AI-based general visual inspection of aircraft

John David Maunder
maunder.j@northeastern.edu

Jiesi Zhang
zhang.jies@northeastern.edu

Jianming Lu
lu.jianm@northeastern.edu

Tianyi Chen
chen.tianyi1@northeastern.edu

Zach Hindley
hindley.za@northeastern.edu

Abstract

Safety is the cornerstone on which the commercial airline industry is built. However, maintaining an aircraft is expensive, and a traditional inspection takes a long time and is prone to mistakes. Deep learning techniques can be used in aircraft maintenance thanks to the availability of Graphic Processing Units. The time required for general visual inspections of aircraft will be drastically reduced by the use of deep learning and remotely piloted aircraft systems (RPAS). In our proof of concept study, we use Yolov5 to build a model that uses high-quality data to find five different aircraft flaws.

1. Introduction

Safety is the cornerstone on which the commercial airline industry is built. Safety is enacted via the safety management system (SMS). The financial obligations, on the other hand, necessitate that tasks be completed as quickly and effectively as possible as it reduces maintenance costs. Maintenance explicitly means that the aircraft is out of operation, which drastically reduces its efficiency during "down time," despite being essential to the safe operation of the aircraft. To get the aircraft back in the air, it makes financial sense to complete the maintenance as quickly and safely as possible. The most frequent procedure in aviation maintenance, repair, and overhaul (MRO) is the general visual inspection (GVI) task. Depending on the aircraft, GVIs are performed every 400–600 flight hours or 200–300 flights and can take one or more aircraft maintenance engineers (AME) up to 10 hours to. The process can be error-prone[7] due to their subjective nature of what constitutes a defect.

Deep learning methods, which are primarily based on convolutional neural networks, have recently been developed as state-of-the-art solutions for computer vision issues like object detection. They typically outperform conventional techniques. Due to the availability of a large cloud infrastructure (such as Google Cloud or Microsoft Azure) or on-device GPU-accelerated microcomputers, many drawbacks of using neural networks (its computationally expensive) have been eliminated. Therefore, we can reduce inspection time, while increasing the effectiveness of a conventional and manual inspection by using a neural network approach. Technological advancements have reduced the time required to inspect all parts of a plane using a Remotely Piloted Aircraft System (RPAS) device. The inspection process will now only take three hours. The images taken by a drone can be passed to an operators tablet for real-time review.

By building on prior research and working together with industry partners, this study aims to address current issues related to applying RPAS technology to GVI of aircraft and report. For the benefit of the entire aircraft maintenance industry, we seek to report on the use of convolutional neural networks to identify airplane defects. The model will be put to the test using actual aircraft surface images captured by RPAS on a variety of aircrafts. The objective of this project is to use Yolov5 to create a model that can identify aircraft flaws using high-quality data. We're looking for as many flaws as we can. This means that we want to decrease the amount of falsely negative samples (defects that we have missed) at the expense of false positives (wrongly suspected defects), but the precise trade-off will be adjusted by a movable decision threshold.

2. Related Works

2.1. Aircraft Defects

Aviation safety depends on minimizing errors in all aspects of the system. While the role of human error has been appreciated, more attention has recently been directed towards reducing human error in maintenance and inspection. Aviation maintenance and inspection tasks are part of a complex system where everyone performs a different task in an environment with time pressure, sparse feedback, and difficult environmental conditions. These situational characteristics, combined with general human error tendencies, lead to various forms of error. In the worst cases, accidents

and loss of life can result. For example, on a Continental Express plane, 14 people died after a failed replacement of the horizontal stabilization screws caused the leading edge to separate.[4]

If an aircraft is to be properly maintained, the maintenance system must be error-free or fault-tolerant. Cracks and corrosion in the metal structures of commercial aircrafts are a reality of life. Defect correction requires defect detection, an area where system identification can be improved. The defect detection system consists of a human inspector assisted by various machines. Humans and machines are error-prone, so there needs to be a way where these system components become less error-prone. The detection/fix strategy used around the world is to specify a maintenance interval so that if a defect is too small to be detected in one inspection, it will be large enough to be detected and small enough to be safe in subsequent inspections. However, failing to detect cracks or corrosion, which are large enough to be detected, does not guarantee that it will not cause a problem until the next inspection.[3]

