Comparing performances of deep learning models to classify and detect defects in wind turbine blades

Letícia V. Gonçalves
Univ. Federal Rural do Semi-Árido
Rio Grande do Norte, Brasil
leticia.vieira@alunos.ufersa.edu.br

Rosana C. B. Rego
Departamento de Engenharias e Tecnologia
Univ. Federal Rural do Semi-Árido
Rio Grande do Norte, Brasil
rosana.rego@ufersa.edu.br

Paulo H. A. Bezerra
Departamento de Engenharias e Tecnologia
Univ. Federal Rural do Semi-Árido
Pau dos Ferros, Brasil
paulo.bezerra@ufersa.edu.br

***Abstract*—Automated inspection has become a relevant tool for structural health monitoring in civil infrastructure, industrial, and energy generation domains, especially in components subject to harsh environmental conditions. This study presents a comparative evaluation of three Deep Learning (DL) architectures for detecting surface cracks in wind turbine blades: a sequential Convolutional Neural Network (CNN) for image classification; a U-Net architecture for segmentation, combined with a CNN classifier; and an object detection model based on an adapted YOLOv8 architecture. All models were trained and evaluated using a public dataset, which comprises images of wind turbine blades subdivided into healthy and faulty classes. Oversampling was employed to improve data quality. Each model was assessed based on performance metrics and suitability for crack detection in blade images. The CNN model provided a baseline for classification tasks, while the U-Net model demonstrated enhanced sensitivity to fine discontinuities. The YOLOv8-based detector allowed simultaneous classification and localization of defects. This study highlights trade-offs between classification and detection approaches, emphasizing the importance of selecting appropriate models depending on application constraints.**

***Index Terms*—Deep Learning, Defect, Detection, Wind Turbine, Inspection.**

I. Introduction

Automated defect detection has gained attention across industrial and civil infrastructure domains due to its potential to enhance inspection efficiency, consistency, and accuracy [1], [2]. Traditional manual inspections are often time-consuming and limited by subjectivity, human error, and the impracticality of assessing large or inaccessible infrastructure, especially under hazardous or remote conditions [3]. In contrast, automated approaches, especially those driven by Artificial Intelligence (AI), offer rapid, objective, consistent, and reproducible evaluations that may improve the maintenance and ensure the quality of processes [4]. These technologies, which diminish inspection time and cost, have been successfully applied in sectors as civil infrastructure, manufacturing, transportation, and energy systems; where identifying defects in advance contributes to operational safety and reduces unplanned downtime [5].

Recent advancements in AI, especially in deep learning (DL) techniques, have benefited automated fault diagnosis in wind turbine components [6], [7]. Convolutional Neural Networks (CNNs) have demonstrated high performance in extracting discriminative features for defect classification and localization tasks, from high-dimensional image data, enabling intelligent fault diagnosis even under complex environmental conditions [5], [8]–[10]. Moreover, advancements in data-driven methodologies such as transfer learning, generative adversarial networks (GANs), and digital twin (DT) technologies have further expanded the applicability of automated defect detection systems [11], [12]. These developments indicate a paradigm shift toward intelligent, autonomous inspection systems capable of sustaining operational reliability across several engineering applications.

UFERSA - Edital funding agency here

In power generation operations, wind farm turbines require rigorous inspection and maintenance to ensure optimal performance and durability. Among their components, wind turbine blades are particularly vulnerable to surface defects due to constant exposure to varying environmental conditions, including wind, rain, ultraviolet radiation, and debris impact [13]. Defects such as cracks and erosion can compromise the aerodynamic performance of blades and lead to failures if undetected [14]. With wind power expected to reach up to 1,200 GW of global installed capacity by 2030 [14], the development of reliable blade inspection systems has become relevant to reduce downtime, minimize operational costs, and extend the service life of wind turbines [11], [15].

Numerous studies have proposed using DL for detecting faults in wind turbine bearings [10], gearboxes [8], and blades [11], [14], [16]. Architectures such as ResNet, VGG, Xception, and custom CNNs, with or without transfer learning, have shown high accuracy scores in identifying visual anomalies in turbine blade images. Additionally, object detection models like YOLO have been employed to localize defects, improving damage identification in high-resolution imagery [11], [15].

Despite the mentioned advancements, several research gaps remain. Existing literature focuses on classification models, which label images as containing a defect, without providing information about its location or shape [5]. Object detection models, while addressing this limitation, often require more computational resources and annotated data. Moreover, comprehensive comparisons between classification models and object detection models are limited. Questions usually regard their relative performance, computational efficiency, robustness to varying environments, and suitability for deployment in real-time systems [11], [15]. These open issues underline the need for a structured performance analysis to guide the selection of appropriate models for specific wind turbine inspection scenarios.

