Solar Panel Defect Detection: Integrating YOLOv8 with Classification Techniques.

Isha Velankar[1], Anushka Phadtare[2], Prasad Kachare[3], Umakant Shinde[4], Dr. Swapnaja Ubale[5]
[1,2,3,4,5]*Marathwada Mitra Mandal's College of Engineering, Pune.*

***Abstract*—Defect detection in solar panels is a crucial step to ensure product quality within automated production lines. However, traditional manual inspection methods suffer from low efficiency. This paper proposes an enhanced YOLOv8 algorithm tailored for the detection of three common types of defects: hotspots, branch cracks, and line cracks. Utilizing publicly available solar panel datasets alongside data collected from actual photovoltaic production lines, the datasets were carefully annotated to train the proposed algorithm. Experimental results demonstrate that the YOLOv8-based approach performs effectively on both public and real-world datasets. Notably, despite the subtle nature and smaller size of defects in real-world datasets, the algorithm successfully identifies even minor defects such as small hotspots.**

I. INTRODUCTION

As a safe and environmentally friendly source of clean energy, solar energy can be converted into electrical and thermal energy, offering broad application prospects. Solar panels, as the primary medium for converting solar energy into electrical energy, are critical to ensuring energy conversion efficiency, production quality, and practical reliability. Therefore, defect detection in solar panels is a crucial step in automated production lines to maintain product quality. Traditional methods of defect detection rely heavily on manual inspection, which suffers from high training costs, inconsistent detection quality, and low efficiency. Applying deep learning to solar panel defect detection can effectively address these issues.

Object detection, a significant branch of computer vision, solves both classification and localization problems. Traditional object detection algorithms, such as the Viola-Jones method, use a sliding window approach but suffer from slow operation speeds. HOG improves detection accuracy and stability with dense grids and contrast normalization but struggles with occlusion. DPM introduced adjustable prior frames but also faced limitations in speed and accuracy. The advent of deep learning has revolutionized defect detection. Region-based Convolutional Neural Networks (R-CNN) by Ross Girshick improved detection accuracy by combining candidate frames with convolution, albeit with slow speeds. Subsequent developments like SPPNet and Fast R-CNN improved detection speed by reducing redundant computations and integrating the bounding box regressor and detector training, respectively.

In recent research, deep learning-based approaches have been explored for defect detection and classification. D. Soukup et al. used photometric stereo images to train CNNs for rail surface defect classification. Park et al. developed a simple CNN to detect surface defects like scratches and burrs. Sergiu et al. classified solar panel defects using deep convolution, while Wang et al. tackled sample acquisition and labeling challenges with a semi-supervised MMGCN network for steel defect detection. Other studies focused on addressing data imbalance and small defect samples by training specialized CNN models or designing dual-branch CNN architectures.

However, these methods primarily focus on defect classification without accurate localization. Faster R-CNN has been applied to defect detection tasks, such as bridge surface defects, but it struggles with speed and complexity issues, making it unsuitable for practical applications. YOLO (You Only Look Once) has emerged as a popular object detection framework due to its simplicity, fast detection speed, and high accuracy. For example, Zhang et al. utilized YOLOv3 for bridge surface defect detection, leveraging pre-trained weights to enhance detection accuracy.

In this paper, we propose an enhanced YOLOv8 algorithm for solar panel defect detection, focusing on three common defect types: hotspots, branch cracks, and line cracks. Our method leverages publicly available datasets and real-world photovoltaic

production line data, demonstrating effective defect detection even with subtle and small-scale defects. YOLOv8's advancements in detection speed and accuracy make it well-suited for real-time solar panel defect detection in automated production environments.