Some of the more complex classification algorithms are convolutional-based neural network systems such as RCNN, Fast R-CNN, Faster-CNN, YOLO and SDD. The majority of these neural networks are very fast but come with their limitations. For example, the article [9] states that using Region of Interests (ROI) and Region of Pooling Networks (RPN) techniques in R-CNN and Faster R-CNN respectively did increase the accuracy and speed of image recognition, however, it required a significant amount of computation power to complete. This trade-off made the overall process slower. Even though the Faster R-CNN was costly in terms of computing power, it was able to be accurate at a frame rate of 7 frames per second. The researchers address how to overcome the time issue when using a Faster R-CNN by implementing the You Only Look Once (Yolo) algorithm. The Yolo algorithm tries to overcome this computational inefficiency by dividing the image into a grid of cells, where the bounding box and class are based on these individual cells. Lastly, the article discusses how the "YOLO-CNN system" or any CNN system can be tested on image data sets to test the accuracy of MS COCO, VOC2012, and PASCAL VOC systems. In the context of flight defect detection, time spent processing is time lost for the plane to be in use. For this reason, we will be implementing the Yolov5 algorithm so we can speed up training and inference speeds.

Abdollahzadeh et al discusses the use of combining various machine learning models to detect airplane defects. The authors created a model where they broke down their images into 65X65 image patches and trained these images on a pre-trained model via transfer learning. Lastly, this model used the SURF interest point extractor in combination with a Support Vector Machine model to classify the defects into various classes. Our model can be used as a feature extractor and a model classifier unlike the author's model. Using one model speeds up inference time as it minimizes the use of multiple models. [6] Yusha et al wrote a systematic literature review on the various visual inspection techniques for aircrafts. This paper discusses the various inspection methods and the current state of inspection techniques for airplanes. The paper identifies the various techniques authors use to identify airplane defects. Some of these techniques use X-Ray images, lasers scan, and drones to capture various images. Once these images are captured, they are then classified into defect sub-classes using techniques like the Fournier transformation, mathematical formulas which identify gradient changes on planes, and machine learning models to identify defects.[11]

The last paper we analyzed while we were conducting our research was the paper by Gunatilake and Siegel in combination with the Roboticist Institute School of Computer Science at Carnegie Mellon University. This paper discusses the option of being able to identify corrosive patches on planes at various confidence levels. The authors of this paper were able to develop their corrosion classification model by applying a discrete wavelet frame transform to an image resulting in a 32x32 pixel image output. This pixel image will have 10 feature vectors applied to it. The output of these 10 feature vectors will be passed to a 3-layer neural network where classification occurs. Our paper differs from this model as our model can classify more defects than just corrosion. [10]

3. Algorithm Description

3.1. Convolutional Neural Networks

Convolutional neural networks can be generally broken down into two stage and one stage detectors [1]. Both detectors have four key stages that transition an image into an output classification for all images in the image. These four key stages are the input, backbone, neck, and head stages. The input stage usually transitions an input image or video into a vector that can be fed into the backbone of the convolutional network. The backbone of the convolutional network is a customized network with a finite amount of convolutional and pooling (average and max) stages. The backbone stage will produce a vector feature map containing the semantic and spatial image of the input image. This feature vector map is converted into a single row vector by the neck stage of the convolutional neural network. The neck stage of the neural network "flattens" (converts) the 2D feature maps into a 1D vector while minimizing the loss of the semantic and spatial information produced from the backbone stage [1]. The output from the neck stage will be a large 1D vector that will be passed to the "Fully Connected" layer (the head stage).

The main difference between the one stage and two stage detectors occurs at the head stage. In one stage detectors, bounding boxes and classification is done in one stage using a Dense Prediction approach. In two stage detectors, the detectors first group the output from the neck into a "region proposal network" before classifying images and placing bounding boxes on these images (see image 6)[2].

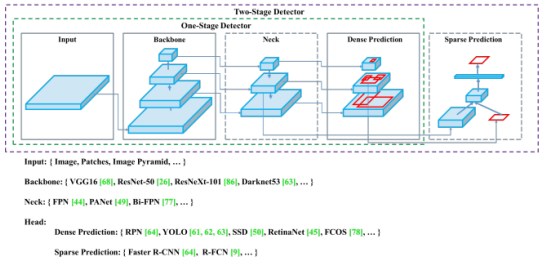

Figure 1. One Stage vs. Two Stage Detectors

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 2. DarkNet-53 Architecture

3.2. Yolo

The first iteration of the Yolo algorithm was developed by Joseph Reymond in 2015, but the algorithm did not become widespread until the third iteration which was released in 2018. The third iteration of Yolo (Yolov3) became more widely used since it was able to achieve a similar mAP-50 score while running at twice the speed of similar CNN's like RetinaNet-50 and RetinaNet-101[8]. The increase in speed and accuracy was a result of the implementation of the Darknet-53 backbone structure and implementation of the new Yolo Head[8].

