Semi-Synthetic Data Augmentation for Computer Vision Applications in Aircraft Defect Detection

by

Jonathan Gallagher

A research paper
presented to the University of Waterloo
in fulfillment of the
requirements for the degree of
Masters
in
Computational Mathematics

Waterloo, Ontario, Canada, 2024

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Aircraft visual inspections, while critical for flight safety, remain vulnerable to both human error and subjective interpretation. The scarcity of high-quality labeled data has historically proven to be a hindrance to the development of robust computer vision models capable of performing defect-detection in aerospace applications. This work explores a novel "semi-synthetic" data generation framework combining user-selected real aircraft defects with high-fidelity 3D vehicle models in Unreal Engine. We investigate the potential benefits of incorporating this semi-synthetic data into the training pipeline of a YOLOv5 model, comparing its performance against a benchmark model trained with strong augmentation. Our analysis reveals an unexpected improvement in training dynamics, with the enhanced model achieving convergence after processing 10.5% fewer sample iterations during training. These findings suggest that domain-specific 3D modeling paradigms may offer promising pathways for improving model training efficiency while maintaining detection performance, with further exploration into synthetic data generation techniques potentially yielding futher improvement.

Acknowledgements

I would like to extend my gratitude to both Dr. Roberto Guglielmi and Dr. Derek Robinson for their support and supervision throughout the course of this project, and most of all for their willingness to let me explore.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Aircraft visual inspection is one of the cornerstones of the regulatory framework responsible for ensuring our shared ability to enjoy safe and reliable air travel. Inspection for vehicle surface defects occurs regularly between commercial flights providing a guarantee that vehicles are properly maintained and safe to travel. Standard visual inspection procedures employ crews of trained professionals for the purpose of identifying and making record of any observable structural and material irregularities, certifying vehicles of their readiness for operation. Unfortunately, even experienced and well-trained maintenance crews are capable of making errors. Given the nature of these inspections, there is no certainty that any given assessment is entirely objective, with existing literature highlighting how human error can negatively impact both inspection quality and reliability [3, 10]. However, recent technological developments in the realm of robotics and computer vision models stand to offer alternatives to current - potentially outdated - procedures. As deep learning models have become increasingly prolific and far-reaching in their applications, there has been a growth of interest toward incorporating suitable and affordable machine learning aids as part of a well-rounded visual inspection toolkit.

A nascent but nonetheless promising practice in aircraft visual inspection is the deployment of appropriately instrumented UAVs (Unpiloted Aerial Vehicles) equipped with embedded computer vision systems capable of detecting surface defects during a fly-around. Ready access to these tools may provide an avenue which could improve the ability of maintenance personnel to detect aircraft surface defects when used in concert with established domain expertise. In many studies, modern computer vision models boast fault detection accuracy which extends well beyond the capability of even the most well-trained human eyes [23].

Figure 1.1: Examples of aircraft fuselage defects.

Unfortunately, the data hungry nature of contemporary deep learning methods is a distinct obstacle to building and implementing effective models suitable for application in aircraft defect detection. Unlike other domains, where training data may be plentiful, aerospace applications suffer from a dearth of high-quality publicly available labeled data. This is the case as data collection from in service aircraft can be uniquely difficult and prohibitively time consuming.

Training data scarcity is by no means an issue which is unique to computer vision in aerospace. Other vision tasks such as Human Action Recognition often also suffer from data sparsity due to factors concerning data privacy; especially pertinent when handling potentially sensitive information. A novel solution to the challenges of data scarcity which is increasingly gaining traction is the use of 3D rendering environments to generate synthetic data for the purpose of model training. Such synthetic datasets are often used in combination with real data to create an augmented dataset which ideally better represents the target distribution [17, 14]. In such instances, digital assets resembling the target classes are used to generate high-quality rendered scenes from which additional training data can be extracted. This approach allows users to have complete control over the synthetically generated data, which can be used in combination with a set of ground truth data to ensure the model is well adjusted for the desired application. Despite successful

application across diverse domains in computer vision research: from weapon detection in CCTV footage [9], facial recognition systems [12], and pedestrian detection [8], synthetic data generation of this kind has yet to be experimented with for aircraft defect detection [23].

