Chapter 2
Interactive AI-based Framework for Automated Damage Augmentation on Limited SHM Datasets

Mohammad Mohammadzadeh and Alessandro Sabato

Abstract Structural Health Monitoring (SHM) plays a crucial role in ensuring the reliability and operational efficiency of large-scale engineering structures such as wind turbines. While extensive visual inspection datasets exist, they predominantly contain images of healthy wind turbine blades, significantly restricting the available data for supervised machine learning damage detection. Conventional data augmentation techniques typically include and increase irrelevant information and features, such as background and sky artifacts, thus reducing the detection accuracy and effectiveness of the machine learning model. To address these challenges, an interactive framework is introduced for damage augmentation, designed explicitly for limited SHM datasets. In this framework, users can upload available data and explicitly annotate damage using bounding boxes, enabling the automated extraction and characterization of damage such as cracks or erosion. Leveraging advanced generative artificial intelligence (AI) techniques, the framework synthesizes realistic and diverse damage instances by using Stable Diffusion fine-tuned with low-rank adapters. In this research, the proposed method is validated by augmenting real-world wind turbine blade imagery, demonstrating improved diversity and realism of the synthesized damages. Results highlight the method's potential to effectively enhance training datasets, paving the way for better-performing supervised models. This research contributes a practical and scalable augmentation solution, addressing critical data scarcity issues in SHM, and providing a robust foundation for future AI-driven damage detection systems.

Keywords Generative Artificial Intelligence · Structural Health Monitoring · Data Augmentation · Machine Learning · Wind Turbine Blades

Introduction and related work

The long service life of wind turbines depends on effective inspection and maintenance practices. Manual blade inspection is slow, costly, and risky. Missed or late detection of surface cracks can lead to downtime and structural failures. Therefore, automated visual detection is desirable for safer operations, lower costs per inspection, and improved maintenance scheduling [1].

Training reliable damage detectors is difficult with small labeled datasets. Data collection at scale is expensive, and many field images contain no cracks, creating severe class imbalance and limited coverage of crack appearances. Conventional augmentation strategies, such as rotation, rescaling, and flipping, improve feature invariance but cannot generate new crack morphologies or realistic surface representations [2]. More sophisticated approaches, including image-mixing and copy–paste approaches [3], extend data diversity but still fall short of bridging the gap between limited data and high-dimensional models. Recent augmentation pipelines have emerged that leverage generative models, such as Contrastive Language–Image Pretraining (CLIP) and Stable Diffusion (SD) [4, 5], to synthesize realistic cracks on real backgrounds. Generative Adversarial Network (GAN) pipelines have become a common choice, but they often require larger datasets and careful training to avoid model collapse and texture artifacts [6]. In contrast, diffusion models can be adapted with fewer parameters [7]. Low-Rank Adaptation (LoRA) makes this adaptation efficient and more stable in the small-data setting used here [8].

In this research, we introduce an interactive augmentation framework designed to address the problem of data scarcity in structural health monitoring of wind-turbine blades. The approach focuses on generating realistic crack appearances that

Mohammad Mohammadzadeh · Alessandro Sabato
Department of Mechanical and Industrial Engineering, University of Massachusetts Lowell, 1 University Avenue, Lowell, MA 01854
e-mail: Mo_moha@uml.edu; Alessandro_Sabato@uml.edu

Kristin B. Zimmerman, PhD (eds.), *Computer Vision and SLDV for Structural Dynamics, 2026, Vol. 6*, Conference Proceedings of the Society for Experimental Mechanics Series, https://doi.org/10.1007/978-8-743-81445-0_2

resemble those observed in practice, thereby enriching small datasets that would otherwise limit the reliability of supervised machine learning models. By combining user guidance with generative AI, the framework ensures that synthetic data remain consistent with the application domain and can be seamlessly integrated into training pipelines. This enables the creation of diverse, realistic datasets that improve model performance and help overcome one of the key barriers to deploying AI-based crack detection in real-world settings.

This work contributes: i) a practical, end-to-end, human-in-the-loop framework that couples LoRA-tuned diffusion with mask-constrained placement for blade cracks; ii) a direct comparison of alpha compositing and Poisson blending for inserting fine synthetic cracks on the surfaces of WTBs; and iii) a controlled, small-data evaluation across the five scenarios investigated in this study demonstrating how generative augmentation can substitute for real training data.

