Journal of
Experimental and Theoretical Analyses

MDPI

Article

A Data-Driven Approach for Automatic Aircraft Engine Borescope Inspection Defect Detection Using Computer Vision and Deep Learning

Thibaud Schaller [1], Jun Li [2,*] and Karl W. Jenkins [2]

[1] ISAE-Supméca—Institute of Mechanics of Paris, 3 Rue Fernand Hainaut, 93400 Paris, France; thibaud.schaller@isae-supmeca.fr
[2] School of Aerospace, Transport and Manufacturing (SATM), Cranfield University, Cranfield MK43 0AL, Bedfordshire, UK; k.w.jenkins@cranfield.ac.uk
* Correspondence: jun.li@cranfield.ac.uk

Abstract: Regular aircraft engine inspections play a crucial role in aviation safety. However, traditional inspections are often performed manually, relying heavily on the judgment and experience of operators. This paper presents a data-driven deep learning framework capable of automatically detecting defects on reactor blades. Specifically, this study develops Deep Neural Network models to detect defects in borescope images using various datasets, based on Computer Vision and YOLOv8n object detection techniques. Firstly, reactor blade images are collected from public resources and then annotated and preprocessed into different groups based on Computer Vision techniques. In addition, synthetic images are generated using Deep Convolutional Generative Adversarial Networks and a manual data augmentation approach by randomly pasting defects onto reactor blade images. YOLOv8n-based deep learning models are subsequently fine-tuned and trained on these dataset groups. The results indicate that the model trained on wide-shot blade images performs better overall at detecting defects on blades compared to the model trained on zoomed-in images. The comparison of multiple models' results reveals inherent uncertainties in model performance that while some models trained on data enhanced by Computer Vision techniques may appear more reliable in some types of defect detection, the relationship between these techniques and subsequent results cannot be generalized. The impact of epochs and optimizers on the model's performance indicates that incorporating rotated images and selecting an appropriate optimizer are key factors for effective model training. Furthermore, models trained solely on artificially generated images from collages perform poorly at detecting defects in real images. A potential solution is to train the model on both synthetic and real images. Future work will focus on improving the framework's performance and conducting a more comprehensive uncertainty analysis by utilizing larger and more diverse datasets, supported by enhanced computational power.

Keywords: blade images; data augmentation; fine-tuning; mAP50; uncertainty and sensitivity analysis; model and evaluation uncertainties; YOLO

check for updates

Academic Editor: Marco Rossi

Received: 8 November 2024
Revised: 7 January 2025
Accepted: 8 January 2025
Published: 5 February 2025

Citation: Schaller, T.; Li, J.; Jenkins, K.W. A Data-Driven Approach for Automatic Aircraft Engine Borescope Inspection Defect Detection Using Computer Vision and Deep Learning. *J. Exp. Theor. Anal.* **2025**, *3*, 4. https://doi.org/10.3390/jeta3010004

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

It is estimated that engine malfunctions accounted for nearly 10.8% of air accidents between 2013 and 2021 [1]. Regular aircraft engine checks and maintenance are crucial for preventing accidents by ensuring aircrafts meet safety and international standards. Engine inspections vary in type. Routine inspections, including type A (monthly) and

type B (every three months), ensure regular monitoring of the engine's condition. Total inspections require dismantling the engine, with type C conducted on average every 12 to 18 months and type D every 3 years, while emergency inspections are performed after unusual events, such as bird strikes or engine failures. Various technologies, including visual inspection, borescope inspection, and non-destructive testing, are utilized for aircraft engine inspections, particularly for detecting blade damage. However, these inspections often rely on operators' judgment, which is a time-consuming and experience-dependent process [2].

Recent studies [3–8] have proposed deep learning methods such as Convolutional Neural Networks (CNNs), R-CNN, and customized YOLO methods, showcasing the potential of the technique in automated aircraft engine defect detection. These methods analyze images, such as borescope images captured by sensory devices, to detect defects automatically. However, the sensitivity and reliability of these defect detection models influenced by the choice, variety, and quantity of datasets [9–11] remain underexplored [12].

This study aims to automate aircraft engine defect detection on compressor blades (specifically, tears, dents, and nicks) to reduce human errors in routine borescope inspections, by applying an integrated Computer Vision and deep learning approach with well-designed dataset groups. Additionally, this research focuses on the models' performance optimization and sensitivity study by applying image transformations and synthetic image generation techniques. The results are also compared and analyzed to identify effective techniques.

The remainder of this paper is organized as follows: Section 2 provides an overview of the current work in the area of aircraft engine defect detection automation; Section 3 introduces the data-driven methodology, with a particular focus on the framework development using Computer Vision and deep learning; Section 4 presents the modeling process, model results, and findings; Section 5 discusses the findings and the potential improvement of the research; and Section 6 concludes this work.

2. Literature Review

Three-dimensional scanning inspects aircraft engine components using three-dimensional scanning technologies such as laser scanning, structured light scanning, and photogrammetry. This method effectively detects defects on reactor blades, identifying both the location and nature of a defect accurately [13]. However, this method requires the engines to be dismantled, limiting its use to type C and D maintenance checks. It is also unsuitable for routine inspections, which require quick and non-invasive detection. Moreover, the scanners used by the method are costly, and the examination of blades is inefficient. While reactor blade damage can occur on both the edges and surfaces, most detected defects are located on the leading and trailing edges [14]. Therefore, techniques such as the Canny edge detector [15] are primarily applied to detect edge defects. In this process, blades are removed from the reactor and photographed against a clear, uniform background. The Canny edge detector is then used to identify a series of edge points in the image where the light background intersects with the darker blade, and algorithms such as the Suzuki algorithm are employed to identify areas with significant angle variations between the segments connecting each point. This method still requires dismantling the reactor blades, making it unsuitable for type A or B inspections. Unlike 3D scanning, this technology is less expensive, though remains inefficient.

Routine inspections require to be conducted quickly without dismantling the engines. Real-time image recognition is ideal for detecting defects in borescope images, allowing cost-effective, real-time blade analysis without dismantling the reactor. Deep learning object detection models such as YOLO, MobileNet, and R-CNN have been explored for routine

inspection recently [6,7]. Svensen et al. [3] applied Deep CNNs to analyze images from GE Aviation's borescope inspections of large turbofan engines for routine monitoring, alerts, and maintenance. However, this research primarily focuses on classifying images of various engine parts rather than specifically on defect detection. An ensembled deep learning method [16] based on multiple Convolutional Neural Network models is developed for automated identification of surface defects in aircraft propeller blades. Unfortunately, the method only identifies the presence of defects and improves the detection rate but does not accurately locate the defects. Wong et al. [5] proposed an automatic workflow based on the Mask R-CNN framework for detecting damage present on rotor blades using borescope videos. This workflow mainly focuses on tracking blades across a video and computing statistics on the types and extent of damage found on rotor blades while overlooking the impact of data size, quality, and variety on model performance and sensitivity. Upadhyay et al. introduced a deep learning framework with a customized U-Net architecture to detect the defects on high-pressure compressor blades. The framework addresses the data imbalance, defect size, and data availability issues in part by applying different model performance metrics and generating synthetic images, while the model's reliability and generalization capability influenced by factors such as data size, image preprocessing, and model hyperparameters like optimizer choice are not studied. Moreover, a multiscale Fully Convolutional Network that combines networks trained at various scales of feature maps is proposed for generic segmentation of engine distress [4]. The reliability and generalization of the model are not considered.

While other studies using deep learning [7,8,17–19] are available, most studies related to image recognition models typically aim for model development, model fine-tuning, or the comparison of models trained on the same dataset. A significant challenge in automated engine defect detection is the lack of blade images in the public domain for model training, which remains unaddressed. Moreover, the impact of data size, quality, and variety on model performance and sensitivity is also not investigated. This study aims to focus on data augmentation based on image processing and synthetic data generation methods and evaluating the effectiveness of these techniques. It also examines the training behavior of defect detection models across different model hyperparameters and data cases.

3. Methodology

This study works on a supervised classification learning problem, in this case, reactor blade images and output labels, i.e., the coordinates of defects present in each image. An aircraft engine borescope inspection automation framework based on Computer Vision and deep learning is developed, as shown in Figure 1.

The framework consists of several phases. First, images are collected or generated and annotated to build a robust dataset for training. These images are then transformed using techniques such as grayscale conversion and contrast adjustments, followed by augmentation through rotations and mirroring to enhance features for effective defect detection. Next, borescope defect detection models based on YOLOv8n [7,20] are trained with parameters fine-tuned to optimize performance. Finally, the models are evaluated using defined metrics and analyses are conducted to assess model uncertainty and sensitivity.

Each Neural Network model workout is conditioned on hyperparameters. In this study, optimizers (e.g., stochastic gradient descent and Adam as detailed in Appendix B) and training parameters such as epochs are selected based on the model's training context to achieve optimal efficiency. To evaluate model performance, this study uses a confusion matrix and its derivative metrics, including precision, accuracy, recall, mAP50, and mAP50-95 (see Appendix C).

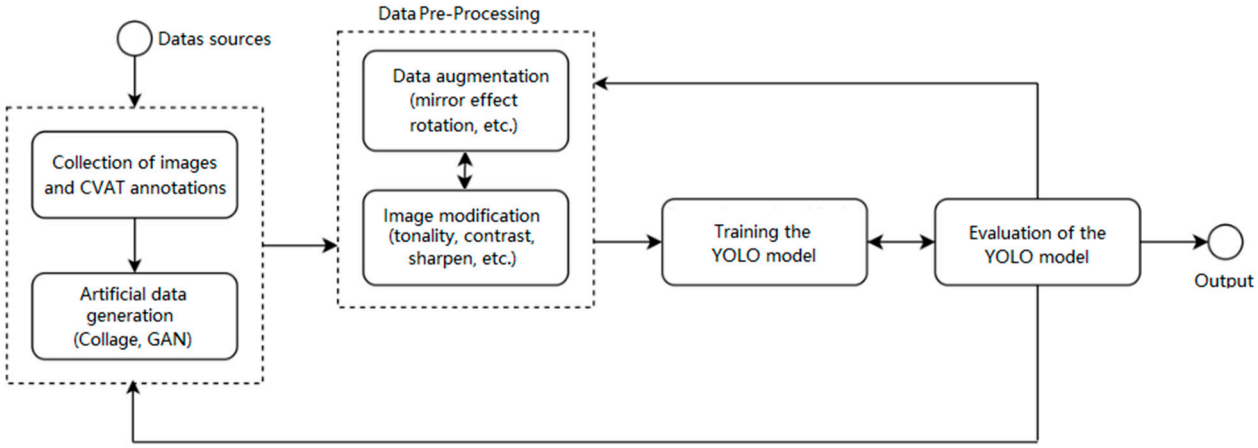


Figure 1. An aircraft engine borescope inspection automation framework based on Computer Vision and deep learning.