To address some of these challenges, this study presents a comparison of DL models for automated defect detection in wind turbine blades. We investigate the performance of DL architectures applied to a dataset with images of wind turbine blades. Specifically, we assessed two classification models, a CNN and a CNN with U-net for segmentation, which detect whether a blade is healthy or damaged, as well as an object detection model, which seeks to identify and localize specific defects within the surfaces of blades. We analyzed accuracy, precision, recall, and F1-score for all models to analyze which is the most suitable approach for DL-based inspection of wind turbine blades.

II. Dataset and Models

In this study, we used a single dataset to analyze the following three models: Model 1, a sequential Convolutional Neural Network (CNN) model; Model 2, a U-Net combined with CNN model; and Model 3, an adapted YOLO model. The dataset and models are described in the next sections.

A. Dataset

We employed images of reduced-scale wind turbine blades, from the public dataset CAI-SWTB, available in [17], which comprises 4000 indoor and 2000 outdoor images (RGB, 300x300 pixels), split into Faulty and Healthy classes. To implement the models, we derived a new dataset from CAI-SWTB, the *Blade Crack*, with two image classes, *Faulty Crack Blades* and *Healthy Blades*. Figure 1 illustrates a sample from class *Faulty Crack Blades*, which contains images of faulty blades with cracks on their surfaces - at least one crack is present, regardless of other existing defects.

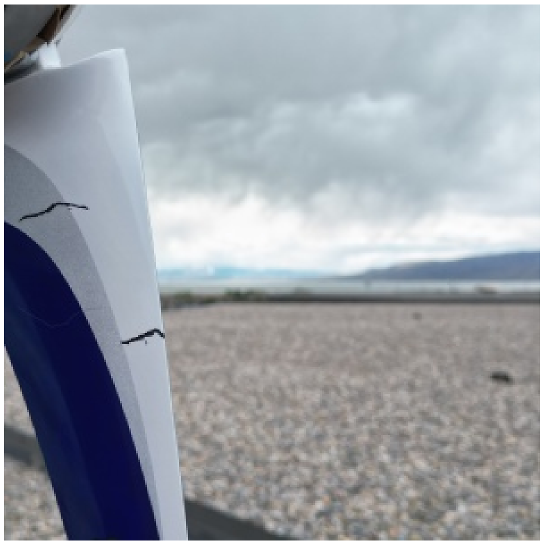


Fig. 1: Sample from Faulty class, image of a blade with cracks.

Images of class *Healthy Blades* have no defects, as the sample in Figure 2. We made no distinctions between indoor and outdoor wind turbine blade images. To ensure consistency and standardization in the performance evaluation of the DL models, we employed a single dataset, the *Blade Crack*, for all three models implemented in this work. The dataset was balanced using the *Random Over Sampling* technique with replacement, to randomly replicate its samples and increase the amount of data, providing a total of 5000 images per class.

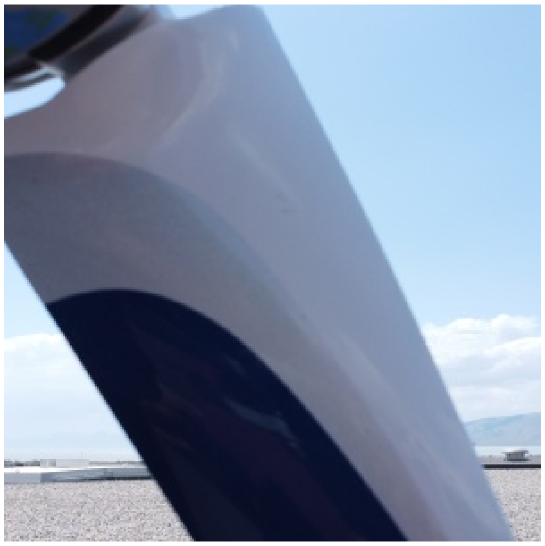


Fig. 2: Exemplar from Healthy class, image of a blade without cracks or any other defect.

The dataset was then divided into train and test subsets, with 80% and 20% of images, respectively. That division is essential to ensure the model will be trained on a representative dataset and learn intrinsic patterns within the images.

B. Model Implementation

All model implementations were conducted using the Google Colab cloud platform. Hardware acceleration was leveraged via NVIDIA Tesla T4 GPU (Turing architecture, 16 GB GDDR6 VRAM).