Methods

A data-efficient augmentation pipeline was developed to improve the detection of surface cracks on WTBs. Real crack regions were first extracted as 512×512 patches from annotated blade images. A SD v1.5 UNet model was then fine-tuned with low-rank adapters (LoRA) using these patches. Two ranks were trained, with rank 8 retained for its superior visual realism. During augmentation, a *per-sample loop* was used: one synthetic crack patch was generated, screened for realism in-app, and—if accepted—placed within user-defined feasible regions of a real background image. Two blending mechanisms were employed: i) alpha blending after thresholding to isolate crack pixels, and ii) gradient-domain Poisson blending via seamless cloning. Each placement accepted by the user produces a single bounding-box annotation. Any overlap with existing boxes triggered a resampling of the location. Synthetic data were restricted to the training split, while validation was kept real-only with a single "crack" class. Five training scenarios were constructed: baseline, adding 200 alpha-blended synthetic cracks, adding 200 Poisson-blended synthetic cracks, adding 200 Poisson-blended synthetic cracks on crack-free (CF) images plus original data, and 200 synthetic-only. YOLOv8m was trained with the provided configuration, with the best checkpoint selected automatically. The results of the five scenarios were evaluated in terms of precision, recall, and mean Average Precision (mAP) at 0.5 Intersection over Union (IoU).

LoRA fine-tuning of Stable Diffusion 1.5

SD v1.5 [9] was fine-tuned on curated 512×512 crack patches extracted from high-resolution images of WTBs. The goal was to train the model on the visual concept of a "hairline surface crack on a white blade". In SD, images are represented in a compressed latent space and are synthesized by iteratively denoising random noise under text guidance. For fine-tuning, each patch was paired with a single caption, and the text encoder remained the same except for the final portion of the training. Images were resized with Lanczos interpolation and normalized to the [-1, 1] range. Training was performed with batch size of 4 for 1,200 steps and a learning rate of 10^{-4}, with other settings left at library defaults. To adapt the model efficiently, LoRA [10] was applied to the cross-attention layers of the UNet. Rather than updating all weights, LoRA introduces small, task-specific updates to each affected linear layer, expressed as:

$$\Delta W = \frac{\alpha}{r} AB \tag{1}$$

Where ΔW is the low-rank update applied to the frozen weight matrix, A and B are trainable thin matrices of rank r, and α is a scaling factor that controls the contribution of the update. The effective weight is thus $W^{'}$=W+ΔW, with the base weights W kept frozen. The LoRA scaling was set to $\alpha = r$, giving the update unit gain. Two ranks were tested (i.e., $r = 4$ and $r = 8$), with $r = 8$ retained for producing crack appearances that were more realistic and closer to the training patches during in-app screening. Fine-tuning followed the library's standard diffusion training, where, at a random timestep t, Gaussian noise is added to the latent space representation z_t. The UNet is then trained to predict this noise by minimizing the mean-squared error (MSE) loss function:

$$\mathrm{L_{MSE}} = \mathrm{E}[\|\epsilon - \widehat{\epsilon}_\theta(\mathrm{z_t}, \mathrm{t}, \mathrm{c})\|_2^2] \tag{2}$$

where z_t denotes the noisy latent variable, c the text conditioning that guides the generation, ϵ the known injected noise, and $\widehat{\epsilon}_\theta(z_t, t, c)$the predicted noise from the UNet parameterized by θ. This loss measures the squared difference between the injected noise and the network's prediction, ensuring that the model learns to gradually denoise the latent representation in alignment with the conditioning. In practice, only the low-rank LoRA parameters are adjusted, guiding the denoising process to generate crack-like features consistent with the supervised patches while leaving the base model largely unchanged.

Per-Sample Generation, Screening, and Placement

Augmentation followed a one-by-one loop. A synthetic 512×512 crack patch was generated with the LoRA-tuned SD model and screened in-app for realism. Non-crack-like outputs were discarded, while accepted patches were placed within user-defined feasible regions on real background images from the 100-image training split. The 180-image validation split was kept real-only (no synthetic crack was added).

