International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
OPEN ACCESS
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

Automatic Corrosion Detection of Metal Surfaces by using Image Processing Technique

Shanthi M[1],Doranala Hemanth[2], Chinmay Gowda M[3], Gopal Reddy[4], Arun Kumar B[5]
[1]HOD and Professor, Dept. of ISE, EPCET, Bangalore
[2,3,4,5]UG Student, Dept. of ISE, EPCET, Bangalore

ARTICLEINFO

Article History:
Published : 30 May 2025

Publication Issue :
Volume 12, Issue 15
May-June-2025
Page Number :
01-07

ABSTRACT

In order to detect six common types of steel surface defects—namely scratches, crazing, rolled-in scale , patches, inclusion, and pitted surface—in real time and with high accuracy, the project "Automatic corrosion detection of metal surfaces using image processing techniques" uses YOLOv8. Flask provides the driving framework for the Python operations performed by the system, and HTML, CSS, and JavaScript power a responsive user interface. The system recognizes and classifies flaws on the surface of steel using the object detection capabilities of YOLOv8, which is useful for quality assurance and defect management during industrial operations. The interrelation of Flask enables users to operate on the system so as to upload photos and see the outcome of identifying flaws, thus making user experience smooth. This method is used to guarantee that products with no flaws reach the market in an effort to increase manufacturing quality and reduce waste.

Keywords: Steel defect detection, YOLOv8, Flask Framework, real-time classification, surface quality control.

INTRODUCTION

Product quality assurance is paramount in the steel manufacturing industry, as defects in the 'skin' of the steel can have tremendous effects on the usability, safety, and presentation of the end product. Traditional inspection methods often rely on manual checks that are time-consuming and prone to human error, resulting in inconsistent quality and elevated production costs. With advancements in deep learning and computer vision technologies, development gives a unique opportunity to gain speed and accuracy while automating the process of flaw identification.

A reliable, efficient, and automated solution provided by real-time surface defect detection using models such as YOLOv8 addresses these industrial problems. This research aims to bridge the gap between conventional practices and modern technology, enabling industries to achieve improved quality control standards. Because they impact the final product's quality, longevity, and aesthetic

appeal, steel surface flaws are a big concern in the manufacturing sector. The structural integrity of steel, which is frequently utilized in essential applications, is compromised by common flaws such as crazing, inclusion, patches, pitted surfaces, rolled-in scale, and scratches.

Traditional inspection methods are based on manual checks that are time-consuming and error-prone. Yet with the latest trends in deep learning, new avenues for improvement and automation of the defect detection process have emerged. To specifically identify and classify various steel surface flaws, a modern object identification model, YOLOv8, is investigated in this study. The system offers real-time analysis and feedback, supported by a Flask-based backend integrated with a user-friendly frontend. This approach is designed to minimize environmental impact caused by waste and rework while also enhancing production quality and lowering operational costs by modernizing steel quality control processes.

CONTRIBUTIONS

The system uses the YOLOv8 model to identify and categorize surface flaws on steel using deep learning and sophisticated image processing techniques. Six forms of fault crazing are quickly detectable by it: inclusion, patches, pitting surfaces, rolled-in scales, and scratches. It prevents human mistakes and significantly improves the operational efficacy of industrial settings by avoiding manual inspections. CSS, in combination with JS and HTML, is used to create interactive content and give space for the loading of photographs and verifying the outcome of the detection of defects, while Flask is configured to establish an effective relationship between the model and user interface. Owing to the model's effectiveness and its adaptability across various surface conditions, data preprocessing steps such as image scaling, normalization, and augmentation have been applied. The proposed solution to this problem in the paper is scalable and cost-effective; hence, implementation of the automated defect detection and quality control to the existing industrial process becomes relatively easy. Furthermore, it lays the groundwork for future improvements, including the variety of the new problem types, the IoT-based monitoring that would be able to send real-time writes, and humans as the tool of predictive maintenance. The addition of all these benefits will be aimed at reducing the cost of operation, improving the process of manufacturing procedures, and maintaining the constant quality of a product in an industrial environment.