1) Model 1: To develop and train Model 1, we employed deep learning and image processing libraries, using Python 3.11.13 and TensorFlow 2.18.0. The model was implemented to detect cracks in images from class *Faulty Crack Blades* and has a sequential model architecture, in which each layer receives an input tensor and generates an output tensor, providing a continuous processing flow (Figure 3). The model's structure begins with a batch normalization layer, which normalizes input data, stabilizing and accelerating the training process. Next, the model incorporates two convolutional layers, *Conv2D*, followed by *Max Pooling* and *Dropout*. The first layer uses 512 filters with a kernel size of (3,3), while the second uses 256 filters with the same dimensions.

The training comprised 42 epochs, with a mean duration of 9 seconds, resulting in 6 minutes and 18 seconds of processing, yielding a validation accuracy exceeding 99% within the initial iterations.

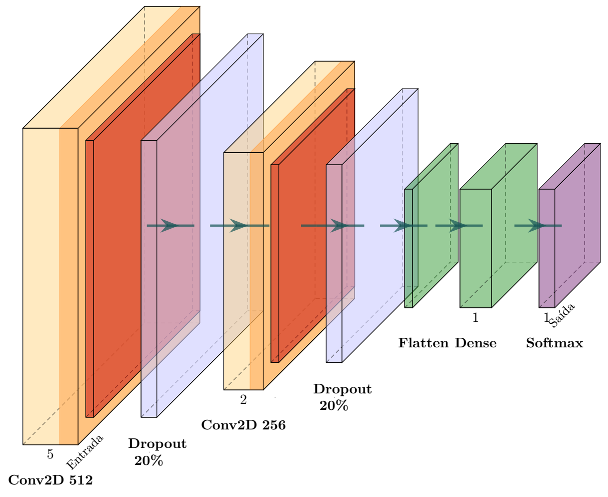


Fig. 3: CNN - Sequential Model.

The convolutional layers, with ReLU activation, focus on extracting hierarchical features from the images, ranging from local patterns, such as edges and textures, to more complex and abstract features. The Max Pooling layers, with 2×2 windows and a stride of 2, progressively reduce the spatial dimension while retaining the most relevant features. The Dropout layer, with a rate of 0.2, acts as a regularization mechanism, preventing overfitting during training.

After feature extraction, the model performs the Flatten operation on the data, transforming the multidimensional feature maps into a one-dimensional vector. This vector is then processed by a Dense layer with 32 neurons and ReLU activation, responsible for learning non-linear combinations of the extracted features. An additional Dropout layer is applied before the output layer to reinforce regularization.

The final layer of the model employs two neurons with Softmax activation, which is suitable for multi-class classification problems. This configuration enables the model to compute normalized probabilities for each class (the sum of probabilities is 1).

2) Model 2: Model 2 combined the U-net architecture and a CNN. U-Net architecture was employed for image segmentation, a process that divides the image into regions of interest, aiding information extraction [18]. Accordingly, we implemented an adapted U-Net architecture for the segmentation of cracks in images of wind turbine blades, with dimensional adjustments adapted to the particularities of the problem. Figure 4 presents original and segmented images, highlighting the importance of this step.

Segmentation is followed by the Softmax binary classifier to ultimately detect cracks. Furthermore, the model was built using TensorFlow 2.18.0 with three main components, namely Encoder, Bottleneck, and Decoder; and the *Times* library was used to measure the model's inference time. The model training spanned 100 epochs, with a duration of 15 minutes.

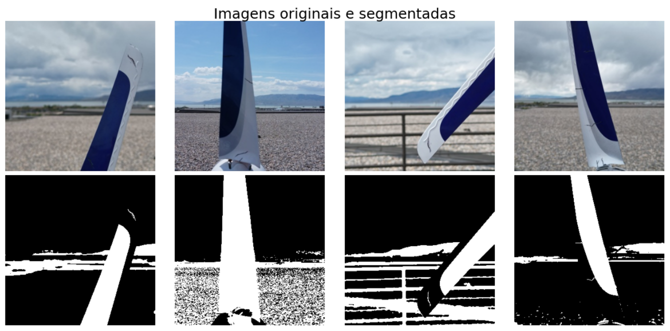


Fig. 4: Imagens originais e segmentadas.

The Encoder block, which performs feature extraction, consists of three sequential stages of double convolutional operations with kernels of size (3×3) and ReLU activation, each followed by MaxPooling with a 2×2 window for progressively reducing spatial dimensionality of images and information to be processed. Additionally, Early Stopping was used to monitor loss, with a patience of 15 epochs. We also used progressive filters of 64, 128, and 256 channels, which enable hierarchical feature capture — from simple edges to complex discontinuity features.