Feasible regions for placing synthetic cracks were defined as polygonal masks, saved as binary images $M \in \{0,1\}^{H \times W}$, where H and W denote the image height and width. In these masks, a pixel value of $M(x,y) = 1$ indicates that the pixel at location (x,y) is a valid placement area, while $M(x,y) = 0$ indicates that placement is not allowed. The complete set of admissible coordinates is therefore:

$$\Omega = \{(x,y)\,|\,M\,(x,\ y) = 1\}, \quad |\Omega| = \sum_{x=1}^{W}\sum_{y=1}^{H} M\,(x,y)\,. \tag{3}$$

Here, Ω is the set of all valid pixel locations, and $|\Omega|$ is its cardinality (the total number of allowed pixels). A placement coordinate (u,v), representing the pixel location of the patch center on the background image, was then drawn according to a uniform probability distribution $p((u,v))$:

$$p\,((u,v)) = \begin{cases} |\Omega|^{-1},\ (u,v)\ \in\ \Omega \\ 0,\ \ otherwise \end{cases} \tag{4}$$

In other words, each valid pixel in Ω has equal probability of being selected. The chosen coordinate (u,v) was used as the center of the synthetic crack patch. No rotation or rescaling was applied, and no minimum distance from the image edges was enforced. Before including the newly generated synthetic image, a human-in-the-loop visual check was performed to ensure that the new bounding box did not improperly overlap existing annotations and that the placement looked plausible on the surface. If an issue was observed, a new $(u,\ v)$ was sampled and the placement was redone. For each accepted placement, background image ID, random seed, mask identifier, $(u,\ v)$ coordinates, selected blending method, and resulting YOLO bounding box were logged.

Blending Method

Two placement mechanisms were employed. The first used alpha compositing, where only crack pixels from the generated patch were transferred to the background image. For each generated patch, a grayscale version was computed, and an interactive per-sample threshold was applied (without morphological operations) to produce a binary mask $\alpha(x,y) \in \{0,1\}$. Let $P(x,y)$ denote the generated crack patch (RGB), $B(x,y)$ the background image (RGB), and $R(x,y)$ the resulting composite image. The compositing operation is given by

$$R\,(x,y) =\ \alpha\,(x,\ y)\ P\,(x,y) +\ \ (1-\alpha\,(x,y))\ B\,(x,y)\,, \tag{5}$$

The second mechanism used gradient-domain blending (Poisson blending) via OpenCV's *seamlessClone* function with the NORMAL_CLONE option. In this case, a binary mask was derived from the crack patch by thresholding its grayscale at 240, then refined with a 3×3 morphological close–open sequence. Let Ω denote the masked region and $\partial\Omega$ its boundary. The blended image I inside Ω was obtained by solving the discrete Poisson objective with boundary conditions taken from the background B:

$$\min \sum_{(i,\ j)\in\Omega} \|\nabla \mathrm{I}\ (\mathrm{i},\ \mathrm{j}) -\ \nabla \mathrm{P}(\mathrm{i},\ \mathrm{j})\|_2^2\ s.t. I\,(i,j) = B\,(i,\ j)\ for\,(i,\ j)\ \in \partial\Omega \tag{6}$$

This optimization preserves the crack gradients from the patch P while ensuring the color and illumination at the boundary match the background B. In practice, the cloning center was set to the placement coordinates (u,v). If the placement was near the image border, masks and patches were cropped, and the OpenCV solver returned the final blended result R.

Experimental Design

Five training scenarios were evaluated to isolate how synthesis and blending on detection. In all cases, synthetic data was only used for training purposes, while validation (180 images) remained real-only. The baseline training split contained 100 real images. Across all scenarios, the same augmentation loop, mask-constrained placement, overlap-redo policy, label format, and YOLOv8m training configuration was applied. Only the composition of the training set varied.