METHODOLOGY

The proposed system adopts the latest deep learning model, YOLOv8, used to detect 6 major types of defects on a steel surface at a high rate and in real time. This system provides a seamless experience for a user since it adopts a responsive frontend that is based on HTML, CSS, and JS, and the backend is developed in accordance with the Python language and using the Flask framework. This method ensures a better fault identification process, assists the process itself, and nullifies the human verification requirement completely. An effective and cheap solution that supports the requirements of a large industrial environment but has a positive effect on the quality of the product and the operating efficacy of an architectural composition of the system does not make it complex to integrate with the existing production lines. (Figure 1).

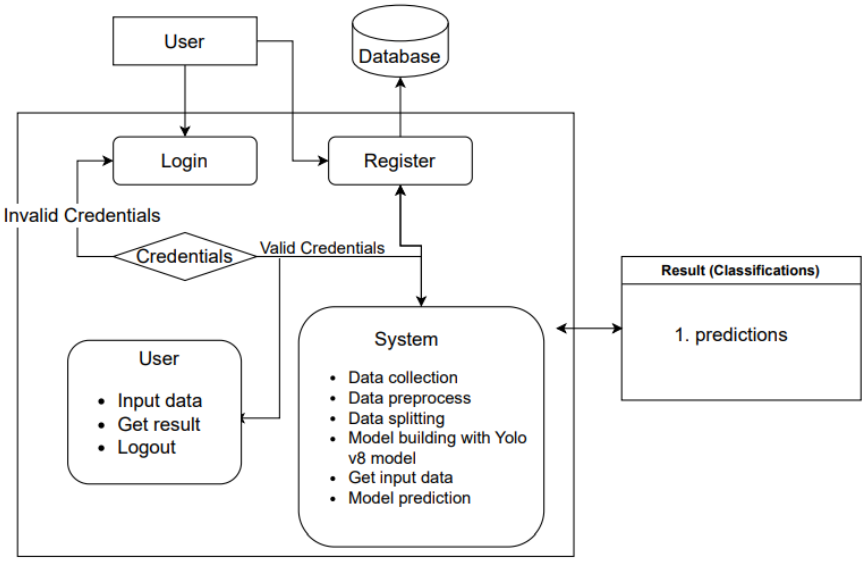


FIGURE 1: System Architecture

Five essential components make up the system design, which also offers completeness and scalability. The first module, known as the Input Module, is in charge of preprocessing and combining images. Steel surface photos are shot for more accurate and consistents, and then the photos are computed in an order of data cleaning, resizing, normalization, and augmentation. The second module, the Defect Detection Module, is configured to use the YOLOv8 model to determine pre-processed photos and six individual types of defects, including crazing, inclusion, patches, pitted surfaces, rolled-in scales, and scratches. This module obtains real-time detection and classification by minimizing human inspection.

For the third module, the Backend Processing Module, Flask is used during implementation to ensure smooth flow of communication between the user interface and the YOLOv8 model. It takes care of data processing, receives input images, feeds them through the model, and supports efficient detection output. The User Interface Module, the fourth module, has been made using JavaScript, HTML, and CSS. The module allows users the possibility to upload photos and visualize the defect detection results in an interactive platform with class and fault positioning given in an easy-to-understand manner.

The Future Enhancement Module outlines potential advancements for the system. These include integrating IoT technologies for real-time monitoring and alerts, incorporating AI-driven predictive maintenance to forecast surface degradation, and expanding the detection capabilities to cover additional types of surface defects. The system is designed to be modular in nature so that it should have been made resilient, scalable, and flexible for future industrializations.