4. Modeling and Results

The initial images of compressor blades with defects used in this study are extracted from YouTube video frames [9–11] depicting real borescope inspection cases. These images are then annotated manually using CVAT (Computer Vision Annotation Tool) [21], with examples shown in Figures 2 and 3. The online tool creates annotations as metadata that are stored separately from the images. The dataset for subsequent model training consists of image files and corresponding text files, each containing bounding box coordinates, dimensions, and a class label that specifies the type of object in the image. During annotation, defects were framed, labeled, and stored in YOLO format, with each label including the defect class, bounding box center coordinates, and dimensions.

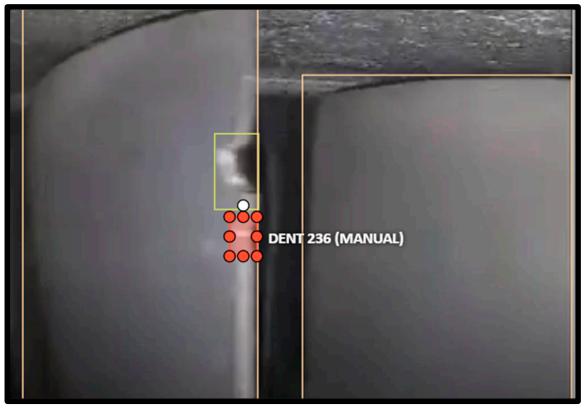


Figure 2. Supervision of objects studied with CVAT.



Figure 3. Different objects and defects studied.

As shown in Figure 3, this study examines various types of objects, including blades, nicks/notches (small V-shaped cuts with a regular surface, often appearing on the blade edge or surface), dents/bumps (blade deformations from impacts or undulations), and tears (material separations that may result in material loss). A key challenge in this phase is to create a dataset with sufficient size, quality, and variety. To address this, data augmentation techniques are applied to enhance the dataset.

Next, object detection models are developed based on fine-tuning, to maximize and compare their accuracy and efficiency in defect detection, using various datasets obtained from the public domain, along with data preprocessing and augmentation techniques. The YOLOv8n model [4] is chosen for its balance of reliability and efficiency, featuring a simpler and lower dimensional Neural Network architecture compared to that of other versions, with parameters configured as outlined in Appendix A. The dataset is split into training and testing datasets according to size, where the optimal proportion for the testing dataset tends to decrease as the overall dataset size increases. Both underfitting and overfitting are carefully managed during model training. Unless otherwise specified, an automated optimization approach is applied, with AdamW for fewer than 10,000 iterations and SGD beyond that. The number of iterations is defined as follows:

$$Nb_iter = \frac{Nb_epoch \times Nb_datas}{Nb_batch}$$

where *Nb_iter*, *Nb_epoch*, *Nb_datas*, and *Nb_batch* are the number of iterations, epochs, images in the dataset, and images processed in a batch (16 by default), respectively.

The datasets created and models developed from them are summarized in Table 1.

Table 1. Datasets for the aircraft engine borescope inspection automation models.

Model	Image Dataset	Environment and Training
Model1_100	Dataset 1: 170 images containing blades with defects and blades without defects.	Google Colab with data stored on Google Drive. (70% Train, 30% Val)
Model2_100	Dataset 2: 320 images containing blades with defects and blades without defects by zooming in on the defects.	Google Colab with data stored on Google Drive. (70% Train, 30% Val)
Model3_50v1, Model3_50v2, Model3_100, Model3_2x50	Dataset 3: Combination of Datasets 1 and 2 (490 images).	Local computer. (70% Train, 30% Val)
Model4	Dataset 4: Based on Dataset 3 but at gray level with contrast (factor 1.5) and sharpening (factor 0.2) update (490 images).	Local computer. (70% Train, 30% Val)
Model5	Dataset 5: Based on Dataset 4 with average brightness equalization (127) and seven gray levels setting (490 images).	Local computer. (70% Train, 30% Val)
Model6	Dataset 6: Based on Dataset 3 with grayscale setting (alpha = 1.0) (490 images).	Local computer. (70% Train, 30% Val)
Model7	Dataset 7: Dataset 4 + Dataset 4 rotation at 90° (980 images).	Local computer. (70% Train, 30% Val)
Model8	Dataset 8: 490 images with Dataset 4 ($^1/_3$), Dataset 5 ($^1/_3$), and Dataset 6 ($^1/_3$).	Local computer. (70% Train, 30% Val)

Table 1. *Cont.*

Model	Image Dataset	Environment and Training
Model9 Collage model	Dataset 9: 1600 images with 800 artificial surface defects (20 defects and 20 backgrounds), and 800 artificial edge defects (20 defects and 20 backgrounds).	Local computer. (50% Train, 50% Val)
DCGAN models, Model10	Dataset 10: Derivative of Dataset 4 with 300 initial images.	Local computer.

Model1_100 and Model2_100 are trained for up to 100 epochs on two small datasets, Dataset 1 and Dataset 2, respectively, with examples and results presented in Figures 4–8.

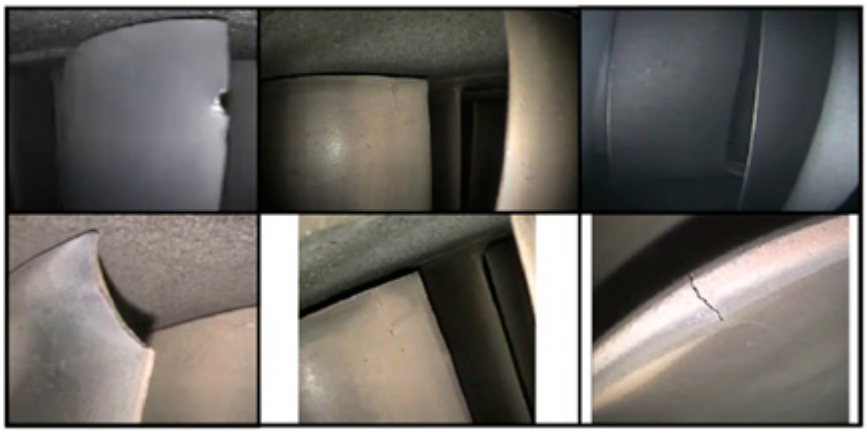


Figure 4. Example images from Dataset 1 with wide view (**top**) and Dataset 2 with zoom-in view (**bottom**).

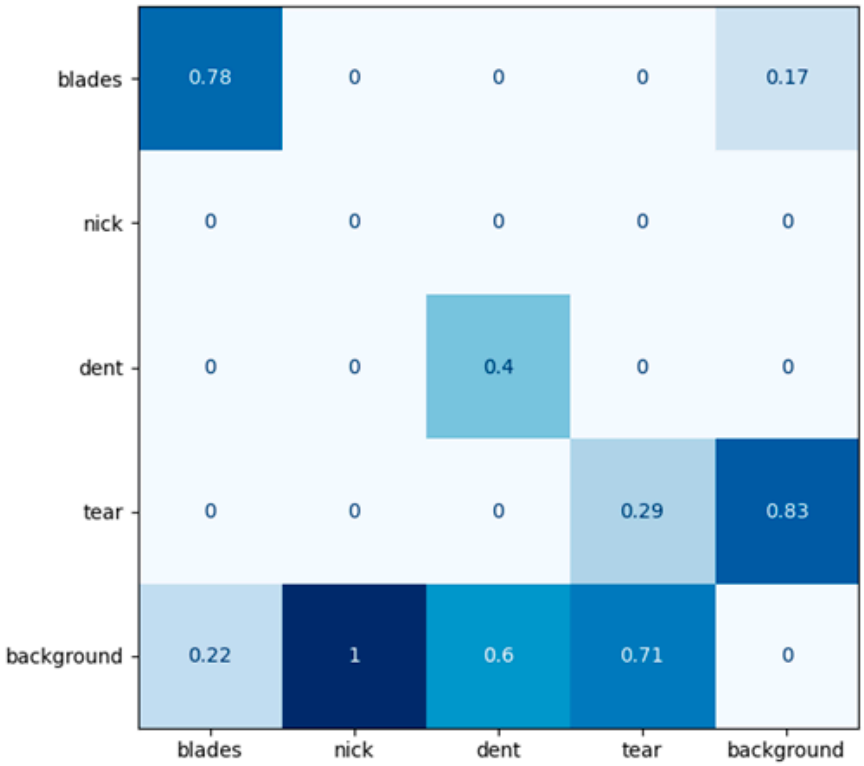


Figure 5. Normalized confusion matrix of Model1_100.

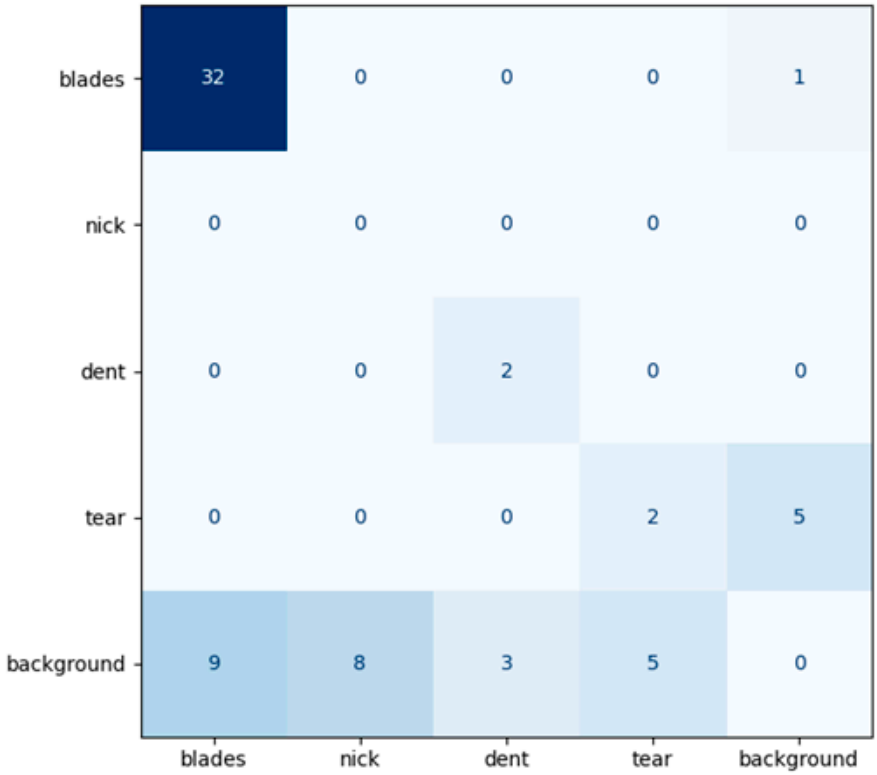


Figure 6. Confusion matrix of Model1_100.