- *Baseline (Exp. 1).* The original 100 real images were used without any synthetic additions, providing the reference detector performance.
- *Alpha-200 on train (Exp. 2).* 200 synthetic cracks were added to the 100 real images of Exp. 1 using the alpha composite method described in (2.6) and used for training. Cracks were generated and screened one at a time, then placed within feasible regions of a training image. Multiple cracks per image were allowed until 200 total placements were reached. Placement centers were sampled uniformly over the valid mask of the chosen background, and any visually problematic overlap with existing boxes triggered a resampling of the location. No rotation or scaling was applied, and the alpha masks were formed by per-sample thresholding without morphology (see Section 2.3).
- *Poisson-200 on train (Exp. 3).* 200 synthetic cracks were added to the 100 real training images using the Poisson blending formulation shown in (2.7) via seamlessClone (NORMAL_CLONE). As in Exp. 2, multiple cracks per image were allowed until 200 accepted placements were reached. Cloning centers, mask construction, and border handling followed the approach discussed in Section 2.3.
- *Poisson-200 on CF + original (Exp. 4).* A separate set of 200 crack-free images was each populated with one Poisson-blended synthetic crack. These images were then combined with the original 100 real training images. Feasible-region masks were enforced on the crack-free images as in the original training images.
- *Synthetic-only-200 (Exp 5).* A set of 200 crack-free images, each populated with one Poisson-blended synthetic crack, was constructed. No original real training images were used. This scenario tests the lower bound achievable when training relies solely on synthetic data, but evaluation remains on real images.

Table 1 Details of the five experiments considered in the study.

Scenario	Real train images	Synthetic cracks	Blending	Per-image rule
Exp. 1	100	0	-	-
Exp. 2	100	200 on train	Alpha (2.6)	Multiple
Exp. 3	100	200 on train	Poisson (2.7)	Multiple
Exp. 4	100	200 on CF + original	Poisson (2.7)	One
Exp. 5	0	200 on CF	Poisson (2.7)	One

All detectors were trained with Ultralytics YOLOv8m (pretrained yolov8m.pt) for 100 epochs using 1024×1024 images, a batch size of 16, and SGD with a base learning rate of 0.001. Checkpoints were selected by the trainer's best metric. While the training dataset differs by scenario, the validation and test sets were fixed and real-only.

Evaluation

Evaluation was performed on the held-out real validation set. Each predicted box $\hat{b}$ was greedily matched to at most one ground-truth box b based on the highest Intersection over Union (IoU). A match was accepted only if

$$IoU\left(\hat{b}, b\right) = \frac{area(\hat{b} \cap b)}{area(\hat{b} \cup b)} \geq 0.5 \tag{7}$$

With this rule, matched predictions were counted as true positives (TP), unmatched predictions as false positives (FP), and unmatched ground-truth boxes as false negatives (FN). Precision and recall at a given confidence threshold were defined as:

$$Precision = \frac{TP}{TP + FP}, \tag{8}$$

$$Recall = \frac{TP}{TP + FN}. \tag{9}$$

Predictions sorted by confidence and threshold were swept to form the precision—recall curve at the fixed IoU threshold of 0.5. Average Precision (AP) at 0.5 IoU, denoted $AP_{0.5}$ was computed as the area under this curve using the discretization provided by the library:

$$AP_{0.5} = \int_0^1 p\left(r\right)\, dr \;\approx \sum_{k=1}^{K} p_k \left(r_k - \, r_{k-1}\right), \tag{10}$$

where p_k and r_k are the precision and recall values at the *k-th* operating point. Since only one class was used, the mean Average Precision at 0.5 IoU equals $AP_{0.5}$,

Results

Data Summary and Patch Body

Training used 100 real images with 154 annotated cracks, while validation used 180 real images with 234 annotated cracks. All reported metrics were computed on the validation split to test robustness under a broader distribution of crack instances while keeping the training dataset deliberately small. For the real images, 154 curated 512×512 crack patches were extracted to adapt SD v1.5 with LoRA. Figure 1 shows the visual basis for synthesis and the effect of the LoRA rank. Figure 1(a) displays a 4×4 grid of real training patches, while Figures 1(b) and (c) display 4×4 grids of synthetic patches generated with LoRA rank 4 and rank 8, respectively, using identical prompts and sampler settings. Rank 8 was retained because its samples more closely matched the real training patches in shape, contrast, and background cleanliness during in-app screening. Figure 2 illustrates the effect of the two blending mechanisms. Alpha compositing transfers only the crack pixels after thresholding, which sometimes leaves faint edges around the patch. In contrast, Poisson blending better integrates the synthetic cracks with the background surface, producing smoother transitions in shading and illumination.