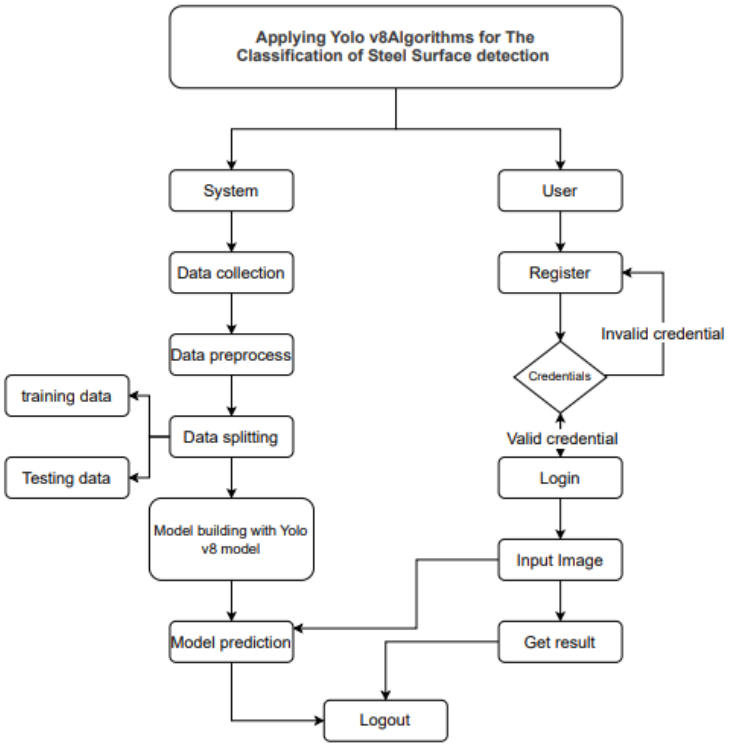


FIGURE 2: Block Diagram

Five essential components make up the system design, which also offers completeness and scalability. The first module, known as the Input Module, is in charge of preprocessing and combining images. Steel surface photos are shot for more accurate and consistents, and then the photos are computed in an order of data cleaning, resizing, normalization, and augmentation. The second module, the Defect Detection Module, is configured to use the YOLOv8 model to determine pre-processed photos and six individual types of defects, including crazing, inclusion, patches, pitted surfaces, rolled-in scales, and scratches. This module obtains real-time detection and classification by minimizing human inspection.

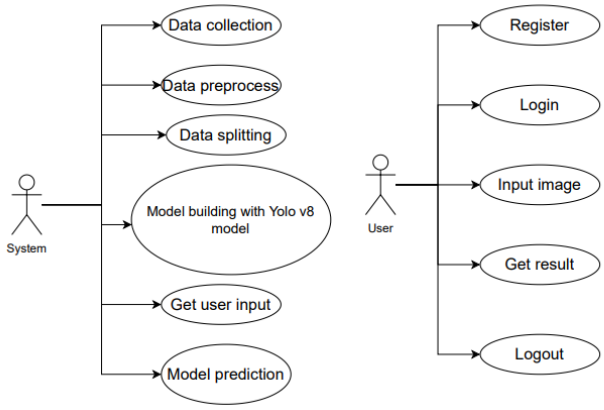
For the third module, the Backend Processing Module, Flask is used during implementation to ensure smooth flow of communication between the user interface and the YOLOv8 model. It takes care of data processing, receives input images, feeds them through the model, and supports efficient detection output. The User Interface Module, the fourth module, has been made using JavaScript, HTML, and CSS. The module allows users the possibility to upload photos and visualize the defect detection results in an interactive platform with class and fault positioning given in an easy-to-understand manner.

The Future Enhancement Module outlines potential advancements for the system. These include integrating IoT technologies for real-time monitoring and alerts, incorporating AI-driven predictive maintenance to forecast surface degradation, and expanding the detection capabilities to cover additional types of surface defects. The system is designed to be modular in nature so that it should have been made resilient, scalable, and flexible for future industrializations.

Use Case

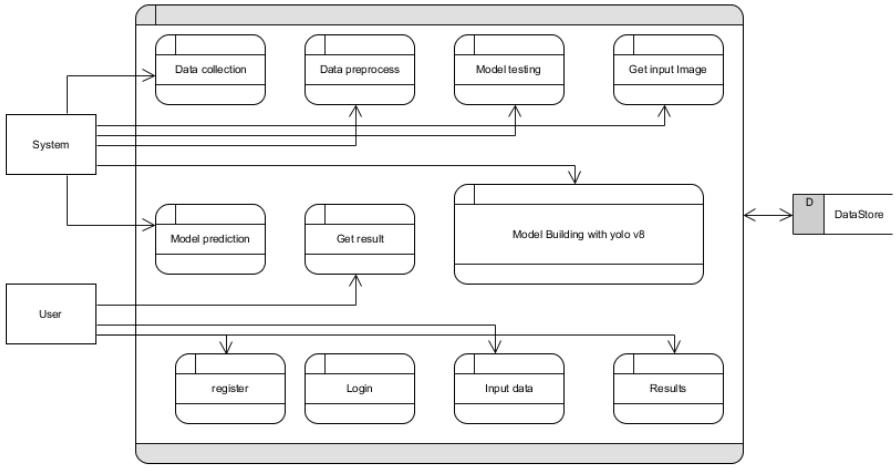
There is a kind of diagram of behavior emerging from use-case analysis known as a use-case diagram by Unified Modeling Language (UML). By drawing the relation of actors and the system's use cases, it serves as a graphical demonstration of how the system operates. The primary purpose of a use case diagram is to visually summarize the system's functionality by illustrating the actions performed by