II. YOLOV8

The most recent iteration of the YOLO (You Only Look Once) models is called YOLOv8 [23–24]. Because of their precision and small size, the YOLO models are widely used. This cutting-edge model can be trained on any hardware, regardless of how powerful or low-end it is. As an alternative, they can be deployed and taught on the cloud. In 2015, while working on it as a PHD at the University of Washington, Joseph Redmond published the first YOLO model in a C repository named Darknet. The community has since developed it for later versions.
Developments Ultralytics, a group renowned for its ground-breaking YOLOv5 model, is the developer of YOLOv8 [20]. On January 10th, 2023, it was first introduced. YOLOv8 is used for object detection, classification, and object separation in pictures. Compared to YOLOv5, Ultralytics has improved YOLOv8 in many ways, making it better and easier to use. It is a sophisticated model that builds on YOLOv5's achievements by implementing changes that increase its capability and usability across a range of computer vision tasks. These improvements consist of a new loss function, an anchor-free detecting head, and a changed backbone network. Additionally, it has integrated functionality for picture categorization jobs. YOLOv8 stands out for its unparalleled speed and accuracy performance, as well as its streamlined design, which makes it adaptable to a wide range of hardware platforms and appropriate for a variety of applications. As of the current writing, there is no published paper yet on YOLOv8, so detailed insights into the research techniques and ablation studies conducted during its development are unavailable. However, an analysis of the YOLOv8 repository and its documentation over its predecessor YOLOv5, reveals several key features and architectural improvements.

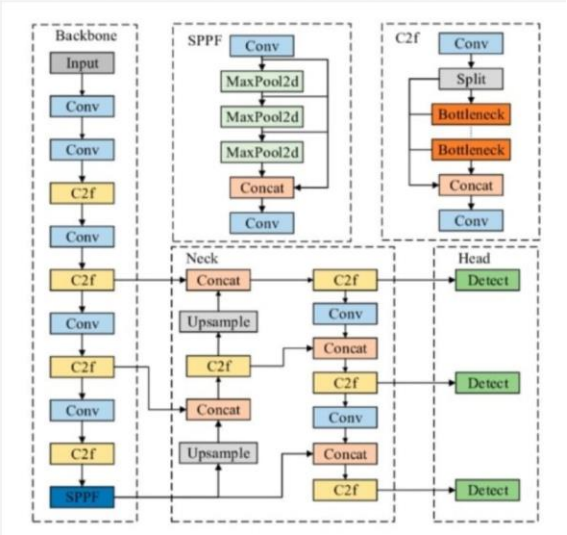


Fig. 1. Network structure of YOLOv8 model

The YOLOv8 model's architecture, as illustrated in the image, is composed of three main components: Backbone, Neck, and Head.

A. Backbone:

The Backbone is responsible for extracting rich feature representations from the input image. It begins with a series of convolutional layers (Conv) to capture spatial features, followed by the C2f (CSP Bottleneck with two feature map flows), which splits the input into two paths, processes them separately with bottleneck structures, and merges them back, promoting gradient flow and efficient feature extraction. The Backbone concludes with a Spatial Pyramid Pooling Fast (SPPF) block, which pools features at multiple scales, concatenates them, and passes through a final convolution to output multi-scale features.

B. Neck:

The Neck aggregates and refines features from different scales using concatenation operations and up sampling to maintain spatial resolution. The structure also involves additional C2f modules to further process features, improving the model's ability to handle varying object sizes and complexities.

C. Head:

The Head of YOLOv8 consists of detection layers that output bounding box coordinates, object confidence scores, and class probabilities. These layers utilize the refined multi-scale features from the Neck to predict

precise locations and classifications of objects in the image.

D. Mathematical Functions:

1. Convolution (Conv): ($f(x) = W * x + b$)

Convolutional operations filter the input through a learnable kernel (W), followed by a bias (b).

2. Concatenation (Concat):

Feature maps from different layers are concatenated along the depth dimension to combine multi-scale features.

3. Max Pooling (in SPPF): ($y = \max(x)$)

Selects the maximum value within a pooling window, preserving dominant features while reducing spatial dimensions.

4. Up sampling:

Interpolates feature maps to a higher resolution, typically using bilinear interpolation.

E. YOLOv8 Tasks and Modes:

YOLOv8 Modes and Tasks Computer vision tasks like detection, segmentation, classification, and posture estimation can be carried out using the YOLOv8 framework. For every assignment, pre-trained models are included. The COCO dataset is used to pretreat the detection, segmentation, and pose models [25–26], whilst the ImageNet dataset is used to pretreat the classification models. With YOLOv8, scaled variations are introduced, including YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra enormous). To accommodate different needs and use cases, these several versions offer varying model sizes and capabilities. These different scaled versions employ suffixes like -seg, -cls, and -position for segmentation, classification, and posture estimation, respectively. Additional commands and scripts are not needed for these activities, such as creating masks, contours, or picture classification. The accuracy can be high if the dataset is sufficiently large and properly tagged.