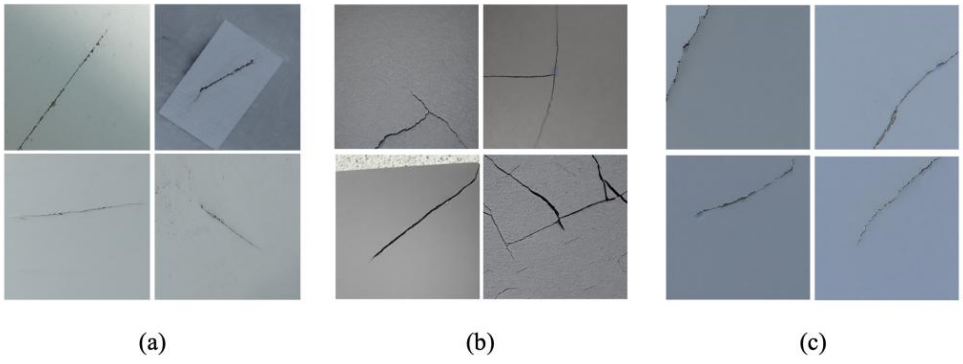

Fig. 1 Patch grids. (a) Real 4×4 training patches. (b) Synthetic 4×4 patches from SD+LoRA rank 4. (c) Synthetic 4×4 patches from SD+LoRA rank 8 (selected for subsequent experiments).

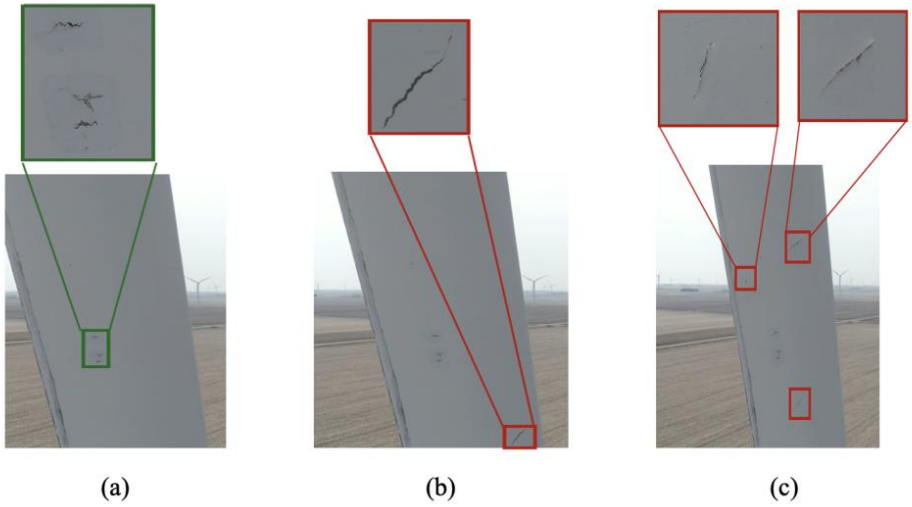

Fig. 2 Side-by-side comparison of blending methods. (a) Original training image with real cracks. (b) Example with alpha-blended synthetic cracks. (c) Example with Poisson-blended synthetic cracks. (The boxes highlight example cracks: green for real and red for synthetic. The specific synthetic instances differ across panels).

End-to-end app snapshot

The augmentation was carried out in an end-to-end app that integrates generation, placement, and auto-annotation into a single workflow (Figure 3). The LoRA-adapted diffusion model produces one crack at a time, which is screened in-app for realism. If accepted, a training image is selected, a feasible region is chosen, and the crack is placed at a sampled location. Alpha or Poisson blending is applied, and a YOLO-format box is written automatically. If a placement overlaps an existing box or looks implausible, a new location is sampled immediately.