various actors and highlighting the relationships or dependencies among the different use cases. This picture is an efficient method of learning and analyzing systems behavior and interaction because the roles and responsibilities of characters in the system are easily understandable.



DFD Diagram

There is a kind of diagram of behavior emerging from use-case analysis known as a use-case diagram by Unified Modeling Language (UML). By drawing the relation of actors and the system's use cases, it serves as a graphical demonstration of how the system operates. The primary purpose of a use case diagram is to visually summarize the system's functionality by illustrating the actions performed by various actors and highlighting the relationships or dependencies among the different use cases. This picture is an efficient method of learning and analyzing systems behavior and interaction because the roles and responsibilities of characters in the system are easily understandable.



RESULTS

Another important feature is the ability to upload data sets to the system, which is essential for processing relevant information. Typically, these data sets consist of sample data or historical records that the algorithm uses to make predictions. When users upload, they can verify the data set to ensure that the data they provided is presented correctly, which guarantees that sharing the data is open to people. While making predictions or results, the users should input specific values or parameters that would correlate with the variables or features at hand within the dataset.

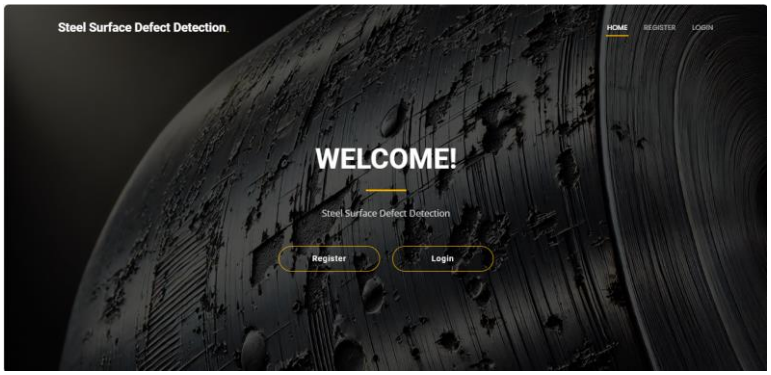
The system's functionality is structured around a series of essential procedures. First, it receives and processes the dataset provided by the user, which is then utilized to develop the prediction model. Before training the model, the system will preprocess the data to ensure the dataset is properly prepared and structured for effective modeling. This includes data cleaning, missing data, and

feature extraction, among others. Then the system trains a prediction model using Python modules and machine learning tendencies to find out the patterns and relations between different parameters of a given dataset according to the preprocessed data. The model is known to generate outputs based on the value that the user has entered after the model has been properly trained. Such outcomes, however, typically apply to specific situations, events, or expectations—for example, ensuring that health insurance premiums are calculated accurately based on the provided inputs.

YOLOv8 (You Only Look Once version 8) is the most recent iteration of the popular YOLO family of real-time object detection models. It combines speed and accuracy, making it perfect for applications needing high-performance detection. By treating object detection as a single regression issue, YOLOv8 performs better than models that depend on region suggestions or multi-step procedures. The first step in the process is picture preparation, which comprises resizing, usually to 640×640 pixels, and normalization, which involves scaling pixel values between 0 and 1. The image is then split into a grid, and a convolutional neural network (CNN) examines it to extract pertinent characteristics.