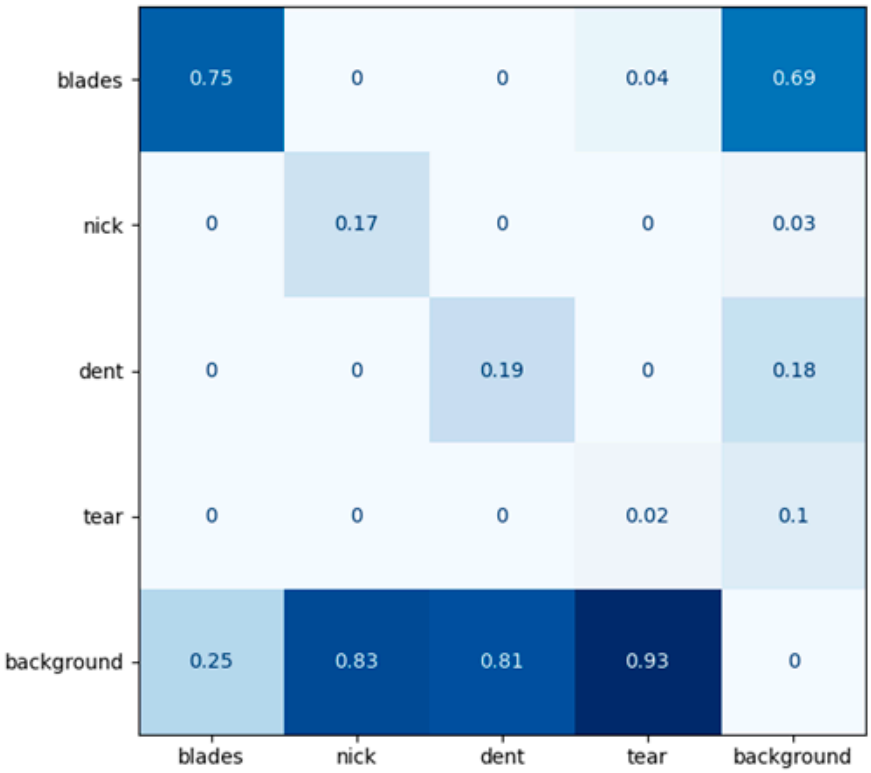


Figure 7. Normalized confusion matrix of Model2_100.

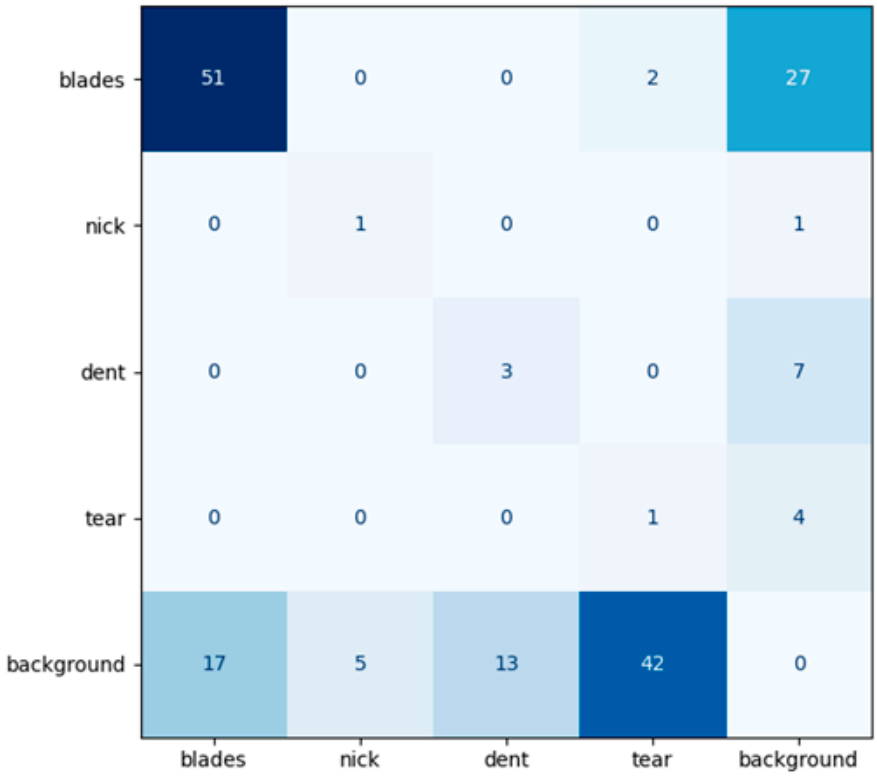


Figure 8. Confusion matrix of Model2_100.

The results show that Model1_100 performs better at detecting defects on blades, with a recall of 78% compared to that of 75% for Model2_100, likely due to the presence of wide-shot blade images in Dataset 1. Model1_100 also outperforms Model2_100 in detecting dents (40% versus 17%) and tears (29% versus 2%). On the other hand, Model2_100 is more effective at detecting nicks, achieving a 17% recall compared to a 0% one for Model1_100, which can be attributed to the close-up images and higher proportion of notches in Dataset 2. However, as the datasets are small, these models may lack reliability. For instance, 17% of the bumps in Dataset 2 come only from a single image. The training on Google Colab with the default configuration runs at 13-14 iterations per second, taking about 3 h for 100 epochs per dataset. To avoid interruptions from disconnections and epoch limits, the training process is divided into phases with a fixed number of epochs each. Next, this study seeks to enhance the datasets by adding more defect images across various categories.

Two models are then trained for 50 epochs with the same hyperparameters and training settings (see Appendix A) on Dataset 3, a combination of Datasets 1 and 2. The training is conducted on a standard computer (Intel(R) Core(TM) i7-14700 2.10 GHz Processor, 32.0 GB RAM, Windows 10 Enterprise 64-bit). The results, as shown in Figures 9–11, highlight the differences in the confusion matrices and precision/recall curves. For instance, 45% of tears are correctly detected by the v1 model, with 35% by the v2 model.

The results also indicate that training variability must be considered when comparing models, as factors like random weight initialization, stochastic optimization, non-deterministic hardware operations, and potential underfitting may contribute to these differences. Subsequently, two new models are trained using the same hyperparameters and dataset, with Model3_100 trained for 100 epochs, and Model3_2x50 trained for 100 epochs with an intermediate checkpoint. The results in Figures 12–14 show that Model3_100 achieves slightly better defect detection performance (0.509 mAP) compared to that of Model3_2x50 (0.503 mAP). On average, 29% of defects are correctly detected by Model3_100, with 27% by Model3_2x50.

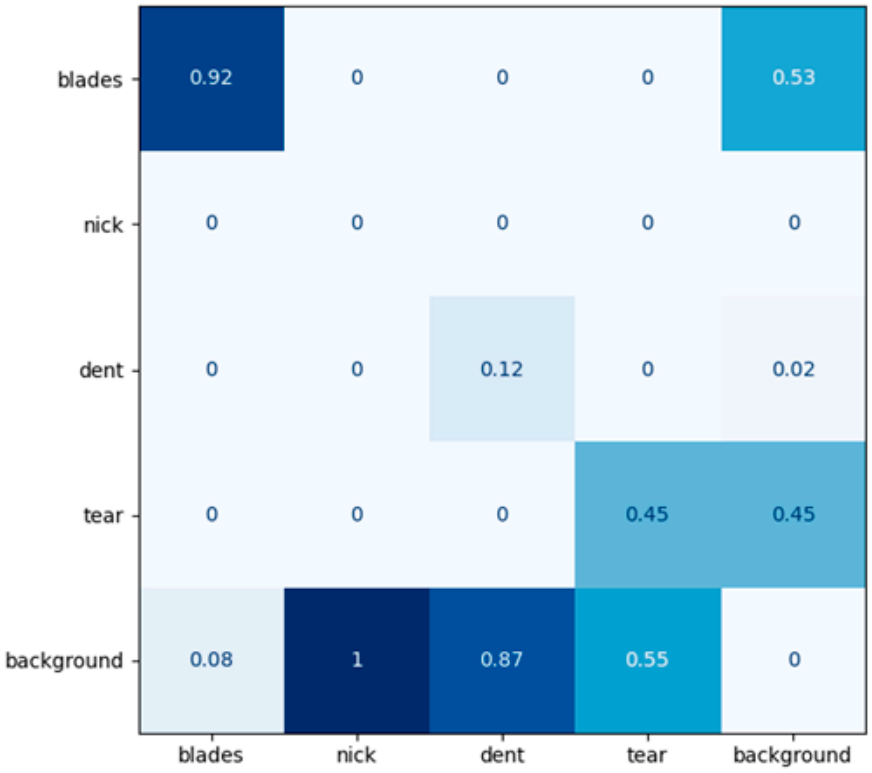


Figure 9. Normalized confusion matrix of Model3_50v1.

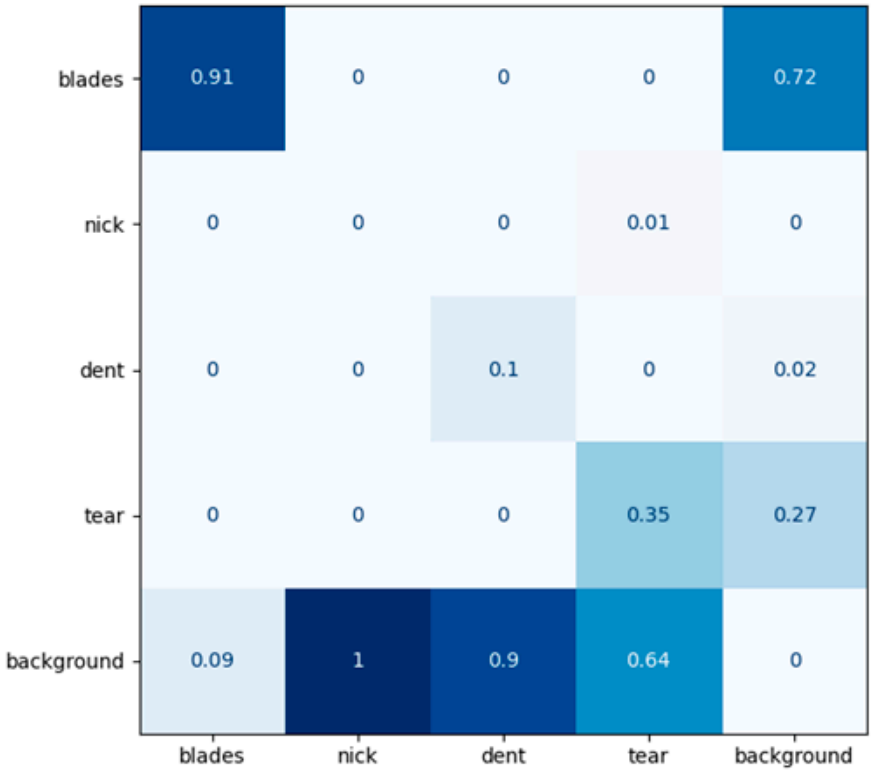


Figure 10. Normalized confusion matrix of Model3_50v2.