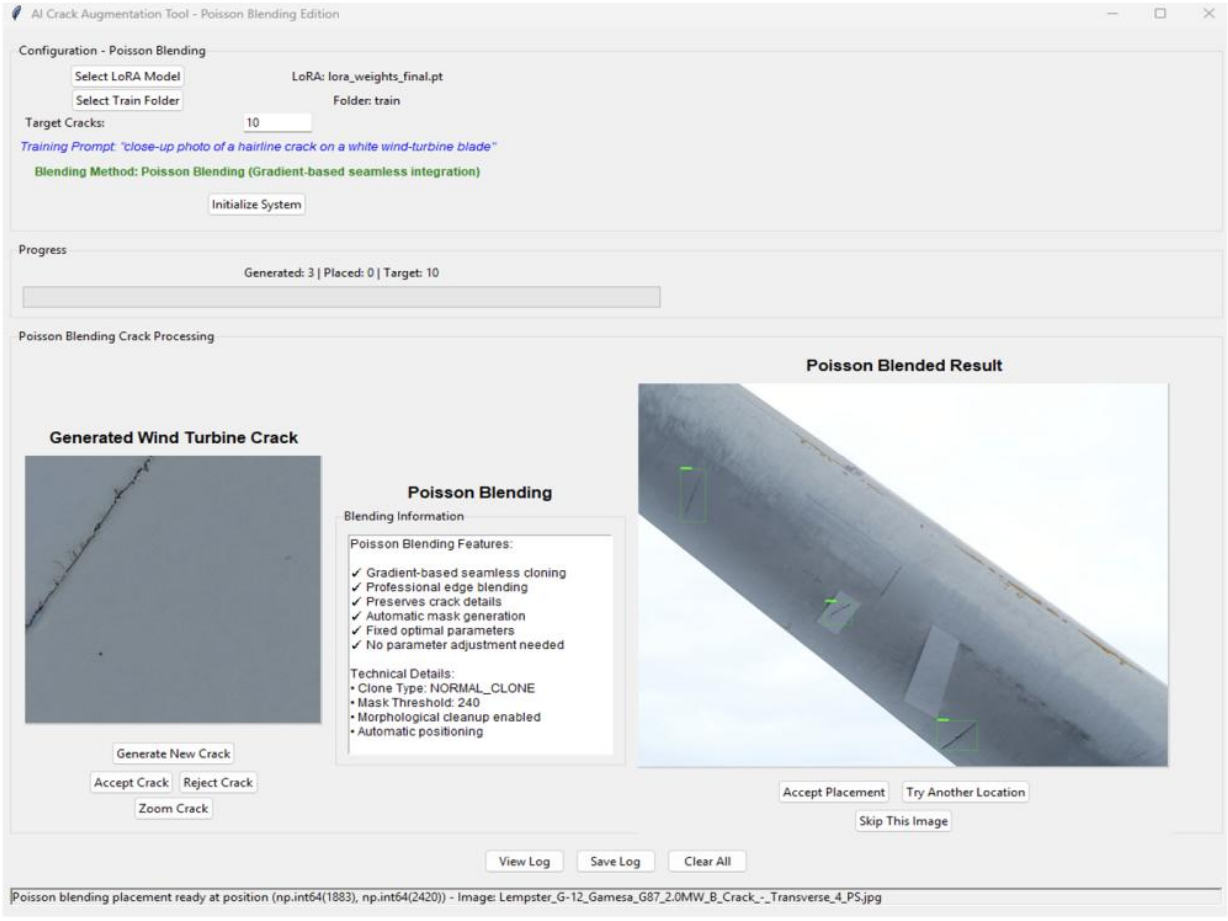

Fig. 3 App snapshot. The interface supports per-sample crack generation and screening, mask-constrained placement and automatic YOLO annotation and logging—all within one window.

Quantitative Performance

All metrics identified in Section 2.5 were computed on the validation split (i.e., 180 images, 234 cracks). The same YOLOv8m configuration was used across all scenarios studied and only the composition of the training dataset was different (i.e., 100 images, 154 cracks).

Table 2 Comparison of experimental scenarios on detection metrics.