Utilizing CSP-Darknet, the backbone network assists feature extraction by detecting spatiotemporal information, reducing the dimensions of the feature maps, and deepening them. In order to successfully detect objects that are of various sizes, items in varied scales are captured by YOLOv8 by incorporating a feature pyramid network and a path aggregation network for better feature extraction to be obtained to capture both higher- and lower-level features. The model's head is responsible for predicting bounding boxes and class probabilities, utilizing multiple scales to improve accuracy across small, medium, and large objects. Anchor boxes and regression are used to make bounding box predictions, with different bounding boxes projected for each grid cell. In relation to the anchor box, the model forecasts each bounding box's height, width, and center coordinates.

The objectness scores are computed to forecast if an object will be present in the previously given bounding box, whereas class predictions result in the likelihood of a class. After determining the Intersection Over Union (IoU), YOLOv8 eliminates multiple bounding regions for the same item using Non-Maximum Suppression (NMS). Suppressed boxes are those with more than a specific IoU score. YOLOv8 optimizes a loss function that includes the classification loss, the confidence loss, and the localization loss (IoU loss) following the proper weighting of each component. The model's confidence loss is based on binary cross-entropy, whilst cross-entropy is used for the categorization loss and IoU-based metrics are used for the localization loss.



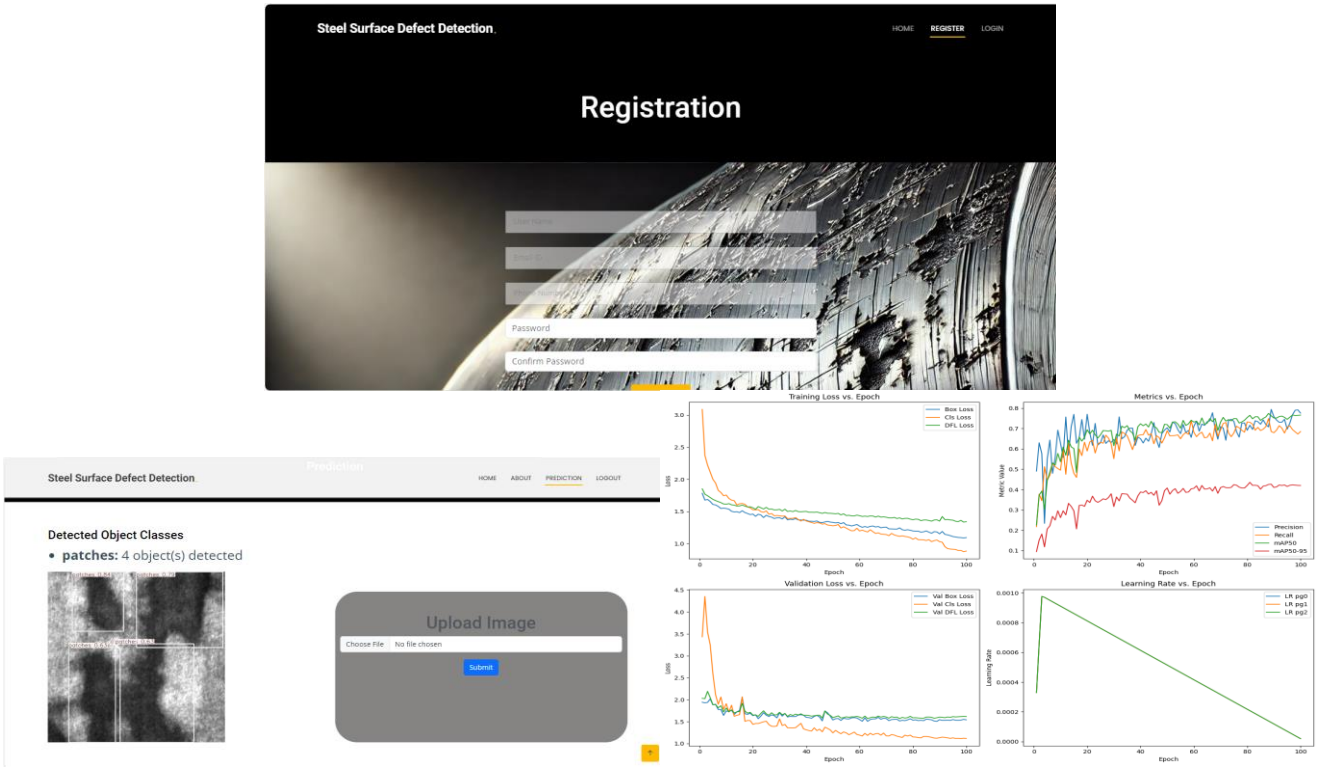


FIGURE 5: Results

DISCUSSION

Indeed, YOLOv8 is a tremendous breakthrough in the models of real-time object detection based on the synergy between the architectural innovations towards better accuracy and speed, which were inherited from its predecessors. CSP-Darknet being the backbone network can provide for the trade-off between detection precision and computing cost to be solved so that proper feature extraction can take place. To help the model quickly and accurately detect an object using an efficient dimensionality reduction for relevant spatial properties of the input data, this backbone is useful in this situation.

The helix connecting the FPN and PAN networks, which enables the multi-scale feature fusion, was the major advance that TYCO concentrated on. Due to its dual-path structure, which retrieves both high-level semantic variables and low-level spatial features, the model is robust in terms of object identification capabilities. The way that YOLOv8 detects tiny objects by utilizing different sizes reflects previous problems.

Anchor boxes and regression operations are employed in the object coordinate and dimension capture for every grid cell in the YOLOv8 bounding box prediction. The sigmoid function is used to provide prediction stability, which guarantees that the bounding box offsets fall within a specific limit. Rather, the predefined anchor boxes may be restrictive if the objects to be detected change greatly from the anchor boxes in terms of aspect ratio.