There are potentially many factors contributing to this gap in published work, perhaps led by the scarcity of high-quality 3D assets possessing representative defects upon which a model could be trained, and the potentially prohibitive temporal investment that would be required to create such assets from scratch. This work aims to address some of these challenges by proposing a novel workflow for semi-synthetic data generation, combining aspects of real aircraft defects with pre-existing synthetic assets in a 3D modeling environment, leading to contextually relevant data augmentation. We first demonstrate the process for generating new semi-synthetic samples, and then validate the effectiveness of our synthetic data by training a series of 30 randomly seeded YOLOv5 object detection models on three datasets. Datasets include a baseline dataset subject to strong data augmentation, an augmented dataset which employs 125 semi-synthetic images alongside strong augmentation, and a fractional dataset containing a random subset equating to 50% of the available semi-synthetic data. In doing so we establish a robust baseline for performance improvement. Our results indicate that the inclusion of a small fraction well-selected semi-synthetic samples bolster a representation of the target domain which is more easily learnable, offering a potential avenue for further improvement in current defect-detection frameworks.

Chapter 2

Background

2.1 Computer Vision in Aircraft Defect Detection

Interest in using technological aids to improve aircraft visual inspection is not new, with published interest dating as far back as the early 1990's [4, 6]. Recent advances in both robotics and computer vision have catalyzed renewed interest and substantial research activity, significantly expanding the scope of the field.

Ground-based defect detection systems consisting of instrumented ground based vehicles have achieved notable success in specific applications [2]. However, their inherent limitations, such as restricted maneuverability and difficulty accessing hard-to-reach areas have resulted in a search for a more suitable form factor. The recent emergence of cost-effective, easy to operate UAVs has overcame many of the challenges faced by previous ground based systems, leading to them being the default development platform in much of the contemporary aircraft defect detection literature [23].

While some work, like that done by Liu et al. [16] explicitly focus on the practical challenges relating to the implementation of UAV systems themselves, much of the existing work focuses on refining deep-learning model architectures to enhance their detection capabilities. For instance, Huang et al. [11] introduced ASD-YOLO, a modified YOLOv5 model optimized for aircraft surface defect detection. By incorporating a deformable convolutional feature extractor, an attention mechanism, and a contextual enhancement module, their approach significantly improved the detection of defects of varying scales and shapes, achieving a substantial increase in mean Average Precision (mAP) over baseline YOLOv5. Similarly, Zhang et al. [25] proposed a semisupervised framework for fuselage defect detection. Their method leverages dynamic attention and class-aware adaptive pseudolabel

assignment to enhance small defect detection while reducing the reliance on extensive manual annotations. Avdelidis et al. [1] further emphasized the importance of automating defect recognition and classification using UAV-based visual inspections. Their two-step approach employed pretrained convolutional neural networks (CNNs), fine-tuned with a custom dataset for defect recognition, followed by defect classification using an ensemble of classifiers. While existing work underscores significant progress in terms of bespoke model development, the scarcity of high quality labeled training data remains a persistent theme.

2.2 Game Engines for Synthetic Data Generation

The use of 3D game engines for the purpose of synthetic data generation is a field which is still in its early stages of development and understanding. Some of the earliest and perhaps most ambitious applications of game engines for deep learning precipitated from industry research and development in pursuit of self driving vehicles. These efforts are exemplified by early projects like CARLA, an open source development environment forked from Unreal Engine. Developed by researchers at Intel, CARLA represents one of the earliest demonstrations of game engines being repurposed to create highly controllable, physically accurate synthetic environments for deep learning [7]. Martinez-Gonzalez et al. [19] further expanded the scope of applications for Unreal Engine in deep learning, by publishing UnrealROX, in which they introduced a virtual reality development environment on top of Unreal Engine 4, used to develop synthetic data for a variety of robotic vision applications.

Since gaining recognition, consistent efforts have been made to take advantage of open source tools for 3D rendering. Pollok et al. [21] released UnrealGT, a framework demonstrating novel approaches for generating ground truth datasets for semantic segmentation tasks utilizing purely synthetic environments . De Souza et al. [5] further demonstrated the flexibility of game development environments for a variety of tasks extending far beyond the scope of autonomous navigation, focusing instead on generating datasets for human action recognition.