Feature Extractor: The Darknet-53 architecture is a neural network with 53 convolutional layers. The significant performance increase in this architecture and prior versions of Yolo, is that this 53-layer backbone uses the below architecture at three various scales (image 7)[2]. By combining the 3X3 with a 1X1 convolution layer at three different scales this algorithm can down sample the input image without losing semantic and spatial details allowing the Yolo algorithm to track large, medium and small objects without losing speed[8].

Head/Detection: For each image passed into the Head, the image will be divided into an S * S grid where S denotes grid size. Each grid cell predicts B bounding boxes. Each bounding box will predict whether there is an object in that bounding box (confidence score), the four coordinates of that bounding box and the probability score for each class. Confidence is defined as Pr(Object) * IOU(truth, pred). The confidence score equals the Intersection Over Union (IOU) between the predicted box and the ground truth. IOU is the area of overlap of bounding boxes divided by the union area of the same bounding boxes. If no object exists in that cell, the confidence score is zero[8].

$$\Pr(Object) * IoU_{pred}^{truth} \tag{1}$$

Each bounding box will only contain at most one object. Therefore, the class (C) with the highest probability will be bounded by that bounding box. The class probability is conditional since it is dependent on a bounding box containing an object (confidence score of 1). The confidence score formula can be seen below. The probability of each class is calculated by using independent logistic classifiers, where these losses are trained using binary cross entropy. If the same or multiple grid cells have overlapping bounding boxes when they are predicting the same object, non maximum suppression will be used to make the final bounding box. Non max suppression takes the bounding box with the largest Intersection over Union (IoU) with the ground truth bounding box.

$$\begin{aligned}\Pr(Class_i|Object) * \Pr(Object) * IoU_{pred}^{truth} = \\ \Pr(Class_i) * IoU_{pred}^{truth}\end{aligned} \tag{2}$$

The bounding boxes are trained using anchor boxes where the shape of the anchor boxes are modified during training using a K-means clustering algorithm. The output prediction for Yolo can be summarized as S * S * 3 * (B * (5 + C)) [8]

Yolo Loss Function:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 \right]$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - (C_i))^2 \quad (3)$$

$$+\lambda_{noob} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \ (C_i - \widehat{C_i})^2$$

$$+ \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \epsilon classes} (p_i(c) - \widehat{p_i}(c))^2$$

3.3. Yolov5

Yolov5 will be used to train our dataset for our hand gesture recognition algorithm. As of September 23rd, 2022. Yolov5 does not have a paper released on the algorithm. The Yolov5 algorithm is very similar to the yolov4 algorithm based on the ultralytics github repo, but for the purpose of this paper, we will discuss the architecture of the Yolov4 algorithm since there is a published paper on this algorithm whereas there is no paper published for the Yolov5 algorithm.

CSPDarknet contains "Dense Blocks". Dense blocks divide the input into two halves where one half is sent through the dense block, while the other half will be routed directly to the next step without any processing. CSP preserves fine-grained features for more efficient forwarding, stimulates the network to reuse features and decreases the number of network parameters. The addition of dense blocks will help minimize the gradient descent problem making the model more robust. The downside of adding dense layers is it can make the model slower, which is why the Yolov4 algorithm minimizes the use of dense layers[1].

Yolov4 uses a spatial pyramid pooling approach and a Path Aggregation Network to transition the feature maps from the CSP Backbone to the Yolo Head. The SPP block is used to preserve spatial dimension for the feature map by applying a 1x1 convolution from the feature map and then duplicating and pooling the different scales so the 3 feature maps will retain the sizes of sizefmap x sizefmap x 512. The SPP process is combined with the PAN process to produce a vector output that will be passed to the Yolo Head[5].

The PAN process is like the feature pyramid pooling approach discussed above except it uses a bottom-up approach to pool semantic features together. This bottom-up approach allows the Yolov4 algorithm to create a shortcut to directly connect fine-grained features from low-level layers to the top one[5].