Non-Maximum Suppression (NMS), an integral element of the post-processing stage of YOLOv8, decreases the probability of several parallels for an item. The model ensures that only the most certain detections are retained by computing the IoU for overlapping boxes and suppressing those that had an IoU score above a threshold. The accuracy of detection and reduction of false positives also require this stage.

There are three primary parts to the YOLOv8 loss function. loss of localization, categorization, and confidence. The localization loss, which is usually IoU-based and represents an inaccuracy between

predicted and ground truth bounding boxes, emphasizes the significance of precise object localization. The confidence loss uses binary cross-entropy to quantify the objectness predictions of the model and make sure it is immune to false detections. Finally, depending on the precision of the class predictions in cross-entropy, the classification loss motivates the model to effectively distinguish between the object categories.

While YOLOv8 excels in speed and accuracy for object detection, several challenges remain, including managing computational costs on resource-limited devices and improving the detection of overlapping objects and small objects in complex backgrounds. Performance can be further enhanced in specialized scenarios by fine-tuning the model on specific datasets and adjusting hyperparameters, such as anchor box dimensions and IoU thresholds.

CONCLUSIONS AND RECOMMENDATIONS

Conclusion

Indeed, YOLOv8 is a tremendous breakthrough in the models of real-time object detection based on the synergy between the architectural innovations towards better accuracy and speed, which were inherited from its predecessors. CSP-Darknet being the backbone network can provide for the trade-off between detection precision and computing cost to be solved so that proper feature extraction can take place. To help the model quickly and accurately detect an object using an efficient dimensionality reduction for relevant spatial properties of the input data, this backbone is useful in this situation.

The helix connecting the FPN and PAN networks, which enables the multi-scale feature fusion, was the major advance that TYCO concentrated on. Due to its dual-path structure, which retrieves both high-level semantic variables and low-level spatial features, the model is robust in terms of object identification capabilities. The way that YOLOv8 detects tiny objects by utilizing different sizes reflects previous problems.

Anchor boxes and regression operations are employed in the object coordinate and dimension capture for every grid cell in the YOLOv8 bounding box prediction. The sigmoid function is used to provide prediction stability, which guarantees that the bounding box offsets fall within a specific limit. Rather, the predefined anchor boxes may be restrictive if the objects to be detected change greatly from the anchor boxes in terms of aspect ratio.

Non-Maximum Suppression (NMS), an integral element of the post-processing stage of YOLOv8, decreases the probability of several parallels for an item. The model ensures that only the most certain detections are retained by computing the IoU for overlapping boxes and suppressing those that had an IoU score above a threshold. The accuracy of detection and reduction of false positives also require this stage.