The bottleneck stage, with 512 filters, incorporates a 50% rate dropout layer as a regularization mechanism to prevent overfitting. The Decoder, which performs spatial reconstruction, employs bilinear upsampling layers concatenated with the Encoder's corresponding feature maps, followed by double convolutional layers. This approach enables precise recomposition of spatial dimensions while preserving high-resolution information for accurate crack localization. The final layer uses a 1×1 convolution with sigmoid activation to produce output binary masks (128×128 pixels).

This model presented higher storage consumption when compared to other approaches, since it saves the segmented images U-Net generates and subsequently inputs them into a sequential CNN model, an additional step to improve classification that enables the network to focus on the target regions.

The decoding path employs transposed convolution operations for progressive spatial reconstruction, with residual connections that transfer high-resolution information from the encoder to the decoder. This symmetric architecture enables the model to accurately segment even complex edge areas and narrow cracks, which are likely to be found in wind turbine blade inspections.

3) Model 3: Implementation of Model 3 employed the Ultralytics YOLO library, based on PyTorch 2.6.0, within an environment configured using Python 3.11.13. Specifically, Model 3 uses YOLOv8's architecture adapted for image analysis and operates via three main stages [19]. Initially, the progressive convolutional layers extract hierarchical features from the images, from simple edge patterns to complex discontinuity characteristics. Subsequently, a central block with attention mechanisms filters and highlights the most relevant features for crack detection. Finally, the spatial reconstruction phase combines information from different scales to produce the bounding boxes and segmentation masks of the detected anomalies (cracks).

Classification differs from object detection in that it identifies what exists in an image but does not specify the location of the object. Based on this distinction, detection combines both localization and classification. The YOLO algorithm performs detection and classification in a single pass through the neural network, offering high processing speed. It divides the image into a grid and, for each cell, predicts bounding boxes, the class probability for each detected object, and the confidence that the box contains an object.

YOLOv8 presents advantages in processing speed when compared to traditional U-Net-based approaches, making it more suitable for real-time applications [20]. However, the performance of Deep Learning models depends on the quality and quantity of the training data.

The YOLO model in this study was trained for 50 epochs, with a total training time of approximately 1 hour. Image preprocessing included resizing to 640×640 pixels, hue adjustments, and applying filters to enhance features. Furthermore, parameters were fine-tuned, including an initial learning rate of 0.01, a momentum of 0.937, and weight decay of 0.0005 as L2 regularization technique to prevent overfitting. The *Times* library was used to measure the model's inference time.

In addition, we employed techniques to enhance training stability and efficiency, such as mixed precision (`AMP=True`), and configured an early stopping mechanism to monitor the validation metric to prevent overfitting. During inference, a maximum of 300 detections per image was set, with an IoU threshold of 0.7.

The bounding boxes were created using Roboflow, a cloud-based platform that simplifies data preparation for computer vision models such as YOLO. This tool enables the drawing of bounding boxes and polygons to delineate cracks in training images. Subsequently, the annotations are organized in a file with a `.json` extension.

At the end of training, the mean average precision at IoU threshold 0.50 (mAP@50) reached 0.889, indicating a strong capability for crack detection, with a balanced performance between precision (0.809) and recall (0.748).

III. Results and Discussion

In this section, we present results for all the models. The main evaluation metrics employed were accuracy, precision, recall, F1-score, mAP, and confusion matrix.

A. *Performance of Model 1*

Figure 5 presents results of Model 1, in which we observe correct and misclassification examples of predictions. Two Faulty instances were incorrectly labeled as Healthy, although the remaining examples in Figure 5 were correctly classified.

Figure 6 illustrates the epoch-wise evolution of accuracy and loss for training and validation of Model 1. The training

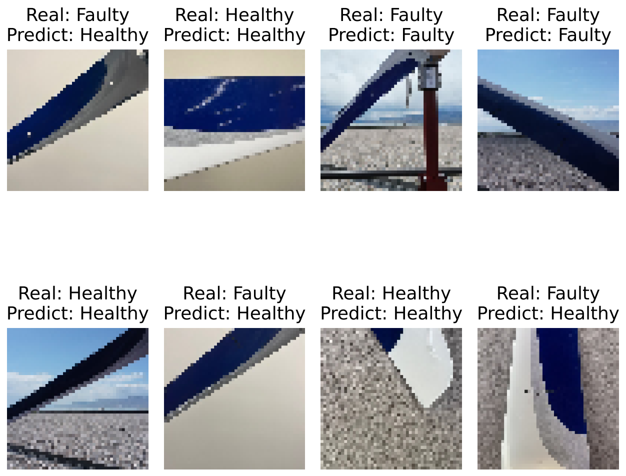


Fig. 5: Test results for model 1.

accuracy rapidly increases and stabilizes above 99%, reaching a final value of 0.9952 (Figure 6 a). The validation accuracy closely follows this trend, indicating that the model generalized well to unseen data without significant performance degradation. Therefore, the model can extract relevant features from the input data despite the limited number of epochs.