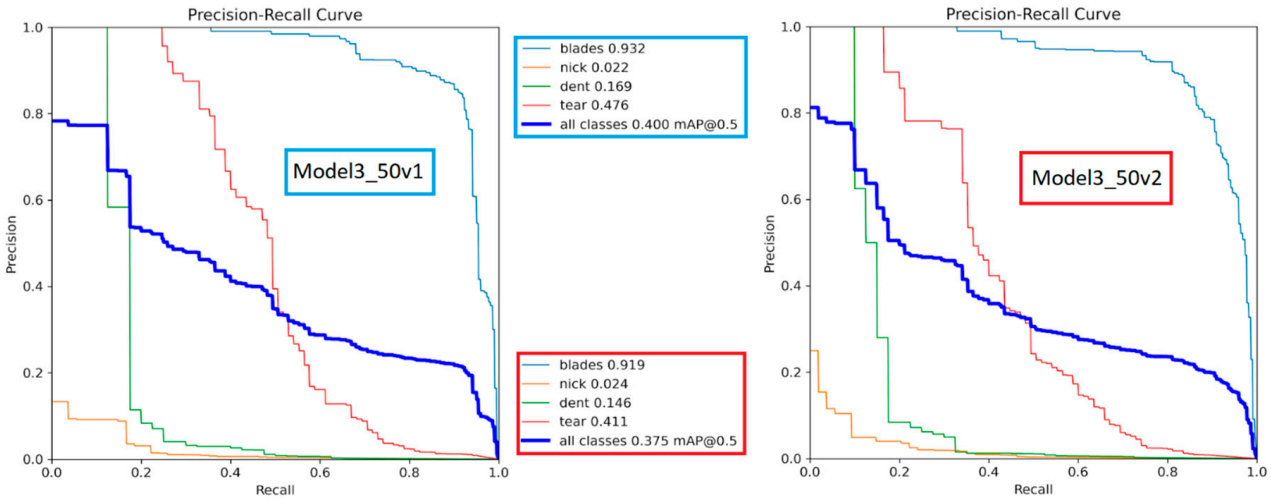


Figure 11. Precision/recall curves for Model3_50v1 and Model3_50v2.

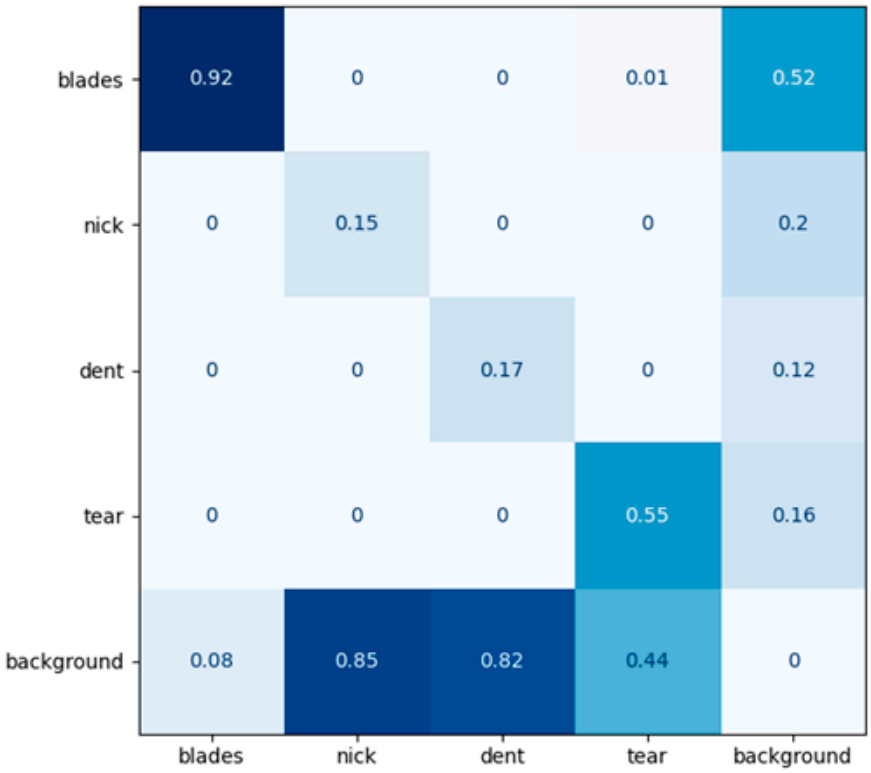


Figure 12. Normalized confusion matrix of Model3_100.

To help reveal details in the lighter areas of the image, contrast enhancement and sharpening are applied to Dataset 3, resulting in Dataset 4. Dataset 5 is generated from Dataset 4 based on gray levels and the average brightness equalization technique to reduce detail in low-contrast background areas while enhancing detail in high-contrast regions, highlighting edge defects and surface variations on the blades. Brightness thresholds are set so that pixels within a specified range are reassigned to enhance detail, optimized for images with a given average brightness. Dark images may obscure details in shadows, while bright images may lose detail in highlights. Thus, equalizing brightness beforehand is crucial for preserving maximum detail. Dataset 6 is targeted to remove unnecessary information, mainly colors, to reduce model variance and lower the risk of overfitting to the specific colors in the dataset. For example, a model trained on images of reactor blades under yellowish lighting might struggle to detect blades under bluish lighting. Reactor

blades are typically gray, but lighting can create variations in tone. The training image samples are shown in Figure 15, and the model testing results based on Datasets 3–6 for various epochs are generated and compared in Table 2.

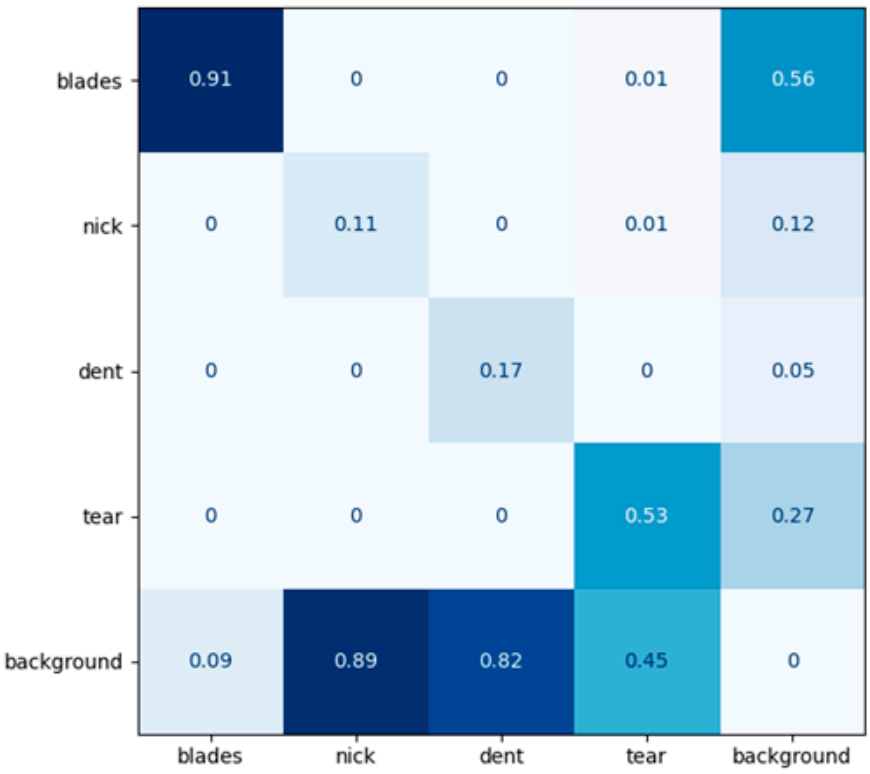


Figure 13. Normalized confusion matrix of Model3_2x50.

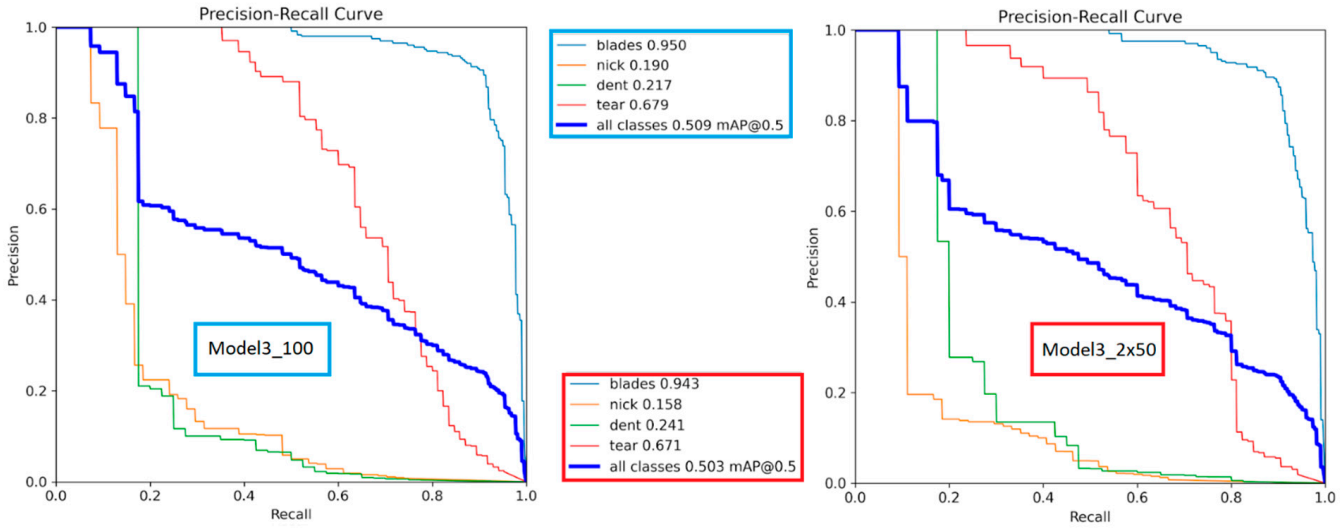


Figure 14. Precision/recall curves for Model3_100 and Model3_2x50.

The first two tables show that Model 3 based on original images performs better, with higher mAP50 and mAP50-95 at each epoch. However, the confusion matrix table exhibits more nuanced results depending on the type of defects detected, in which Model 6 shows better performance in detecting blades and nicks while Model 4 shows better performance in detecting tears and Model 3 shows better performance in detecting dents. These results should be interpreted cautiously, as the limited datasets may not fully represent the models' generalization capabilities. The models are subsequently tested on new cases, including

images that differ significantly from the training datasets, such as those in row 3 of Figure 16, where cat images help highlight potential false positives. While some models may appear more reliable according to metrics, for instance, Model3_3x50 achieves higher mAP50 and mAP50-95 scores, the tests reveal inherent uncertainties in model performance. Notably, defects identified by a model with lower performance may go undetected by a model that is previously considered to perform better. The detection capability of each model can vary depending on factors such as defect shape, image brightness, and color proximity within the dataset.