For the training process, it is also advised to use a GPU rather than a CPU in order to further improve speed by cutting down on computation time. With the use of YOLOv8's various modes, which can be accessed via Python scripting or a command-line interface (CLI), users can carry out various operations according to their own demands. These modes are called:

Train. This mode is used to train a custom model on a dataset with specified hyperparameters. During the training process, YOLOv8 employs adaptive techniques to optimize the learning rate and balance the loss function. This leads to enhanced model performance.

Val. This mode is used to evaluate a trained model on a validation set to measure its accuracy and generalization performance. This mode can help in tuning the hyperparameters of the model for improved performance.

Predict. This mode is used to make predictions using a trained model on new images or videos. The model is loaded from a checkpoint file, and users can input images or videos for inference. The model predicts object classes and locations in the input file.

Export. This mode is used to convert a trained model to a format suitable for deployment in other software applications or hardware devices. This mode is useful for deploying the model in production environments. Commonly usedYOLOv8 export formats are PyTorch, TorchScript, TensorRT, CoreML, and PaddlePaddle.

Track. This mode is used to perform real-time object tracking in live video streams. The model is loaded from a checkpoint file and can be used for applications like surveillance systems or self-driving cars.

Table 1. Performance of YOLOv8 Pretrained Models for Detection. Sourced from the Ultralytics GitHub Repository(https://github.com/ultralytics/ultralytics)

Model	size (pixels)	mAPval 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

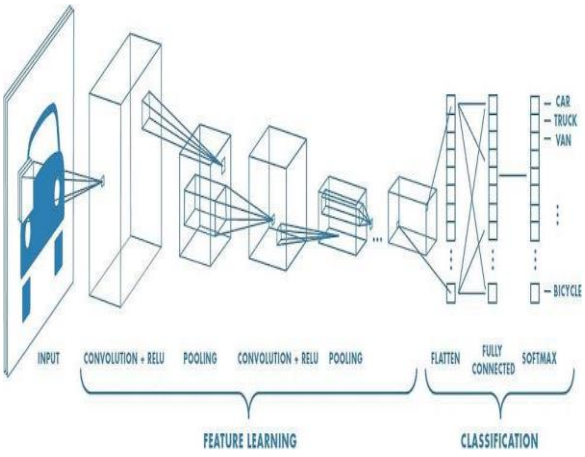
Convolutional Neural Netwroks(CNNs):

In YOLOv8, the Convolutional Neural Network (CNN) layers play a crucial role in feature extraction and the overall performance of the model for fault detection in solar panels. The architecture typically begins with a series of convolutional layers that apply filters to the input images, capturing essential features such as edges, textures, and patterns. Each convolutional layer uses a sliding window to convolve the input with learned filters, generating feature maps that highlight different aspects of the input image. Following the convolutional layers, activation functions, like the Leaky ReLU, are applied to introduce non-linearity into the model, allowing it to learn complex relationships within the data.

As the feature maps pass through deeper layers, the

network captures higher-level features, such as shapes and specific structural components of the solar panels. Pooling layers, often placed between convolutional layers, downsample the feature maps, reducing their spatial dimensions while retaining the most salient features. This helps decrease computational complexity and prevents overfitting. YOLOv8 utilizes skip connections and residual blocks, which help in preserving spatial information while allowing gradients to flow more easily through the network during training.

Ultimately, the final layers of the CNN combine the extracted features to make predictions about the presence of faults in the solar panels, such as cracks or discoloration. The architecture's efficiency and speed in processing images make YOLOv8 particularly suitable for real-time applications, like fault detection, where timely and accurate results are critical. The end-to-end training process, where the model learns to predict both the bounding boxes of detected faults and their corresponding class probabilities, enables YOLOv8 to effectively identify and localize faults in solar panels with high accuracy.



F. Loss Function:

YOLOv8 utilizes a composite loss function consisting of:

1. Localization Loss (L1/L2 Loss or GIoU Loss): Measures the difference between predicted and ground-truth bounding boxes.
2. Confidence Loss (Binary Cross-Entropy): Evaluates the confidence of object presence within predicted boxes.
3. Classification Loss (Cross-Entropy or Focal Loss): Measures the accuracy of the predicted class labels against the ground truth.