The training and validation loss curves in Figure 6b demonstrate a fast, sharp decrease, reaching low and stable values. The convergence of the curves with minor oscillations suggests smooth optimization and an absence of overfitting. The proximity between training and validation performance confirms that regularization techniques and hyperparameter tuning were adequate to ensure learning stability and generalization. These results validate the effectiveness of Model 1 in learning robust representations for the defect classification task.

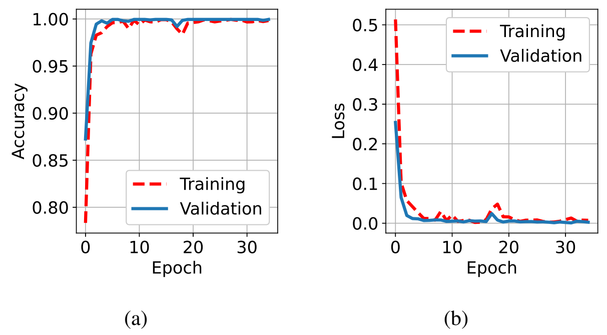


Fig. 6: Training and validation metrics of Model 1: (a) Accuracy and (b) Loss.

The detecting performance of Model 1 is substantiated by the confusion matrix in Figure 7. The model achieved an accuracy of 94.4%, a precision of 100%, a recall of 88.9%, and F1-score of 94.10%. These results demonstrate the model's strong generalization across classes.

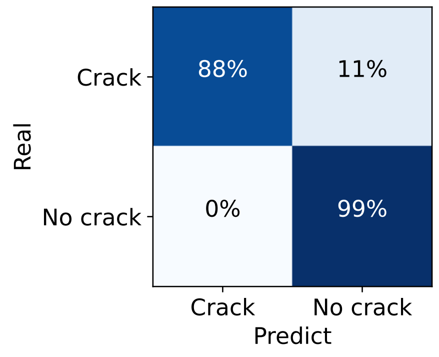


Fig. 7: Confusion matrix of Model 1.

B. *Performance of Model 2*

Model 2 presented the training and validation performances illustrated in Figure 8.

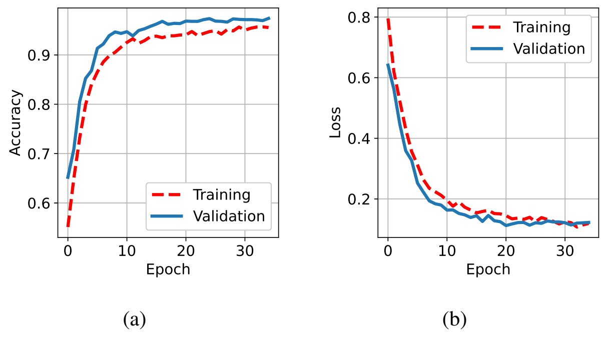


Fig. 8: Training and validation metrics of Model 2: (a) Accuracy and (b) Loss.

Training accuracy remained above 90% from epoch 8 onward, peaking at 96% (Figure 8a). Besides, training and validation curves behaved similarly, suggesting the model presented no overfitting. Validation loss is similar to the training loss, confirming the performance and stability of the model.

Image segmentation highlights regions of interest; however, it increases inference time due to the additional computational cost associated with the extended CNN processing. Moreover, it demands greater memory resources, as the segmented images must be temporarily stored before being fed into the CNN. This occurs because the images must undergo expensive preprocessing operations, such as pixel clustering or the application of binary masks. Furthermore, segmentation can result in loss of relevant information, particularly in images with low contrast or complex textures, forcing the CNN to operate on partially degraded data. This often requires additional post-processing layers to correct inconsistencies, increasing the system's latency.

The confusion matrix of Model 2 is illustrated in Figure 9. The model achieved an accuracy of 66.39%, a precision of 61.34%, a recall of 63.75%, and an F1-score of 62.50%. These results indicate a significant number of false positives and false negatives. The average inference time was 2.5942 seconds per image, indicating computational efficiency of Model 2 during the evaluation phase.