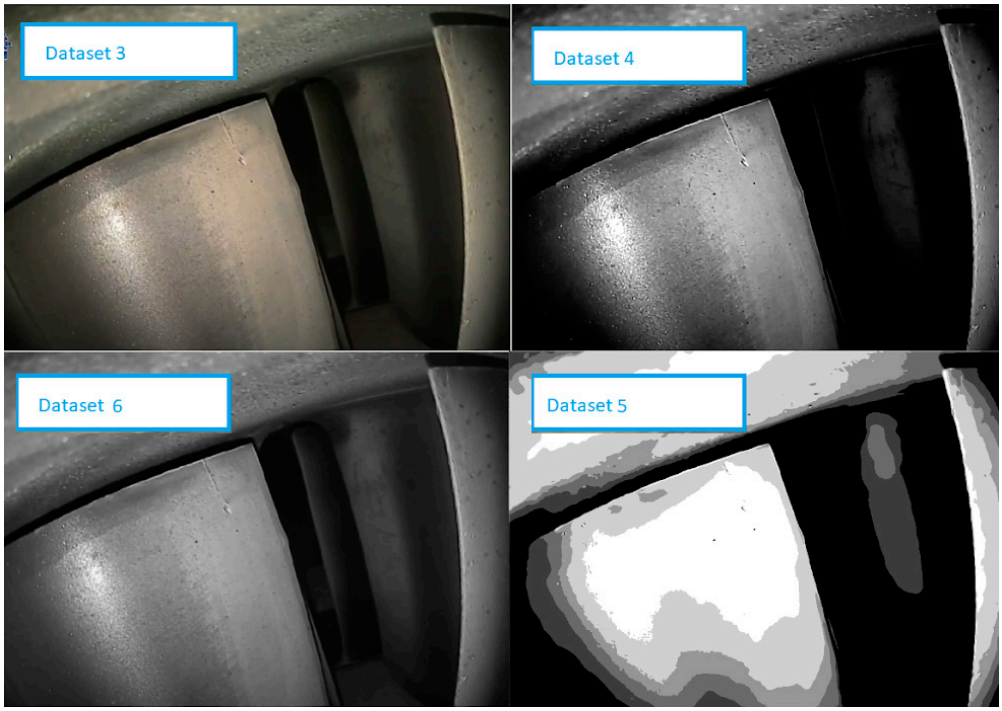


Figure 15. Image examples from Datasets 3, 4, 5, 6.

Table 2. Results from Model3_3x50, Model 4, Model 5, and Model 6 (mAP50, mAP50-95, confusion matrix).

mAP50/model/epoch	**Model 3**	**Model 4**	**Model 5**	**Model 6**
1 × 50 epochs	0.399	0.362	0.326	0.362
2 × 50 epochs	0.503	0.464	0.405	0.448
3 × 50 epochs	0.564	0.527	0.448	0.537
mAP50-95/model/epoch	**Model 3**	**Model 4**	**Model 5**	**Model 6**
1 × 50 epochs	0.286	0.262	0.232	0.267
2 × 50 epochs	0.366	0.338	0.291	0.344
3 × 50 epochs	0.401	0.382	0.325	0.378
mAP50-95/model/epoch	**Model 3**	**Model 4**	**Model 5**	**Model 6**
Blade	93	90	90	94
Nick	19	28	9	41
Dent	20	6	6	18
Tear	68	76	59	72

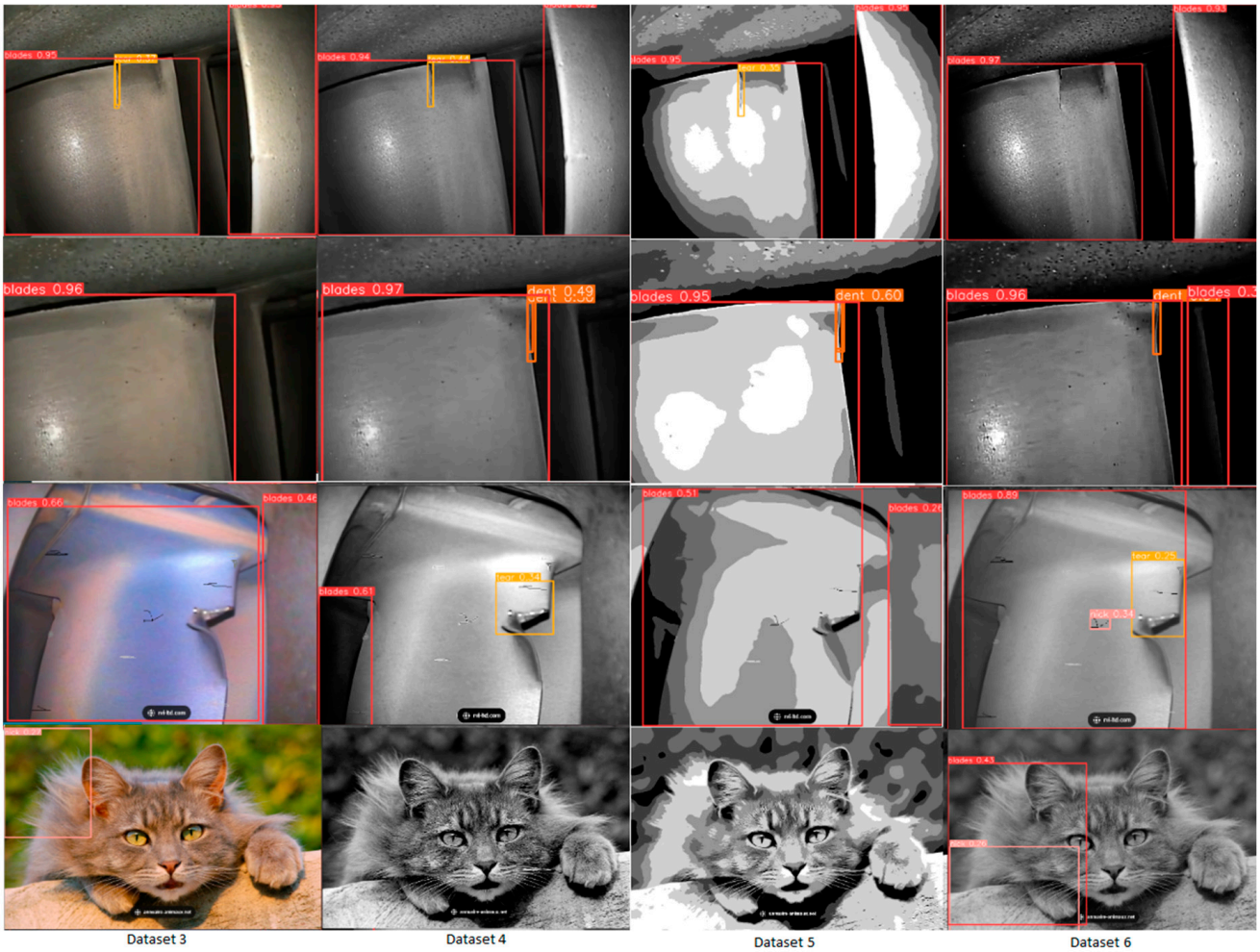


Figure 16. Result examples from Models 3_3x50, 4, 5, and 6 on new images.

To address the limited data problem, this study applies data augmentation techniques using 90-degree rotations to artificially expand Dataset 4, resulting in Dataset 7. Tests are conducted with the data, and the results from Model 7 are shown in Table 3.

Table 3. Results from Models 4 and 7 (mAP50, mAP50-95, confusion matrix).

mAP50/model/epoch	**Model 4**	**Model 7**
1 × 50 epochs	0.262	0.281
2 × 50 epochs	0.338	0.360
3 × 50 epochs	0.382	0.389
mAP50-95/model/epoch	**Model 4**	**Model 7**
1 × 50 epochs	0.362	0.405
2 × 50 epochs	0.464	0.523
3 × 50 epochs	0.527	0.555
mAP50-95/model/epoch	**Model 4**	**Model 7**
Blade	90	90
Nick	28	37
Dent	6	11
Tear	76	74

According to the mAP50 and mAP50-90 metrics, Model 7 performs slightly better than Model 4, with the expanded dataset. However, the number of tears detected by Model 7 is slightly lower than by Model 4. Comparative tests are subsequently carried out on 40 newly selected images, including 20 original images and 20 rotated 90 degrees. The results are shown in Table 4.

Table 4. Model results on 20 original images and 20 rotated 90 degrees.

Out of 20 Selected Images	Model 4	Model 7
Correct detection on standard images	14/20	14/20
Correct detection on rotated images	3/20	18/20

The table shows that Model 4 detects very few defects on rotated images while Model 7 performs better. This suggests that variations in camera angle have a positive impact on the model performance and incorporating rotated images in the training dataset may help the model generalize in identifying defects on reactor blades.

This study then fine-tunes a model using a combined Dataset 8, which contains images from Datasets 4, 5, and 6, to assess whether a model trained on multiple dataset types can perform effectively. The results are compared with Models 3–6, as shown in Table 5.

Table 5. Comparison of results from Models 3, 4, 5, 6, and 8 (mAP50, mAP50-95, confusion matrix).

mAP50/model/epoch	Model 3	Model 4	Model 5	Model 6	Model 8
1 × 50 epochs	0.399	0.362	0.326	0.362	0.334
2 × 50 epochs	0.503	0.464	0.405	0.448	0.440
3 × 50 epochs	0.564	0.527	0.448	0.537	0.503
mAP50-95/model/epoch	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 8**
1 × 50 epochs	0.286	0.262	0.232	0.267	0.242
2 × 50 epochs	0.366	0.338	0.291	0.344	0.321
3 × 50 epochs	0.401	0.382	0.325	0.378	0.364
Correct detection/3x50 epoch	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 8**
Blade	93	90	90	94	92
Nick	19	28	9	41	17
Dent	20	6	6	18	22
Tear	68	76	59	72	58

It is noted that while Model 8's average detection performance is lower than that of the other models, it performs best in detecting dent defects. Subsequent tests on Model 8 are conducted using alternative images from Datasets 4, 5, and 6. The results in Table 6 show that Model 8 performs worse than models trained on corresponding datasets, except for Dataset 5. Additionally, all defects detected by Model 8 are also detected by the model trained on the corresponding dataset.

Table 6. Comparison of results from Models 3, 4, 5, 6, and 8 based on Datasets 4, 5, and 6.

Out of 20 Selected Images	Trained Model with Corresponding Dataset	Model 8
Correct detection on images from Dataset 4	13/20	9/20
Correct detection on images from Dataset 5	11/20	11/20
Correct detection on images from Dataset 6	15/20	8/20

The trained models above struggle with detecting small defects. To address this, this study fine-tunes a model using a new data augmentation technique, by randomly placing small defects on reactor blades. Defect images are first clipped with background transparency and then overlaid onto the blade backgrounds generated from Dataset 3 by applying random transformations such as resizing, rotation, mirroring, and noise effects. Image preprocessing techniques, including gray scaling and contrast adjustment as in Dataset 4, are also applied. The resulting image examples are shown in Figures 17–19.

