of 0.83, YOLOv11 clearly outperformed them, reaching nearly 99% accuracy and an F1-score of 0.88. These findings demonstrate that YOLOv11 exhibits higher generalization ability to adverse visual conditions, such as stickers or black paint, which posed difficulties for the classification-based models. Nevertheless, this superior accuracy came at the expense of longer inference times (3.35 seconds per part, approximately double that of ResNet18 and EfficientNet-B0).

From an industrial perspective, this trade-off between accuracy and speed is critical. In production environments where thousands of parts must be inspected daily, the selection of the most appropriate model depends on the operational priorities: maximizing detection reliability or minimizing latency. In either case, the proposed system demonstrates the feasibility of integrating deep learning into real-time quality control workflows, contributing to enhanced efficiency, reduced human dependency, and improved overall reliability of inspection processes.

Future work will focus on augmenting and diversifying the dataset, exploring advanced augmentation strategies to further mitigate class imbalance, and optimizing inference through lighter architectures or hardware acceleration. Such improvements are expected to reinforce the robustness and scalability of the proposed solution, paving the way toward its deployment in large-scale industrial scenarios.

## References

[1] Bhatia, M.S., Kumar, S.: Critical success factors of Industry 4.0 in automotive manufacturing industry. IEEE Trans. Eng. Manag. **69**, 2439–2453 (2022) https://doi.org/10.1109/TEM.2020.3017004

[2] Ahmad, H.M., Rahimi, A.: Deep learning methods for object detection in smart manufacturing: A survey. J. Manuf. Syst. **64**, 181–196 (2022) https://doi.org/10.1016/j.jmsy.2022.06.011

[3] Ameri, R., Hsu, C.-C., Band, S.S.: A systematic review of deep learning approaches for surface defect detection in industrial applications. Eng. Appl. Artif. Intel. **130**, 107717 (2024) https://doi.org/10.1016/j.engappai.2023.107717

[4] Gill, R., Srivastava, D., Hooda, S., Singla, C., Chaudhary, R.: Unleashing sustainable efficiency: The integration of computer vision into Industry 4.0. Eng. Manag. J., 1–19 (2024) https://doi.org/10.1080/10429247.2024.2383518

[5] Jia, Z., Wang, M., Zhao, S.: A review of deep learning-based approaches for defect detection in smart manufacturing. J. Opt. **53**, 1345–1351 (2024) https://doi.org/10.1007/s12596-023-01340-5

[6] Revathy, G.., Kalaivani, R..: Fabric defect detection and classification via deep learning-based improved Mask RCNN. Signal Image Video P. **18**, 2183–2193 (2024) https://doi.org/10.1007/s11760-023-02884-6

[7] Silva, L.M., Alcalá, S.G.S., A. Barbosa, T.M.G., Araújo, R.: Object and defect detection in additive manufacturing using deep learning algorithms. Prod. Eng. Res. Devel. **18**, 889–902 (2024) https://doi.org/10.1007/s11740-024-01278-y

[8] Zhang, W., Huang, T., Xu, J., Yu, Q., He, Y., Lai, S., Xu, Y.: DF-YOLOv7: Steel surface defect detection based on focal module and deformable convolution. Signal Image Video P. **19**, 97 (2025) https://doi.org/10.1007/s11760-024-03679-z

[9] Wang, C., Sun, Q., Dong, X., Chen, J.: Automotive adhesive defect detection based on improved YOLOv8. Signal Image Video P. **18**, 2583–2595 (2024) https://doi.org/10.1007/s11760-023-02932-1

[10] Dai, W., Li, D., Tang, D., Jiang, Q., Wang, D., Wang, H., Peng, Y.: Deep learning assisted vision inspection of resistance spot welds. J. Manuf. Process. **62**, 262–274 (2021) https://doi.org/10.1016/j.jmapro.2020.12.015

[11] Yang, Y., Yang, R., Pan, L., Ma, J., Zhu, Y., Diao, T., Zhang, L.: A lightweight deep learning algorithm for inspection of laser welding defects on safety vent of power battery. Comput. Ind. **123**, 103306 (2020) https://doi.org/10.1016/j.compind.2020.103306

[12] Din, N.U., Zhang, L., Zhou, Y., Chen, Z., Yao, Y., Yang, Z., Yang, Y.: Laser welding defects detection in lithium-ion battery poles. Eng. Sci. Technol. Int. J. **46**, 101495 (2023) https://doi.org/10.1016/j.jestch.2023.101495

[13] Hachem, C.E., Perrot, G., Painvin, L., Couturier, R.: Automation of quality control in the automotive industry using deep learning algorithms. In: Int. Conf. on Computer, Control and Robotics (ICCCR), pp. 123–127 (2021). https://doi.org/10.1109/ICCCR49711.2021.9349273

[14] Muresan, M.P., Cireap, D.G., Giosan, I.: Automatic vision inspection solution for the manufacturing process of automotive components through plastic injection molding. In: IEEE 16th Int. Conf. on Intelligent Computer Communication and Processing (ICCP), pp. 423–430 (2020). https://doi.org/10.1109/ICCP51029.2020.9266249

[15] Lee, S., Luo, C., Lee, S., Jung, H.: Two-stream network one-class classification model for defect inspections. Sensors **23**(12) (2023) https://doi.org/10.3390/s23125768

[16] Hwang, S., Lee, J.: Classification of battery laser welding defects via enhanced image pre-processing methods and explainable artificial intelligence-based verification. Eng. Appl. Artif. Intel. **133**, 108311 (2024) https://doi.org/10.1016/j.engappai.2024.108311

[17] Khalifa, N.E., Loey, M., Mirjalili, S.: A comprehensive survey of recent trends in deep learning for digital images augmentation. Artif. Intell. Rev. **55**, 2351–2377 (2022) https://doi.org/10.1007/s10462-021-10066-4

[18] Yang, B., Peng, Q., Zhang, Z., Zhang, Y., Li, Y., Xi, Z.: An offset-transformer hierarchical model for point cloud-based resistance spot welding quality classification. Comput. Ind. **161**, 104134 (2024) https://doi.org/10.1016/j.compind.2024.104134

[19] Khan, A.A., Chaudhari, O., Chandra, R.: A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation. Expert Syst. Appl. **244**, 122778 (2024) https://doi.org/10.1016/j.eswa.2023.122778

[20] LabelImg: GitHub repository - LabelImg. https://github.com/HumanSignal/labelImg (2018)