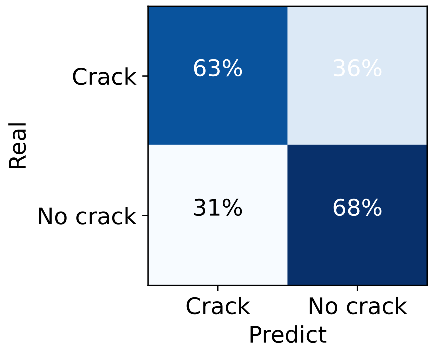


Fig. 9: Confusion matrix of Model 2.

This suggests a limitation in generalization when compared to Model 1. The relatively balanced misclassification rates imply that further optimization is required, possibly involving adjustments to the segmentation quality, or classification thresholds. Despite the use of a segmentation-based approach, Model 2 did not outperform the simpler classification-only model, which highlights the need to evaluate trade-offs between model complexity and actual predictive gains.

Figure 10 shows examples of results from Model 2 testing data.

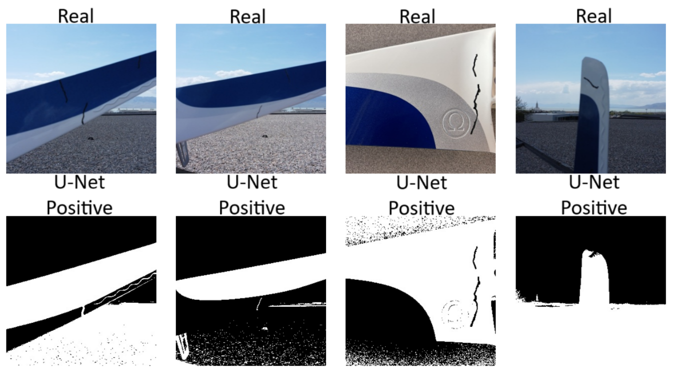
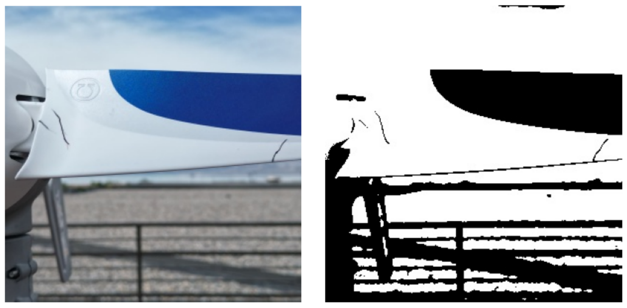


Fig. 10: Test results of Model 2.

Figure 11 exemplifies an original image and its respective segmented version output from U-Net, demonstrating a precise segmentation of 0.991.



(a) (b)

Fig. 11: U-Net segmentation example: (a) original image and (b) segmented image.

C. Performance of Model 3

Results in Figure 12 illustrate the automated detection of cracks in images, with ground truth annotations compared to YOLO model detections of cracks on wind turbine blades. Each ground truth box represents a manually labeled crack or defect, while the YOLO detections include bounding boxes labeled as cracks, along their confidence scores. The model successfully detected cracks in most samples. However, some detections are imprecise, either missing true positives (e.g., undetected cracks in the top-left) or generating false positives (e.g., blue boxes not aligned with any green ones).

Moreover, the bounding box predictions exhibited irregularities such as excessive overlaps or misalignments, indicating challenges in spatial reconstruction during the detection process. Confidence scores vary between 0.20 and 0.79, suggesting the model is relatively confident in some cases but uncertain in others, especially where cracks are subtle or visually complex. These inaccuracies may be attributed to inherent limitations of YOLO's grid-based segmentation approach or to inconsistencies in the training data annotations.

Furthermore, average inference time per sample was 0.003209 seconds and the model achieved an IOu of approximatley 0.85.

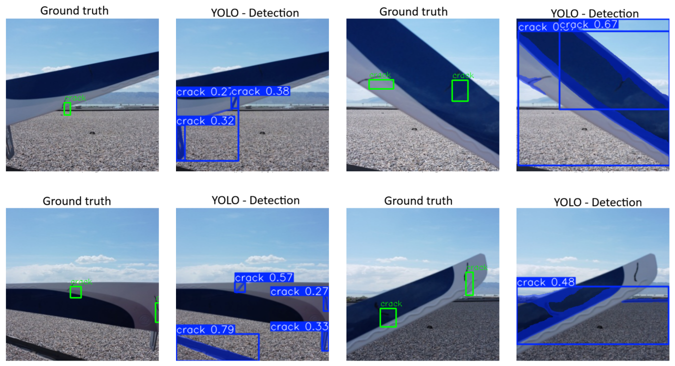
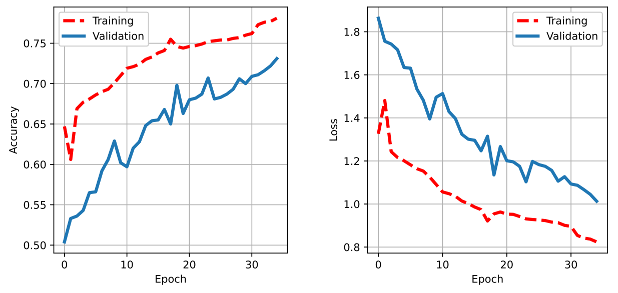


Fig. 12: Yolo detection results.