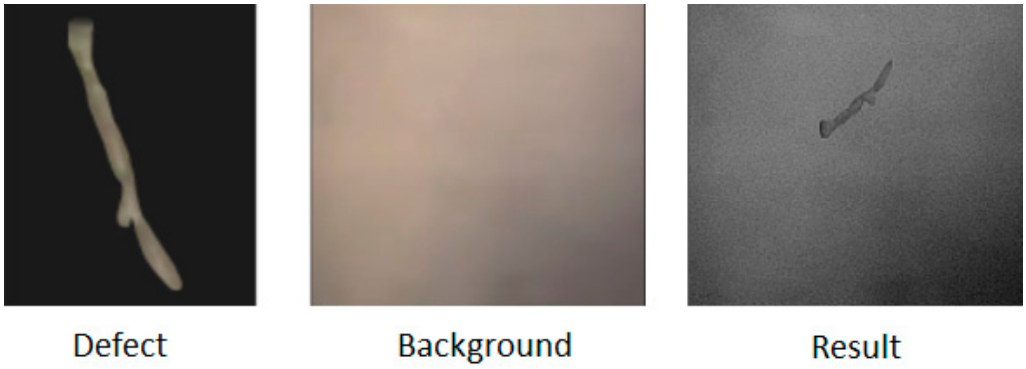


Figure 17. Decomposition of a synthetic image.

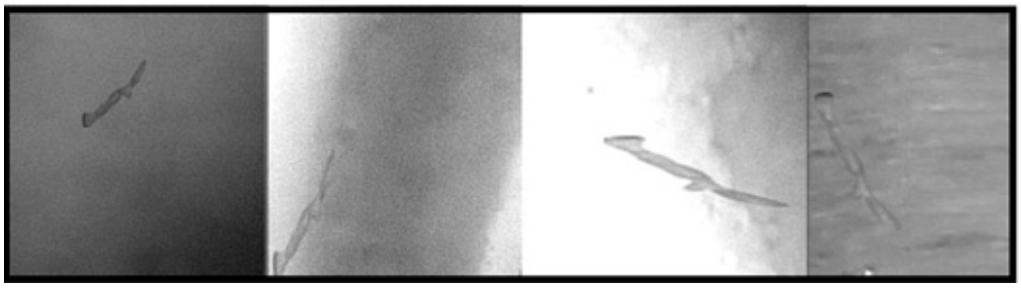


Figure 18. Examples of synthetic images, with a single defect and several backgrounds.

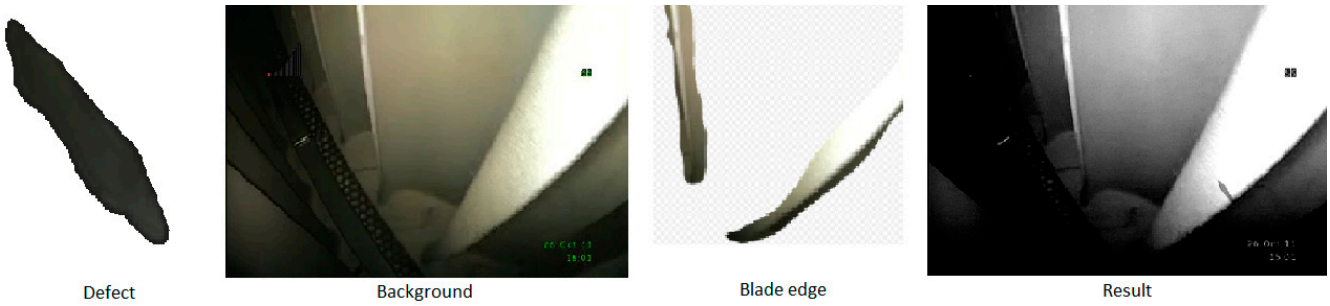


Figure 19. A synthetic image with a defect on the blade edge.

The main advantage of this method is that large amounts of images with various defects and backgrounds can be generated automatically. Additionally, the coordinates of defect collages can be retained, eliminating the need for manual annotation. Although the resulting images may not always look realistic, they are expected to enhance model performance. This is tested by training Model 9 on new Dataset 9 for 100 epochs and then evaluating it on unseen images, consisting of 20 real and 20 synthetic images. From Table 7, Model 9, trained exclusively on artificial defects, performs well at detecting artificial defects though it struggles with real defect images. Additionally, it is observed that some small-size real defects missed by previous models are detected.

Table 7. Model results on 20 original images and 20 synthetic images.

20 Selected Images	Model 9
Correct detection on standard images	5/20
Correct detection on artificial images	17/20

This study further examines the model's sensitivity to the training optimizer while keeping all other model settings at their default values (see Appendices A and B). The model is trained on Dataset 4 using various optimizers, and the results are presented in Table 8.

Table 8. Model performance with various optimizers (mAP50, mAP50-95, confusion matrix).

mAP50/Model 4/optimizer	Adam	Adamax	AdamW	NAdam	RAdam	RMS Prop	SGD	Auto
1 × 50 epochs	0.313	0.312	0.342	0.276	0.357	0.134	0.331	0.362
2 × 50 epochs	0.426	0.447	0.469	0.395	0.486	0.107	0.449	0.464
3 × 50 epochs	0.498	0.529	0.468	0.459	0.536	0.103	0.495	0.527
mAP50-95/Model 4/optimizer	**Adam**	**Adamax**	**AdamW**	**NAdam**	**RAdam**	**RMS Prop**	**SGD**	**Auto**
1 × 50 epochs	0.215	0.226	0.246	0.178	0.251	0.072	0.239	0.262
2 × 50 epochs	0.303	0.323	0.331	0.266	0.341	0.057	0.329	0.338
3 × 50 epochs	0.339	0.369	0.324	0.318	0.372	0.03	0.366	0.382
Correct detection/3x50 epochs	**Adam**	**Adamax**	**AdamW**	**NAdam**	**RAdam**	**RMS Prop**	**SGD**	**Auto**
Blade	91	92	92	89	90	0	92	90
Nick	31	30	20	19	28	0	31	28
Dent	0	0	0	6	24	0	3	6
Tear	65	76	65	64	71	15	72	76

As shown, while RAdam achieves the highest overall detection rate and mAP50 values, the "Auto" setting yields the best mAP50-95 performance. Certain optimizers, such as RMSProp, are less effective in this case. This may be attributed to the characters of the optimizers. While RAdam's strengths lie in stabilizing the training and "Auto" likely adjusts itself to maximize overall precision, RMSProp's sensitivity to settings may limit its effectiveness here. Additionally, the limited dataset could be a significant factor affecting the model's sensitivity in performance.

5. Discussion

This study develops a data-driven deep learning framework for automating aircraft engine borescope inspections by leveraging Computer Vision and deep learning techniques. Deep Neural Network models based on YOLOv8n are trained and evaluated to detect defects on turbine blades by varying various datasets, epochs, and optimizers.

Firstly, the reactor blade datasets with and without defects are created from real reactor blade videos and augmented through techniques such as grayscale settings, average brightness equalization, rotation, and synthetic dataset generation. These methods are adopted to address the challenges of obtaining high-quality, diverse, and sufficient endoscopic inspection images from public sources, which are essential for training a reliable Deep Neural Network model. Object detection models are then developed, trained, and evaluated based on YOLOv8n with these datasets accordingly. The training process is divided into several stages due to the significant duration of each session on a standard computer and the high number of epochs required by model performance. The results show that Model1_100 performs better overall than Model2_100 at detecting defects on blades, likely due to the presence of wide-shot blade images in Dataset 1, as opposed to the zoomed-in

images in Dataset 2. In general, image transformations help the models generalize defect identification at low epochs. However, this remains inconclusive at higher epochs. The comparison of model results, particularly among Models 3, 4, 5, and 6, reveals inherent uncertainties in model performance that while some models trained on data enhanced by Computer Vision techniques such as contrast, brightness equalization, and gray-level settings may appear more reliable in some types of defect detection, the relationship between these techniques and subsequent results cannot be generalized. Potentially, a factor analysis can be applied to identify the latent variables that cause and influence the model accuracy and reliability in defect detection by type. Additionally, a more comprehensive study is needed to investigate model accuracy as a function of computational efficiency and hyperparameter settings such as the number of epochs and batch sizes.

To study training variability or uncertainty, models such as Model3_50v1 and Model3_v2 are trained on the same hyperparameters, training settings, and data. The results indicate that training variability has to be considered when comparing models, potentially caused by factors such as random weight initialization, stochastic optimization, non-deterministic hardware operations, and possible underfitting. Additionally, the small size of datasets may also contribute to the model's lack of reliability.

This study compares model performance when trained with unrotated reactor blade images versus images rotated by 90° with Models 4 and 7, respectively. Results show that the expansion of the dataset using data augmentation techniques, primarily 90-degree rotation, improved model performance, as evidenced by Model 7's performance increase. This indicates that incorporating rotated images in training improves model generalization in identifying reactor blade defects and enhances defect detection performance. However, the generalization achieved by Model 8 did not outperform the specialized performance of individual models, such as Models 4, 5, and 6.

In general, the mAP50 values at higher epochs demonstrate moderate performance across models in detecting defects, while the low mAP50-95 values, particularly in Models 5 and 8, suggest that the two models struggle to localize objects accurately, especially at higher Intersection over Union (IoU) thresholds. This may indicate the ineffectiveness of the average brightness equalization and gray-level features used by these models. However, the results should be interpreted with caution due to the uncertainty introduced by factors such as data size, defect type, and potential metric bias.

This study also addresses the challenge of small defect detection by fine-tuning a model with a new data augmentation technique, i.e., randomly placing small defects on reactor blades. However, results show that a model trained with these synthetic collages is generally ineffective at detecting defects in real images, though it does detect certain defect types that models trained solely on real images missed. A potential qualitative solution is to have experts in the borescope inspection domain evaluate the realism and quality of the artificially generated images and then train the model on both synthetic and real images to assess if the synthetic features positively contribute to the model performance. Furthermore, this study attempts to generate synthetic images using the Deep Convolutional Generative Adversarial Network (DCGAN) technique [6]. A DCGAN model with retained settings [22] is developed and trained on Dataset 10 for 1500 epochs (approximately 3 h and 40 min). The results are displayed in Figures 20 and 21.

From the figures, similarities are observed among certain images, such as the location of defects and shadow areas. This can be explained by the generator's tendency to repeat pixel sequences once it successfully fools the discriminator, rather than exploring new variations. A more varied and extensive dataset would likely increase the diversity in the generated images. Another drawback of the GAN approach is the significant training time and large number of epochs required to generate coherent images.



Figure 20. Images generated by DCGAN showing similar characteristics.

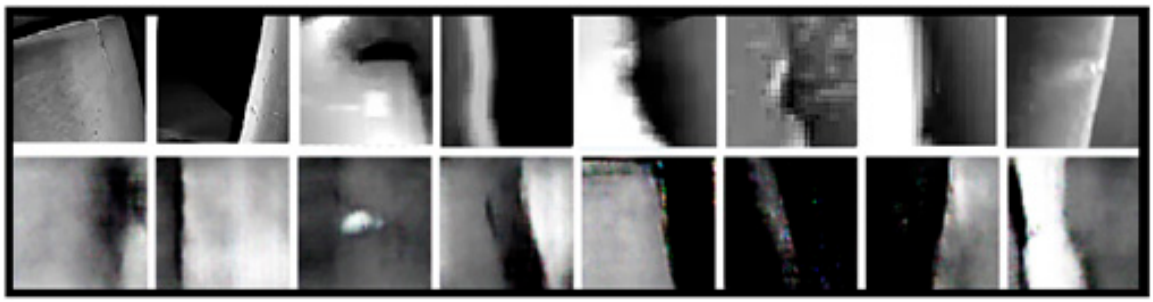


Figure 21. Comparison between real images and images produced by DCGAN.