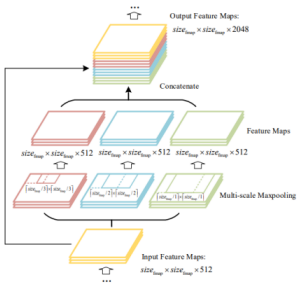

Figure 3. Spatial Pyramid Pooling

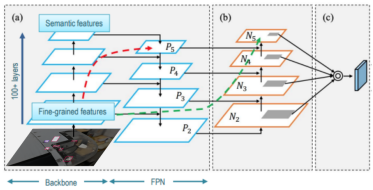

Figure 4. Path Aggregation Network

3.4. Optimizations for the Yolo Algorithm

Since the goal of this paper is to implement a speedy algorithm that can still accurately detect a hand gesture, we will modify the algorithm in the following away:

1. Minimize the depth of the algorithm
2. Hyper Tune parameters
   (a) Minimize Bounding box amounts
   (b) Fine tune anchor boxes
   (c) Apply batch normalization
   (d) Fine tune activation functions
3. Perform data augmentation to make the data set more robust

4. Methodology

4.1. Aircraft Defect Detection

The image below will provide an overview on the process we used to train the Aircraft Defect Model.

The goal of this paper is to train and optimize a convolution neural network so it can detect airplane defects. The first step of this process is to first build a data set to train our model on. The photos for the data set were taken by students at Northeastern on various dates throughout the June and July time period in 2022. The photos were broken up into two different data sets before they were ultimately combined into one final data set. As each batch of photos were taken, they were uploaded to an image annotation site called "Roboflow". Each image had each defect in the image labelled into one of five classes (please see the list

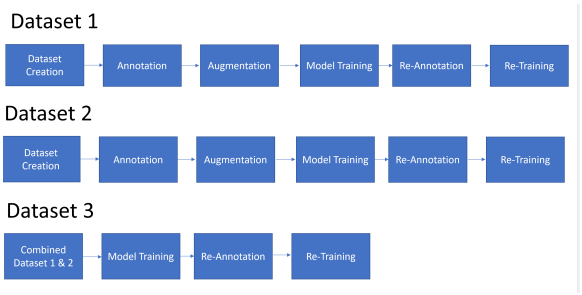

Figure 5. Methodology Overview

below). If the image did not have at least one defect in the image, the image was discarded since it would not help the image model train. After annotation was completed for all images, these images were then broken up randomly into ”training”, ”validation”, and ”testing” groups. From here, these images were imported into ”Google Collab” so the model could train its weights on the training class, update its weights using the validation class, and run inference (result testing) on the testing class.

1. Peeling
2. Cracks
3. Missing Screws
4. Dent
5. Corrosion

The defects in question are not square or rectangular in shape. Most of the defects (figure 6) are long, thin, curved, or are spherical in nature. These non-rectangular shapes increase the likelihood that a defect will not be detected since Yolov5 uses rectangular anchor boxes to make its detections. The difference in shape between the defects and the anchor boxes (figure 6) will cause more background noise to be picked up during training. This will impact the model's ability to recognize patterns during training, which will negatively affect the results during inference testing.

To combat the above problem, we went through multiple iterations of re-labelling various images in both image data sets. This process was undertaken to increase the likelihood that all defects will be detected during inference testing. We started the re-labelling process with data set 1. Data set 1 includes up close photos that are smaller in size and were taken outside of an airport hanger. The images in data set 2 were shot at a wider angle and were of higher resolution. Before we combined these two data sets into one, we trained these two data sets separately on the Yolov5 algorithm to see which individual images were more likely to predict a defect. After we trained each data set, we went back and re-labelled certain defects so they would appear in larger bounding boxes. This is important since larger and tighter bounding boxes increases the chance of detection. The final step was combining these two data sets and training the combined data set on various pixel sizes ranging from 640px-1088px.

Examples from both data sets can be seen below. In addition, the breakdown of the data set by class is also shown below.