The results include a *mAP50* of 0.889, which indicates high precision in crack localization. Although pixel-wise segmentation presents moderate metrics, it proved suitable for practical applications, particularly considering the inherent complexity of image analysis.

The training loss curve of Model 3, illustrated in Fig. **??**, exhibits a sharp decrease during the initial epochs, followed by a gradual decline. This behavior indicates progressive learning and improved object detection performance through bounding box predictions. The reduction in the loss metric suggests the model's ability to capture relevant patterns during training. However, the validation loss presents noticeable fluctuations after epoch 30, suggesting challenges in generalizing to previously unseen data.



(a) (b)

Fig. 13: Training and validation metrics of Model 3: (a) Accuracy and (b) Loss.

Confusion matrix in Figure 5 illustrates the performance of Model 3 on detecting and locating defects (crack class) versus indicating their absence (background class).

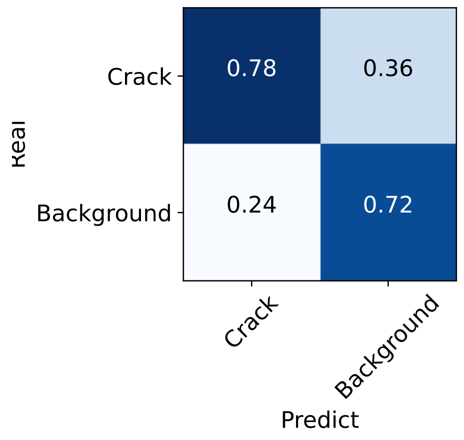


Fig. 14: Confusion matrix of Model 3.

D. Overall Comparison of Models

Table I presents results that may be used to provide a comparative analysis of the three evaluated models. Model 1 shows high values of accuracy, precision, and F1-score, indicating a well-balanced performance. The high precision observed in Model 1 reflects a low number of false positives, while its recall indicates strong capability in identifying positive cases. Model 3 also shows consistent results across all metrics and, although slightly lower than those of Model 1, it presents a trade-off between precision and recall. In contrast, Model 2 exhibits inferior performance, with lower accuracy and F1-score, suggesting issues in classification.

TABLE I: Model metrics for testing dataset.

Model	Accuracy	Precision	Recall	F1-score
Model 1	0.944	0.986	0.889	0.941
Model 2	0.663	0.613	0.637	0.625
Model 3	0.884	0.809	0.748	0.869

IV. Conclusion

This study evaluated three Deep Learning approaches for the automated detection of cracks in wind turbine blades, demonstrating that each model offers specific advantages depending on the application context. Model 1, a Sequential CNN, achieved the highest accuracy, proving robust for binary classification in standardized images. Model 2, which employs U-Net and CNN, exhibited greater sensitivity to subtle cracks, although at a significantly higher computational cost, due to the additional segmentation step. In contrast, despite successfully detecting cracks in most samples, Model 3 presented errors, especially where cracks are subtle or visually complex.

The findings indicate that none of the models was the absolute most suitable approach for DL-based inspection of wind turbine blades. Model selection should consider the model's performance concerning the intended task specifications. For critical applications such as continuous wind turbine monitoring, combining techniques may be a promising strategy.

Future work may focus on expanding model training with field-collected images, captured under varying environmental conditions, as well as implementing cross-validation, and conducting a systematic evaluation of model performance when processing low-quality images for large-scale inspections or embedded systems requiring fast processing. Furthermore, new research may explore transfer learning techniques and incorporate datasets from multiple sources to enhance the model's generalization.

ACKNOWLEDGMENT

We thank the Scientific Initiation Program of the Federal Rural University of the Semi-Arid Region (UFERSA), Call 21/2024, for supporting this work.

V. REFERENCES

[1] Y.-h. Liu, Y.-q. Zheng, Z.-f. Shao, T. Wei, T.-c. Cui, and R. Xu, "Defect detection of the surface of wind turbine blades combining attention mechanism," *Advanced Engineering Informatics*, vol. 59, p. 102292, 2024.