Additionally, it is found that each model's detection capacity may depend on defect shape and image features like brightness and color proximity, making one model potentially more effective than another for a given image. The research results in this study also indicate that the choice of optimizers may improve model performance in detecting aircraft engine defects, although the limited dataset may significantly impact the model's sensitivity. This study adopts an early stopping strategy during model training (e.g., Model 3) to prevent overfitting and address limited computational resources, though the loss likely continues to decrease. A more comprehensive study could be conducted with sufficient computing power and a larger set of blade images.

From this study, borescope image defect detection results exhibit significant uncertainty in model outputs and model sensitivity. There are many sources of uncertainty in Machine Learning applications. This study introduces an uncertainty and sensitivity analysis classification along the Machine Learning pipeline. Specifically, Machine Learning uncertainty can be classified into three types, i.e., data uncertainty, model uncertainty, and evaluation uncertainty. Data uncertainty arises from incomplete datasets and imperfect information, such as the size and diversity of image datasets, as demonstrated in this study. Model uncertainty is related to inductive biases, including factors such as model selection and training variability such as dataset splitting, weight initialization, and epoch decision. This research has extensively examined model performance variability with various optimizers, but it is limited to a single model (YOLOv8n) with a few discrete epochs. Evaluation uncertainty pertains to the analysis and interpretation of model results, encompassing both quantitative metrics and qualitative methods, such as comparative approaches and visualization. A more comprehensive uncertainty and sensitivity analysis based on this classification is required for automatic aircraft engine borescope image defect detection, such as further studying image quality and defect type imbalance for data uncertainty analysis; applying dataset splitting and epoch stopping strategy for model uncertainty analysis; and using additional metrics (e.g., Fréchet Inception Distance [23]) along with a comparative approach (e.g., comparing synthetic and real images) for evaluation uncertainty analysis.

Moreover, a limitation of deep learning-based object detection is its requirement for a large amount of reactor blade images to produce an effective and reliable model.

The training process can be time-consuming and demands substantial computing power, making it challenging to perform on low-power computers. Further research can be conducted on the relationship between various image augmentation and synthetic image generation techniques and the performance uncertainty of the aircraft engine borescope inspection automation framework, as well as the model efficiency improvement through strategies such as model optimization, transfer learning, and distributed or remote training and implementation.

6. Conclusions and Future Work

This study develops a data-driven deep learning framework based on Computer Vision and YOLOv8n object detection techniques, capable of automatically detecting defects on reactor blades. It also examines the impact of data preprocessing on model training and performance. A limited set of borescope inspection images of engine blades, sourced from the public domain, is used for automatic defect detection, with Computer Vision and data augmentation techniques subsequently applied to expand the dataset. Given constrained computational and data resources, training on models is generally conducted with a low number of epochs.

The results indicate that the model trained on wide-shot blade images performs better overall at detecting blade defects compared to the model trained on zoomed-in images. Certain image transformations using enhancement, contrast, and gray-leveling filters, etc., may help models generalize defect identification at lower epochs. The comparison of multiple models' results reveals inherent uncertainties in model performance so that the relationship between image processing techniques and subsequent model results cannot be simply generalized. The results also indicate that training variability has to be considered when comparing models. The impact of epochs and optimizers on the model's performance indicates that incorporating rotated images and selecting an appropriate optimizer are keys for model training. Furthermore, models trained solely on artificially generated images from collages perform poorly at detecting defects in real images. A potential solution is to train the model both on synthetic and real images. This study also suggests a classification for a comprehensive uncertainty and sensitivity analysis of borescope inspection defect detection results, based on data uncertainty, model uncertainty, and evaluation uncertainty.

Utilizing more computational power would facilitate training models on larger datasets with more epochs. This would also enable more efficient training of a GCGAN, resulting in the generation of sufficiently realistic images. Future work will focus on improving the framework and conducting a more comprehensive uncertainty analysis by utilizing larger and more diverse datasets, particularly in image acquisition and model training with enhanced computational power. With improved stability and effectiveness, the framework could be further developed to classify defects by severity.

Author Contributions: Conceptualization, J.L. and T.S.; methodology, J.L. and T.S.; software, T.S.; validation, J.L.; formal analysis, T.S.; investigation, T.S.; resources, J.L. and T.S.; data curation, T.S.; writing—original draft preparation, T.S.; writing—review and editing, J.L.; visualization, T.S.; supervision, J.L. and K.W.J.; project administration, J.L. and K.W.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

```
# Default training settings and hyperparameters for medium-augmentation COCO training
task:  detect             # (str) YOLO task, i.e., detect, segment, classify, pose
mode:  train              # (str) YOLO mode, i.e., train, val, predict, export, track, benchmark

# Train settings -------------------------------------------------------------------------
model:                    # (str, optional) path to model file, i.e., yolov8n.pt, yolov8n.yaml
data:                     # (str, optional) path to data file, i.e., coco128.yaml
epochs:  100              # (int) number of epochs to train for
patience:  50             # (int) epochs to wait for no observable improvement for early stopping
batch:  16                # (int) number of images per batch (-1 for AutoBatch)
imgs:  640                # (int | list) input images size as int for train and val modes, or list[w,h] for predict and export modes
save:  True               # (bool) save train checkpoints and predict results
save_period:  -1          # (int) Save checkpoint every x epochs (disabled if < 1)
cache:  False             # (bool) True/ram, disk or False.  Use cache for data loading
device:                   # (int | str | list, optional) device to run on, i.e.  cuda device=0 or device=0,1,2,3 or device=cpu
workers:  8               # (int) number of worker threads for data loading (per RANK if DDP)
project:                  # (str, optional) project name
name:                     # (str, optional) experiment name, results saved to 'project/name'
exist_ok:  False          # (bool) whether to overwrite existing experiment
pretrained:  True         # (bool | str) whether to use a pretrained model (bool) or a model to load weights from (str)
optimizer:  auto          # (str) optimizer to use, choices=[SGD, Adam, Adamax, AdamW, NAdam, RAdam, RMSProp, auto]
verbose:  True            # (bool) whether to print verbose output
seed:  0                  # (int) random seed for reproducibility
deterministic:  True      # (bool) whether to enable deterministic mode
single_cls:  False        # (bool) train multi-class data as single-class
rect:  False              # (bool) rectangular training or validation if mode='train' or 'val'
cos_lr:  False            # (bool) use cosine learning rate scheduler
close_mosaic:  10         # (int) disable mosaic augmentation for final epochs (0 to disable)
resume:  False            # (bool) resume training from last checkpoint
amp:  True                # (bool) Automatic Mixed Precision (AMP) training, choices=[True, False]
fraction:  1.0            # (float) dataset fraction to train on (default is 1.0, all in train set)
profile:  False           # (bool) profile ONNX and TensorRT speeds during training for loggers
freeze:  None             # (int | list, optional) freeze first n or list of layers in training
overlap_mask:  True       # (bool) masks should overlap during training (segment train only)
mask_ratio:  4            # (int) mask downsample ratio (segment train only)
dropout:  0.0             # (float) use dropout regularization (classify train only)

# Val/Test settings ----------------------------------------------------------------------
val:  True                # (bool) validate/test during training
split:  val               # (str) dataset split to use for validation, i.e., 'val', 'test' or 'train'
save_json:  False         # (bool) save results to JSON file
save_hybrid:  False       # (bool) save hybrid version of labels (labels + additional predictions)
conf:                     # (float, optional) object confidence threshold for detection (default 0.25 predict, 0.001 val)
iou:  0.7                 # (float) intersection over union (IoU) threshold for NMS
max_it:  300              # (int) maximum number of detections per image
half:  False              # (bool) use half precision (FP16)
dnn:  False               # (bool) use OpenCV DNN for ONNX inference
plots:  True              # (bool) save plots during train/val

# Prediction settings --------------------------------------------------------------------
source:                   # (str, optional) source directory for images or videos
show:  False              # (bool) show results if possible
save_txt:  False          # (bool) save results as .txt file
save_conf:  False         # (bool) save results with confidence scores
save_crop:  False         # (bool) save cropped images with results
show_labels:  True        # (bool) show object labels in plots
show_conf:  True          # (bool) show object confidence scores in plots
at_battle:  1             # (int) video frame-rate stride
stream_buffer:  False     # (bool) buffer all (True) or return the most recent (False) frames
line_width:               # (int, optional) line width of the bounding boxes, auto if missing
visualize:  False         # (bool) visualize model features
augment:  False           # (bool) apply image augmentation to prediction sources
agnostic_nms:  False      # (bool) class-agnostic NMS
classes:                  # (int | list[int], optional) filter results by class, i.e.  classes=0, or classes=[0,2,3]
retina_masks:  False      # (bool) use high-resolution segmentation masks
boxes:  True              # (bool) Show boxes in segmentation predictions
```

```
# Export settings ------------------------------------------------------------------------
format:  torchscript      # (str) https://docs.ultralytics.com/modes/export/#export-formats
hard:  False              # (bool) use Kera=s
optimise:  False          # (bool) TorchScript:  optimise for mobile
you8:  False              # (bool) CoreML/TF INT8 quantization
dynamic:  False           # (bool) ONNX/TF/TensorRT: dynamic axes
simplify:  False          # (bool) ONNX: simplify model
setup:                    # (int, optional) ONNX: opset version
workspace:  4             # (int) TensorRT: workspace size (GB)
nms:  False               # (bool) CoreML: add NMS

# Hyperparameters ------------------------------------------------------------------------
lr0:  0.01                # (float) initial learning rate (i.e., SGD=1E-2, Adam=1E-3)
lrf:  0.01                # (float) final learning rate (lr0 * lrf)
momentum:  0.937          # (float) SGD momentum/Adam beta1
weight_decay:  0.0005     # (float) optimizer weight decay 5e-4
warmup_epochs:  3.0       # (float) warmup epochs (fractions ok)
warmup_momentum:  0.8     # (float) warmup initial momentum
warmup_bias_lr:  0.1      # (float) warmup initial bias lr
box:  7.5                 # (float) box loss gain
cls:  0.5                 # (float) cls loss gain (scale with pixels)
dfl:  1.5                 # (float) dfl loss gain
pose:  12.0               # (float) pose loss gain
kobj:  1.0                # (float) keypoint obj loss gain
label_smoothing:  0.0     # (float) label smoothing (fraction)
nbs:  64                  # (int) nominal batch size
hsv_h:  0.015             # (float) image HSV-Hue augmentation (fraction)
hsv_s:  0.7               # (float) image HSV-Saturation augmentation (fraction)
hsv_v:  0.4               # (float) image HSV-Value augmentation (fraction)
degrees:  0.0             # (float) image rotation (+/- deg)
translate:  0.1           # (float) image translation (+/- fraction)
scale:  0.5               # (float) image scale (+/- gain)
shear:  0.0               # (float) image shear (+/- deg)
perspective:  0.0         # (float) image perspective (+/- fraction), range 0-0.001
flip flops:  0.0          # (float) image flip up-down (probability)
fliplr:  0.5              # (float) image flip left-right (probability)
mosaic:  1.0              # (float) image mosaic (probability)
mixup:  0.0               # (float) image mixup (probability)
copy_paste:  0.0          # (float) segment copy-paste (probability)

# Custom config.yaml ---------------------------------------------------------------------
cfg:                      # (str, optional) for overriding defaults.yaml

# Tracker settings -----------------------------------------------------------------------
tracker:  botsort.yaml    # (str) tracker type, choices=[botsort.yaml, bytetrack.yaml]
```