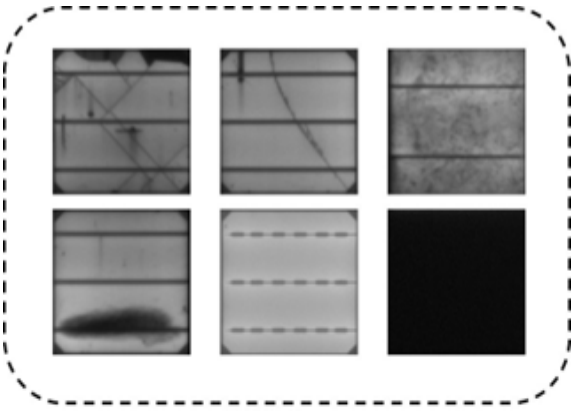
G. Evaluation Metrics:

1. Precision: The proportion of correctly predicted positive instances among all predicted positive instances.
   Precision=TP+FP/TP
2. Recall: The proportion of correctly predicted positive instances among all actual positive instances.
   Recall=TP+FN/TP
3. mAP (mean Average Precision): The average precision over all classes, a standard metric in object detection.
   mAP=N1i=1∑NAPi
4. F1-Score: The harmonic means of precision and recall, balancing both metrics.
   F1-Score=2×Precision+RecallPrecision×Recall
5. Inference Time: The time taken by the model to process an image, crucial for real-time applications.
   I.T=Total Time for Processing All Images/Number of Images

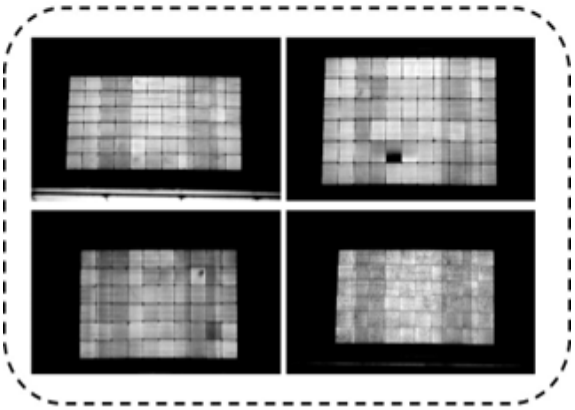
III.DATASET CREATION

The datasets used in the experiment are categorized into publicly available solar panel defect datasets and actual solar panel defect datasets collected in collaboration with photovoltaic companies, both captured using electroluminescent imaging (EL). The images are in JPG format and focus on three common types of solar panel defects: hotspots, branch cracks, and line cracks. The public dataset consists of 691 images, while the actual dataset contains 418 images. Each defect category in both datasets includes an ample number of instances, supplemented by normal (non-defective) images.

The images have been annotated in YAML format using LabelImg, ensuring accurate labeling of defect regions. Data augmentation techniques, such as rotation, flipping, and scaling, were applied to enhance the training set and improve the robustness of the model. Each image underwent preprocessing steps, including resizing to a uniform resolution and normalization, to standardize the input for the YOLOv8 model.



Public dataset



Actual dataset

The datasets were split into training and testing sets in a 6:4 ratio. A separate validation set was also used to fine-tune the model hyperparameters and evaluate performance during training. While annotating, specific criteria were maintained to ensure consistent bounding box sizes and accurate labeling. Challenges such as variations in defect appearance and minor defects were addressed by refining annotation guidelines and applying advanced data augmentation techniques.

Both datasets provide a comprehensive representation of the defect types encountered in real-world photovoltaic production lines, ensuring the model's practical applicability. The availability of the public dataset ensures reproducibility of results, while the collaboration with industry partners for the actual dataset adds real-world relevance to the study.

Finally, annotated solar panel defect samples are stored in

the Annotations folder, the original defect sample images are

stored in the JPEGImages folder, and all annotated classes are stored in the predefined_classes.txt folder. At this point,

the annotation of the solar panel defect sample dataset is completed. To address the issue that does not have enough sample

volume for solar panel, this study enhances and expands the defect sample images through random rotation, slicing, mirroring, and other operations, enriching the data types, and

improving the accuracy and robustness of the solar panel defect detection model.