[2] H. C. Dantas, L. M. Morais, P. H. Bezerra, and R. C. Rego, "Concrete crack detection using embedded machine learning," in *2024 8th International Symposium on Instrumentation Systems, Circuits and Transducers (INSCIT)*. IEEE, 2024, pp. 1–6.

[3] X. Ye, L. Wang, C. Huang, and X. Luo, "Wind turbine blade defect detection with a semi-supervised deep learning framework," *Engineering Applications of Artificial Intelligence*, vol. 136, p. 108908, 2024.

[4] X. Ran, S. Zhang, H. Wang, and Z. Zhang, "An improved algorithm for wind turbine blade defect detection," *IEEE Access*, vol. 10, pp. 122 171–122 181, 2022.

[5] Z. Zhu, Y. Lei, G. Qi, Y. Chai, N. Mazur, Y. An, and X. Huang, "A review of the application of deep learning in intelligent fault diagnosis of rotating machinery," *Measurement*, vol. 206, p. 112346, 2023.

[6] L. Lv, Z. Yao, E. Wang, X. Ren, R. Pang, H. Wang, Y. Zhang, and H. Wu, "Efficient and accurate damage detector for wind turbine blade images," *IEEE Access*, vol. 10, pp. 123 378–123 386, 2022.

[7] H. Chon and J. Noh, "Review of wind turbine blade defect detection algorithm based on ai," *JMST Advances*, pp. 1–6, 2025.

[8] J. Wang, J. Liu, P. Tong, W. Yu, and Z. Wang, "A new class of fault recognition method for wind turbine systems based on deep learning," *International Journal of Adaptive Control and Signal Processing*, vol. 37, no. 12, pp. 3328–3342, 2023.

[9] M. Islam, A. Sarwar, A. Hamza, M. J. Afzal, R. Amir, S. N. Yadav, and A. Sarwar, "A deep learning fault diagnose method for turbine bearing: Digital twin mechanism," *European Journal of Applied Science, Engineering and Technology*, vol. 2, no. 2, pp. 378–387, 2024.

[10] A. Amin, A. Bibo, M. Panyam, and P. Tallapragada, "A bayesian deep learning framework for reliable fault diagnosis in wind turbine gearboxes under various operating conditions," *Wind Engineering*, vol. 48, no. 2, pp. 297–309, 2024.

[11] B. Altice, E. Nazario, M. Davis, M. Shekaramiz, T. K. Moon, and M. A. Masoum, "Anomaly detection on small wind turbine blades using deep learning algorithms," *Energies*, vol. 17, no. 5, p. 982, 2024.

[12] A. Bouzidi, L. Claeys, R. Randrianandraina, L. Rajaoarisoa, H. Wannous, and M. Sayed-Mouchaweh, "Deep learning for a customised head-mounted fault display system for the maintenance of wind turbines," in *2024 International Conference on Control, Automation and Diagnosis (ICCAD)*. IEEE, 2024, pp. 1–6.

[13] Y. Zhang, L. Wang, C. Huang, and X. Luo, "Wind turbine blade damage detection based on the improved yolov5 algorithm," in *2023 IEEE/IAS Industrial and Commercial Power System Asia (I&CPS Asia)*. IEEE, 2023, pp. 1353–1357.

[14] Z. Cao and Q. Wang, "Deep learning-based image recognition technology for wind turbine blade surface defects." *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 9, 2024.

[15] I. Gohar, W. K. Yew, A. Halimi, and J. See, "Review of state-of-the-art surface defect detection on wind turbine blades through aerial imagery: Challenges and recommendations," *Engineering Applications of Artificial Intelligence*, vol. 144, p. 109970, 2025.

[16] A. Shihavuddin, X. Chen, V. Fedorov, A. Nymark Christensen, N. Andre Brogaard Riis, K. Branner, A. Bjorholm Dahl, and R. Reinhold Paulsen, "Wind turbine surface damage detection by deep learning aided drone inspection analysis," *Energies*, vol. 12, no. 4, p. 676, 2019.

[17] M. Shekaramiz, "Small wind turbine blade dataset (cai-swtb)," Kaggle, 2024, [Online]. Available: https://www.kaggle.com/datasets/mohammadshekaramiz/small-wind-turbine-blade-dataset-cai-swtb.

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[19] 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.

[20] H. Zhu, X. Luo, Z. Zhai, J. Li, F. Li, and X. Zhu, "Unet-based yolo for underwater object detection," in *2024 IEEE First International Conference on Data Intelligence and Innovative Application (DIIA)*. IEEE, 2024, pp. 1–7.