There are three primary parts to the YOLOv8 loss function. loss of localization, categorization, and confidence. The localization loss, which is usually IoU-based and represents an inaccuracy between predicted and ground truth bounding boxes, emphasizes the significance of precise object localization. The confidence loss uses binary cross-entropy to quantify the objectness predictions of the model and make sure it is immune to false detections. Finally, depending on the precision of the class predictions in cross-entropy, the classification loss motivates the model to effectively distinguish between the object categories.

While YOLOv8 excels in speed and accuracy for object detection, several challenges remain, including managing computational costs on resource-limited devices and improving the detection of

overlapping objects and small objects in complex backgrounds. Performance can be further enhanced in specialized scenarios by fine-tuning the model on specific datasets and adjusting hyperparameters, such as anchor box dimensions and IoU thresholds.

Recommendations

To improve the efficiency of automatic corrosion detection systems using image processing, several key recommendations can be followed. First, accurately identifying genuine corrosion patterns and incorporating advanced machine learning techniques, such as deep learning models like CNNs, can significantly boost detection accuracy. The model's robustness and versatility can also be enhanced by expanding the dataset to include a diverse range of surfaces and corrosion types across varying climatic conditions. Real-time processing can be achieved by integrating frameworks like OpenCV or TensorFlow Lite, enabling on-site corrosion detection using mobile or embedded devices. Additionally, optimizing image pre-processing—through methods like contrast enhancement, adaptive thresholding, and histogram equalization—can improve detection performance in challenging lighting conditions. Applying edge detection algorithms and texture analysis allows the system to identify even the most subtle or minute corrosion features.

Rotation, flipping, and scaling are data augmentation techniques that decrease overfitting through varying the training sample to accommodate more cases. By integrating with predictive analytics through which corrosion process can be understood, these analytics will help the firm plan maintenance and take action when may be necessary thus save the firm money. The facility would be able to develop easy to use interface, or mobile app, which would allow convenient input of data and representation of corrosion detection results, e.g., severity evaluations and corrective actions. Moreover, a broad picture of the risk of corrosion may be provided by Internet of Things sensors controlling such factors as the temperature and the humidity as well.

REFERENCES

[1] A. Landstrom and M. J. Thurley, "Morphology-based crack detection for steel slabs," IEEE J. Sel. Topics Signal Process., vol. 6, no. 7, pp. 866–875, Nov. 2012.

[2] K. Song and Y. Yan, "A noise-robust method based on completed local binary patterns for detecting surface defects in hot-rolled steel strips," Appl. Surf. Sci., vol. 285, pp. 858–864, Nov. 2013, doi: 10.1016/j.apsusc.2013.09.002.

[3] Y. J. Jeon, D. Choi, S. J. Lee, J. P. Yun, and S. W. Kim, "Steel-surface defect detection using a switching-lighting scheme," Appl. Opt., vol. 55, no. 1, pp. 47–57, 2016, doi: 10.1364/AO.55.000047.

[4] R. Girshick, J. Donahue, T. Darrell, U. Berkeley, and J. Malik, "R-CNN: Region-based convolutional neural networks," in Proc. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 2–9. 148824 VOLUME 12, 2024 T. Zhang et al.: GDM-YOLO: A Model for Automatic Corrosion Detection of Metal Surfaces by Using Image Processing Techniques Based on YOLOv8s

[5] R. Girshick, "Fast R-CNN," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1440–1448.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 9, pp. 1904–1916, Sep. 2015, doi: 10.1109/TPAMI.2015.2389824.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[8] Y. Xu, D. Li, Q. Xie, Q. Wu, and J. Wang, "Automatic defect detection and segmentation of tunnel surface using modified mask R-CNN," Measurement, vol. 178, Jun. 2021, Art. no. 109316.

[9] M. Chen, L. Yu, C. Zhi, R. Sun, S. Zhu, Z. Gao, Z. Ke, M. Zhu, and Y. Zhang, "Improved faster R-CNN for fabric defect detection based on Gabor filter with genetic algorithm optimization," Comput. Ind., vol. 134, Jan. 2022, Art. no. 103551.