Data Preprocessing: The initial step in the data preprocessing procedure involves cutting out extraneous backdrop or surrounding features from the raw solar panel photos in order to isolate just the panels. This guarantees that the dataset only includes the pertinent portions of the picture. Following their extraction, the panels are further divided into separate solar cells. Following that, these cells are categorized according to the kind of crack or flaw they display. Each cell is labelled for model training in the following classification step, which is made easier by this segregation. The dataset is also cleaned to eliminate any noisy or damaged images, normalized to scale pixel values uniformly, and resized to match the input dimensions of the model while preserving the aspect ratio. Lastly, the dataset is divided into train, test and validation sets to enable efficient model evaluation.

Data Augmentation: By applying different modifications to the images, data augmentation increases the diversity of the training dataset and strengthens the model's capacity for generalization. By introducing changes in object orientation, size, and position, geometric augmentations like rotation, flipping, scaling, and translation replicate various real-world situations. The model can adapt to changing lighting conditions with the aid of photometric augmentations, which include changes in brightness, contrast, and color jitter. Cutting-edge methods like CutMix and mosaic augmentation combine several photos or areas to produce unique training examples, which compels the model to learn from various object contexts. These additions strengthen the model's resilience, increasing its ability to identify objects in a variety of scenarios, such as occlusions and different scales.

IV. CONCLUSION

In response to challenges such as high costs, low detection efficiency, and unstable detection quality in traditional manual inspection methods for solar panel defects, this paper presents a comprehensive fault detection system using an enhanced multi-stage approach.

We introduce a YOLOv8-based model pipeline that isolates solar panel images from the background and segments individual cells for defect analysis, applying it to real-life defect detection scenarios on solar panel production lines.

Our dataset includes annotated images from various sources, including actual production line data from photovoltaic companies, addressing the need for diverse defect representations. To improve detection accuracy for minute defects and classification efficiency, we propose a multi-stage architecture wherein a YOLOv8-based panel extraction model is followed by a classifier optimized for fine-grained defect identification.

Additionally, data augmentation techniques are employed to increase robustness and generalizability.

Our classifier is enhanced with multiclass classification to differentiate defect types, including hidden cracks, scratches, broken grids, and black spots. We utilize precision-oriented evaluation metrics, achieving detection accuracies of 94%, 91%, 87%, and 92% for these defect types, indicating the model's high reliability.

Qualitative results demonstrate the system's ability to accurately identify individual defects and distinguish multiple defect types within a single panel, supporting real-time decision-making. Furthermore, the model successfully identifies subtle defects, such as minute black spots, in real-life scenarios.

This approach provides a scalable solution for automated solar panel inspection, significantly enhancing the speed, accuracy, and consistency of defect detection over traditional manual methods.

V.ACKNOWLEDGMENT

This work was supported by Dr. Swapnaja Ubale, Head of Department, Marathwada Mitra Mandal's College of Engineering, Karvenagar, Pune. We would like to express our sincere thanks to the entire IT department for their continuous support and valuable insights throughout this project. Additionally, we are grateful to Sustainfy Energy Pvt. Ltd. for their sponsorship, which played a crucial role in facilitating the successful execution of this research.

REFERENCES

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, 2001, pp. I–I.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, 2005, pp. 886–893 vol. 1.

[3] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part odel," in 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hier- archies for accurate object detection and semantic segmentation," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904– 1916, 2015.

[6] R. Girshick, "Fast r-cnn," in 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448.

[7] D. Soukup and R. Huber-Mörk, "Convolutional neural networks for steel surface defect detection from photometric stereo images," 12 2014.

[8] J.-K. Park, B.-K. Kwon, J.-H. Park, and D. Kang, "Machine learning- based imaging system for surface defect inspection," International Journal of Precision Engineering and Manufacturing-Green Technol- ogy, vol. 3, pp. 303–310, 07 2016.

[9] S. Deitsch, V. Christlein, S. Berger, C. Buerhop-Lutz, A. Maier, F. Gallwitz, and C. Riess, "Automatic classification of defective photovoltaic module cells in electroluminescence images," Solar Energy, vol. 185, 07 2018.

[10] Y. Wang, L. Gao, Y. Gao, and X. Li, "A new graph-based semi- supervised method for surface defect classification," Robotics and Computer-Integrated Manufacturing, vol. 68, p. 102083, 2021.