Appendix B

Optimizer	Description
Gradient Descendant (GD) [24]	$Update\ Rule\ :\ \theta_{t+1} = \theta_t - \alpha\nabla J(\theta_t)$ θ_t : *Parameters at iteration t* α : *Learning rate* $\nabla J(\theta_t)$: *Gradient of the cost function J with respect to* θ_t
Stochastic Gradient Descent (SGD) [24]	$Update\ Rules\ : \theta_{t+1} = \theta_t - \alpha\nabla J\left(\theta_t, x^{(i)}, y^{(i)}\right)$ Performs a parameter update for each training examplex (i) and label y(i)
Gradient Descent with momentum [24]	$Update\ Rules\ :\ \theta_{t+1} = \theta_t - v_t$ $v_t = \gamma v_{t-1} + \alpha\nabla J(\theta_t)$ γ : Decay term for squared gradients (usually close to 1) v_t : *Momentum at iteration t*
RMSprop (Root-Mean-Square Propagation) [24]	$Update\ Rules\ : \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E\left[(\nabla J(\theta_t))^2\right]+\epsilon}}\nabla J(\theta_t)$ $E\left[(\nabla J(\theta_{t+1}))^2\right] = \gamma\, E\left[(\nabla J(\theta_t))^2\right] + (1-\gamma)(\nabla J(\theta_t))^2$ ε : Small constant to avoid division by zero
Adam (Adaptative Moment Estimation) [24]	$Update\ Rules\ :\ \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}\,\hat{m}_t$ $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$ $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$ $m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla J(\theta_{t+1})$ $v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla J(\theta_{t+1})$
Adamax [24]	$Update\ Rules\ :\ \theta_{t+1} = \theta_t - \frac{\alpha}{u_t}\,\hat{m}_t$ $u_t = \max(\ (\beta_2.u_{t-1})\ ,\ \lvert\nabla J(\theta_t)\rvert\ )$
AdamW [25]	$Update\ Rules\ :\ \theta_{t+1} = (1-\alpha\lambda)\theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}\,\hat{m}_t$ λ : *Weight decay value*
Radam (Rectified Adam) [19]	Similar to Adam optimizer but incorporates a correction to control the accumulation of bias in order to stabilize the optimization, particularly in the early stages of training. It uses a war-up phase at the start of training and corrects the variance term during optimization.
Nadam (Nesterov-accelerated Adam) [24]	$Update\ Rules\ :\ \theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon}\left(\beta_1\hat{m}_t + \frac{(1-\beta_1)\nabla J(\theta_t)}{1-\beta_1^t}\right)$

Appendix C

Metric Name	Equation			Description
Confusion matrix [26]		Predicted		It evaluates the performance of a binary or multi-class classification model. It displays the numbers of observations correctly and incorrectly classified by a model.
	True	True Positive (TP)	False Negative (FN)	
		False Positive (FP)	True Negative (TN)	
Precision [26]	$\frac{TP}{TP+FP}$			It measures the proportion of positive instances correctly identified among all the instances identified as positive by the model.
Accuracy [26]	$\frac{TP+TN}{TP+TN+FP+FN}$			It measures the model's ability to not misclassify negative instances as positive, depending on the confidence threshold. High accuracy means the model tends to make few false positives.

Metric Name	Equation	Description
Recall [26]	$\frac{TP}{TP+FN}$	It measures the proportion of correctly identified positive instances among all truly positive instances, i.e., the model's ability to identify all real positive instances, depending on the confidence threshold. High recall means that the model tends to detect positive instances well, thus minimizing false negatives.
Precision/recall curve		It shows the tradeoff between precision and recall for different thresholds, in which the larger the area under the curve, the higher the precision and recall, and the better the classifier perform.
Intersection over Union or Jaccard Index (IoU) [5]	$\frac{Aera(\text{ground truth} \cap \text{prediction})}{Aera(\text{ground truth} \cup \text{prediction})}$	It evaluates the overlap between two sets. In object detection, it evaluates how well the prediction of an object overlaps with the ground truth. The IoU is in the range of [0, 1]. The closer the IoU is to 1, the more overlapping there is.
mAP50 (mean average precision) [27]	$AP = \sum_{i=1}^{N} (R_i - R_{i-1}).P_i$ (the average precision for an object class) $mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$ (the mean average precision between the different classes)	It corresponds to the average of the areas under the precision/recall curves for a confidence threshold of 0.5.
mAP50-95 [27]	$mAP50_95 = \frac{1}{9} \sum_{\substack{i=0 \\ k=0.5+i*0.05}}^{9} mAP(k)$ k : Confidence threshold	It corresponds to the average of the mAP for a confidence threshold ranging from 0.5 to 0.95 with a step of 0.05.

References

1. International Civil Aviation Organization (ICAO). International Civil Aviation Organization. 2024. Available online: https://www.icao.int/about-icao/Pages/default.aspx (accessed on 1 December 2023).
2. Aust, J.; Pons, D. Bowtie Methodology for Risk Analysis of Visual Borescope Inspection during Aircraft Engine Maintenance. *Aerospace* **2019**, *6*, 110. [CrossRef]
3. Svensen, M.; Hardwick, D.; Powrie, H. Deep neural networks analysis of borescope images. In Proceedings of the European Conference of the PHM Society, Utrecht, The Netherlands, 3–6 July 2018; Volume 4.
4. Bian, X.; Lim, S.N.; Zhou, N. Multiscale fully convolutional network with application to industrial inspection. In Proceedings of the 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, 7–10 March 2016.
5. Wong, C.Y.; Seshadri, P.; Parks, G.T. Automatic borescope damage assessments for gas turbine blades via deep learning. In Proceedings of the AIAA Scitech 2021 Forum 2021, Online, 11–15 & 19–21 January 2021; p. 1488.
6. Upadhyay, A.; Li, J.; King, S.; Addepalli, S. A deep-learning-based approach for aircraft engine defect detection. *Machines* **2023**, *11*, 192. [CrossRef]
7. Li, Y.; Han, Z.; Xu, H.; Liu, L.; Li, X.; Zhang, K. YOLOv3-Lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions. *Appl. Sci.* **2019**, *9*, 3781. [CrossRef]
8. Shang, H.; Sun, C.; Liu, J.; Chen, X.; Yan, R. Deep learning-based borescope image processing for aero-engine blade in-situ damage detection. *Aerosp. Sci. Technol.* **2022**, *123*, 107473. [CrossRef]
9. RVI Ltd. Remote Visual Inspections. Remote Visual Inspections/Borescope Inspections–Videos. 2024. Available online: https://www.youtube.com/@rviltdremotevisualinspecti2187 (accessed on 7 March 2024).
10. Aircraft Maintenance Engineer. Engine Borescope Inspection After Bird Strike–Airbus A330–Videos. 2019. Available online: https://www.youtube.com/watch?v=zv29VSUZWg4 (accessed on 1 December 2023).
11. RVI Ltd. Remote Visual Inspections. RB211 535 HPC2. 2016. Available online: https://www.youtube.com/watch?v=nWk6IeL1IiM (accessed on 1 December 2023).
12. Abdulrahman, Y.; Eltoum, M.M.; Ayyad, A.; Moyo, B.; Zweiri, Y. Aero-Engine Blade Defect Detection: A Systematic Review of Deep Learning Models. *IEEE Access* **2023**, *11*, 53048–53061. [CrossRef]
13. Aust, J.; Pons, D. Comparative Analysis of Human Operators and Advanced Technologies in the Visual Inspection of Aero Engine Blades. *Appl. Sci.* **2022**, *12*, 2250. [CrossRef]

14. Aust, J.; Mitrovic, A.; Pons, D. Assessment of the Effect of Cleanliness on the Visual Inspection of Aircraft Engine Blades: An Eye Tracking Study. *Sensors* **2021**, *21*, 6135. [CrossRef] [PubMed]
15. Aust, J.; Shankland, S.; Pons, D.; Mukundan, R.; Mitrovic, A. Automated defect detection and decision-support in gas turbine blade inspection. *Aerospace* **2021**, *8*, 30. [CrossRef]
16. Ren, I. An Ensemble Machine Vision System for Automated Detection of Surface Defects in Aircraft Propeller Blades. Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2020.
17. Shen, Z.; Wan, X.; Ye, F.; Guan, X.; Liu, S. Deep Learning based Framework for Automatic Damage Detection in Aircraft Engine Borescope Inspection. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; pp. 1005–1010. [CrossRef]
18. Uzun, İ. Damage Detection in Aircraft Engine Borescope Inspection Using Deep Learning. Ph.D. Thesis, Aksaray University, Aksaray, Turkey, 2023. Available online: https://acikerisim.aksaray.edu.tr/xmlui/handle/20.500.12451/10763 (accessed on 1 December 2023).
19. Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; Han, J. On the Variance of The Adaptive Learning Rate and Beyond. Pub-lished as a Conference Paper at ICLR 2020. 2020. Available online: https://arxiv.org/pdf/1908.03265.pdf (accessed on 1 December 2023).
20. Explore Ultralytics YOLOv8. Roboflow. 2023. Available online: https://yolov8.com/ (accessed on 1 December 2023).
21. CVAT–Open Data Annotation Platform. CVAT.ai Corporation. 2023. Available online: https://www.cvat.ai/ (accessed on 1 December 2023).
22. Inkawhich, N. DCGAN Tutorial–PyTorch Tutorials 2.1.1+cu121 Documentation. 2024. Available online: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html (accessed on 7 March 2024).
23. Rezazadeh, N.; Perfetto, D.; Polverino, A.; De Luca, A.; Lamanna, G. Guided wave-driven machine learning for damage clas-sification with limited dataset in aluminum panel. *Struct. Health Monit.* **2024**, 14759217241268394. [CrossRef]
24. Ruder, S. An overview of Gradient Descent Optimization Algorithms. 2017. Available online: https://arxiv.org/pdf/1609.04747.pdf (accessed on 1 December 2023).
25. Guan, L. Weight Prediction Boosts the Convergence of AdamW. 2023. Available online: https://arxiv.org/pdf/2302.00195.pdf (accessed on 1 December 2023).
26. Powers, D.M.W. Evaluation: From Precision, Recall and F-Measure to Roc, Informedness, Markedness & Correlation. *arXiv* **2020**, arXiv:2010.16061.
27. Ultralytics. Ultralytics YOLO Docs: Analyse Approfondie des Mesures de Performance. 2024. Available online: https://docs.ultralytics.com/fr/guides/yolo-performance-metrics/ (accessed on 1 November 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.