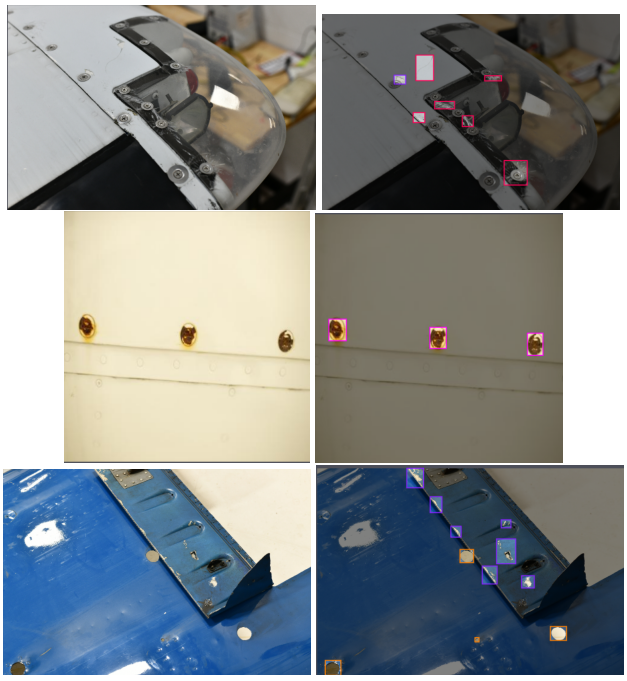

Figure 6. Data Set Examples

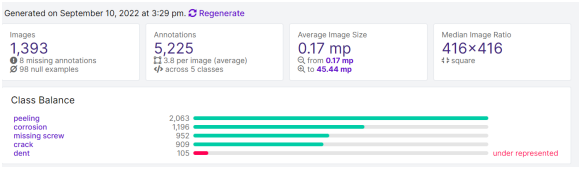

Figure 7. Class Breakdown

5. Results

The goal of this test was to determine if a combined custom data set could be trained on the Yolov5 model to determine if a Yolov5 model could identify airplane defects. Based on the inference tests ran on Google Collab all images were able to be detected during inference testing.

As can be seen in image 8 and 9, our model was able to detect that the plane had 10 missing screws in one photo and 8 missing screws in another photo. In photo 10, our model was able to identify which screws were corroded and which were not. In addition, our model also was able to detect an area of peeling in the same photo. Image 11 shows our

model picking up paint peeling and cracks on an airplane wing. Lastly, in image 12 our model was able to detect peeling and a dent formed on the plane.

We ran and trained both models on data set 1, data set2, and data set 1 and 2 combined. Data set 1 preformed better at detecting four of the five various defects (the "missing screws" class preformed better on data set 2) than data set 2. It also preformed equally as well with the combined data set for all classes other than the missing screw class. The photos that were taken closer to the plane allowed the bounding boxes to be bigger when annotating the training images. This resulted in the Yolov5 model having more room for error when detecting a defect.

An increase in image size resulted in longer training and inference testing but, it did increase the accuracy of the model by about 2-4 percent from size 640px to 1080px. Lastly, in terms of detection speed, on a Telsa T4 GPU, the Yolov5 algorithm was able to detect a defect(s) in 14ms or 71 frames per second on average.

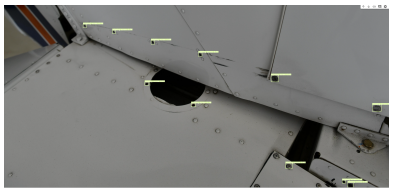

Figure 8. 10 Missing Screws

Figure 9. 8 Missing Screws

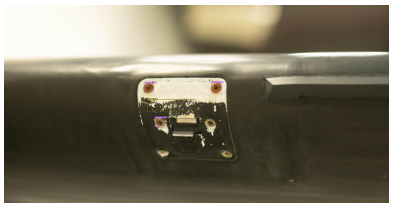

Figure 10. Peeling and Corrosion

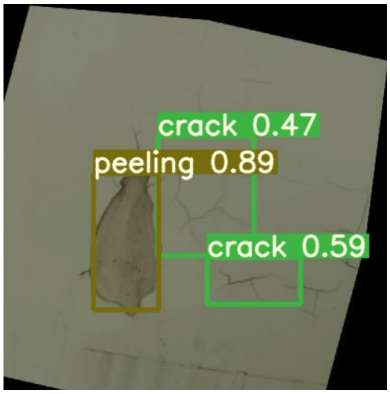

Figure 11. Crack and Peeling

Figure 12. Peeling and Dent

6. Conclusion and Future Work

6.1. Conclusion

This paper examined the use of using the Yolov5 algorithm to detect airplane defects. The combined custom data set was trained and ran on Google Collab. The model was able to accurately detect all five classes. The two original data sets were originally trained, re-labelled, and then re-trained separately before they were combined into one final data set. We underwent this process so we could best identify which image sizes, angles, distance from the plane and resolution would result in creating the highest probability of detecting all five defects.