Scenario	mAP	Recall	Precision
Exp. 1 Baseline	48.8	46.6	61.6
Exp. 2 Apha-200 on train	46.8	45.3	53.5
Exp. 3 Poisson-200 on train	**59.4**	**55.5**	**70.9**
Exp. 4 Poisson-200 on CF + original	52.4	53.6	65.0
Exp. 5 Synthetic-only-200	30.3	33.3	42.5

The Poisson-200 on-train scenario *(Exp. 3)* achieved the best results with mAP = 59.4, Recall = 55.5, and Precision = 70.9. Relative to the baseline *(Exp. 1)*, absolute gains of +10.6 in mAP, +8.9 in Recall, and +9.3 in Precision were observed. The Poisson-200 crack-free + original scenario *(Exp. 4)* also improved recall and precision over baseline (53.6 and 65, respectively) with a smaller mAP increase (54.4), suggesting that inserting synthetic images into the original training dataset is most effective under the given constraints. The Alpha-200 on-train scenario *(Exp. 2)* underperformed baseline across all metrics (i.e., 46.8 / 45.3 / 53.5), consistent with qualitative artifacts observed with alpha compositing on varying backgrounds.

The synthetic-only scenario *(Exp. 5)* was included to quantify how much of the baseline can be recovered using *only* synthetic data. With 200 Poisson-blended images, it reached 30.3 / 33.3 / 42.5. This corresponds to about 62% of baseline mAP (30.3 vs. 48.8), 72% of baseline recall (33.3 vs. 46.6), and 69% of baseline precision (42.5 vs. 61.6). Thus, while a small synthetic body of synthetic images alone was insufficient to match the real-data baseline, it recovered a substantial fraction of the baseline performance under the same evaluation on real images, establishing a practical lower bound for zero-real training in this setting. Figure 4 shows sample of predicted cracks on the wind turbine blade.

Fig. 4 Samples of predicted cracks on wind turbine using YOLOv8.

Conclusion and Future Work

A data-efficient, end-to-end augmentation framework and app for surface crack detection on wind turbine blades is presented in this work. SD v1.5 was adapted with LoRA (rank 8) using curated 512×512 crack patches. Augmentation proceeded one sample at a time with in-app realism screening, mask-constrained placement on real training images, and automatic YOLO annotations. Two blending strategies were tested: alpha compositing, which transfers crack pixels after thresholding, and Poisson blending, which preserves shading and illumination on white gel-coat surfaces. Results of the tests performed showed how Poisson blending delivered the most reliable improvements over baseline, while the synthetic-only case, though lowest, recovered a substantial fraction of baseline performance, establishing a practical lower bound for zero-real training.

This work establishes an end-to-end, human-in-the-loop app that unifies generation, placement, blending, and annotation into a YOLO-ready workflow. Moreover, it is the first controlled comparison of alpha and Poisson blending for fine crack insertion under strict data limits; and an evaluation of synthetic–real mixing strategies, showing that Poisson-based insertions on real training images yield the largest gains. Future work will automate realism scoring, introduce distribution-aware placement, extend labels to segmentation masks, broaden detector baselines and metrics, and ablate LoRA hyperparameters. This study shows that generative augmentation, paired with human oversight, can significantly improve crack detection in low-data situations.

References

1. Ciang, C. C., Lee, J. R., & Bang, H. J. (2008). Structural health monitoring for a wind turbine system: a review of damage detection methods. *Measurement science and technology*, *19*(12), 122001.
2. Sengupta, P., Mehta, A., & Rana, P. S. (2023, July). Enhancing performance of deep learning models with a novel data augmentation approach. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1-7). IEEE.

3. Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T. Y., Cubuk, E. D., ... & Zoph, B. (2021). Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2918-2928).
4. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, *1*(2), 3.
5. Song, J., Meng, C., & Ermon, S. (2020). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.
6. Luleci, F., Catbas, F. N., & Avci, O. (2023). Generative adversarial networks for labeled acceleration data augmentation for structural damage detection. *Journal of Civil Structural Health Monitoring*, *13*(1), 181-198.
7. Fang, H., Han, B., Zhang, S., Zhou, S., Hu, C., & Ye, W. M. (2024). Data augmentation for object detection via controllable diffusion models. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 1257-1266).
8. Zhang, Z., Zhou, Y., & Ma, J. (2025). Defect-LoRA: Controllable defect data augmentation based on low-rank adaptation for surface defect recognition under limited data. *Expert Systems with Applications*, *268*, 126251.
9. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10684-10695).
10. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). Lora: Low-rank adaptation of large language models. *ICLR*, *1*(2), 3.