By going through the above process, we were able to identify which images work best at identifying the five defects. If our model is used in production to identify which planes have defects via a drone or a manual camera, the images taken by these objects should meet the following criteria to maximize the likelihood of detecting a defect:

1. The images should be taken close to the plane
2. Images should be taken in a well-lit area
3. Higher resolution cameras tend to increase accuracy
4. Photos should not be taken at an angle
5. Larger sizes tend to work better but they hit their detection limit around 1080 pixels

6.2. Future Work

Yolov5 is an algorithm that preforms quite well when it comes to detecting defects (missing screws, peeling, corrosion, dents, cracks) on a plane. This algorithm is advanced, but it is two years old and newer algorithms have been created. For example, the Yolov7 and yolov6 algorithm was released in July 2022. These algorithms were tested and trained on our combined data set, but they preformed worse than our Yolov5 algorithm.

As previously discussed, some defects are shaped in a non-rectangular fashion which will limit the accuracy of most image detection algorithms. For example, the crack class will be thin, span in multiple direction and will be small in nature. This type of defect may be more easily detected when using an "instance segmentation" model, since these models better capture non-rectangular shapes in images. Yolov7 has come out with an image segmentation model but since the model is new there is not enough support to make this model functional. Each year (or every other year) a new more advanced algorithm is created or modified by the computer vision community. Therefore, each time a new community tested model is released, these models should be trained on the existing data set. Based on the accuracy and speed tests from the new models, we can determine if the newer model preforms better than our current Yolov5 model. If it does, we can easily swap the newer model into our airplane defect classification pipeline.

Lastly, for the Yolov5 algorithm to work fully, a full data pipeline needs to be established with the device that is taking the image and the hardware that is preforming the calculations. For example, if all images are taken by a drone or a handheld camera, the following steps must be taken so detection of defects can remain consistent:

1. Images Need to be cropped to the chosen image size that the model was trained on
2. Lighting needs to be consistent with the data set
3. The number of images captured should be adjusted for the computational power of the hardware preforming the detection of defects as less powerful hardware can take 1-100 times as long to preform image calculations.

Lastly, if the calculations are being preformed on a less powerful device like a CPU (so image inference can be run on a drone) we can modify the Yolov5 algorithm to run more efficiently on that device.

References

[1] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni. Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3. In *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, pages 1–6, 2019.

[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[3] D. E. Bray and R. K. Stanley. *Nondestructive evaluation: A tool in design, manufacturing, and service*. CRC press, 2018.

[4] C. G. Drury, P. Prabhu, and A. Gramopadhye. Task analysis of aircraft inspection activities: methods and findings. In *Proceedings of the Human Factors Society Annual Meeting*, volume 34, pages 1181–1185. SAGE Publications Sage CA: Los Angeles, CA, 1990.

[5] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang. Dc-spp-yolo: Dense connection and spatial pyramid pooling based yolo for object detection. *Information Sciences*, 522:241–258, 2020.

[6] T. Malekzadeh, M. Abdollahzadeh, H. Nejati, and N.-M. Cheung. Aircraft fuselage defect detection using deep neural networks, 2017.

[7] U. Papa and S. Ponte. Preliminary design of an unmanned aircraft system for aircraft general visual inspection. *Electronics*, 7(12):435, 2018.

[8] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[9] M. Saqib, S. Daud Khan, N. Sharma, and M. Blumenstein. A study on detecting drones using deep convolutional neural networks. pages 1–5, Aug 2017.

[10] M. Siegel and P. Gunatilake. Remote enhanced visual inspection of aircraft by a mobile robot. In *Proc. of the 1998 IEEE Workshop on Emerging Technologies, Intelligent Measurement and Virtual Systems for Instrumentation and Measurement (ETIMVIS'98)*, pages 49–58, 1998.

[11] Y. D. Yasuda, F. A. Cappabianco, L. E. G. Martins, and J. A. Gripp. Aircraft visual inspection: A systematic literature review. *Computers in Industry*, 141:103695, 2022.