More recently, an emerging branch of related research has begun exploring the application of 3D rendering environments to address data scarcity challenges in infrastructure-related computer vision tasks. While research exploring the utility of synthetic data of this nature specifically for infrastructure and equipment condition monitoring is sparse, available studies have demonstrated promising results [18, 20]. These works suggest that game-engine based approaches may assist in overcoming persistent challenges facing the construction of reliable infrastructure inspection datasets.

Notably absent from the small body existing literature is the application of game engine-based synthetic data generation to the task of aircraft defect detection. Despite the domain sharing many of the same data collection challenges present in infrastructure monitoring, it has yet to be meaningfully explored. This gap in the literature, combined with the demonstrated success of synthetic data in closely related domains, inspired the present work's investigation into game engine-based solutions for aircraft defect detection tasks.

2.3 YOLO Single Stage Detectors

YOLO are a family of object detection models widely considered to be state of the art for real time applications. Proposed by Redmon et al. [22], the core idea behind YOLO is to treat object detection as a single regression problem, rather than a series of problems as was standard in prior works. The novelty of the YOLO architecture is that it predicts both bounding boxes and class probabilities directly from the input image during a single forward pass. Unlike region-based methods, which rely on distinct region proposal networks and require multiple passes to make a prediction, YOLO offers a simple end-to-end approach, resulting in the ability to make rapid inferences. The speed and accuracy of YOLO models have made them one of the leading choices for deployment in real-time applications.

In YOLO, input images are divided into an $S \times S$ grid, where each grid cell predicts B bounding boxes and corresponding confidence scores. Each bounding box is represented by four parameters defining the center coordinates, width, and height, normalized to the grid cell size. Each grid cell also predicts C class probabilities, assuming a single object is present in the cell. The confidence score reflects both the probability that an object exists in the box and the accuracy of the box's coordinates.

YOLO models employ a backbone convolutional neural network to extract features from the input image. These features are then processed through additional convolutional layers to generate the final prediction tensor with dimensions $S \times S \times (B * 5 + C)$, where 5 represents the four box coordinates plus one confidence score. This unified architecture allows the model to learn global contextual information about objects and their relationships within the image. The training process utilizes a multi-part loss function that balances several competing objectives: localization accuracy (box coordinates), confidence prediction (objectness), and classification accuracy.

2.3.1 Composite Loss Function

The YOLO object detection framework employs a multi-task loss function that integrates four objectives: bounding box regression, object confidence prediction, no-object confidence prediction, and classification. The total loss is expressed as:

$$\mathcal{L} = \lambda_{\text{coord}}\mathcal{L}_{\text{coord}} + \mathcal{L}_{\text{obj}} + \lambda_{\text{noobj}}\mathcal{L}_{\text{noobj}} + \mathcal{L}_{\text{cls}},$$

where $\mathcal{L}_{\text{coord}}$, $\mathcal{L}_{\text{obj}}$, $\mathcal{L}_{\text{noobj}}$, and $\mathcal{L}_{\text{cls}}$ correspond to losses for bounding box regression, object confidence, no-object confidence, and classification, respectively. The hyperparameters λ_{coord}, λ_{obj} and λ_{noobj} balance the contributions of these terms.

Bounding Box Regression Loss The coordinate loss penalizes errors in bounding box center, width, and height predictions:

$$\mathcal{L}_{\text{coord}} = \sum_{i=1}^{S^2}\sum_{j=1}^{B} \mathbf{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \sqrt{\hat{w}_i})^2 + (h_i - \sqrt{\hat{h}_i})^2 \right],$$

where S^2 is the number of grid cells, B is the number of bounding boxes per cell, and $\mathbf{1}_{ij}^{\text{obj}}$ an indicator function which indicates whether the j-th box in cell i contains an object in order to ensure loss is only tabulated over relevant grid cells.

Objectness and No-object Confidence Losses The object confidence loss evaluates the correctness of object presence predictions:

$$\mathcal{L}_{\text{obj}} = \sum_{i=1}^{S^2}\sum_{j=1}^{B} \mathbf{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2,$$

while the no-object confidence loss penalizes false positives:

$$\mathcal{L}_{\text{noobj}} = \sum_{i=1}^{S^2}\sum_{j=1}^{B} \mathbf{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2.$$

Here, C_i and $\hat{C}_i$ represent the ground truth and predicted confidence scores, and $\mathbf{1}_{ij}^{\text{noobj}}$ indicates cells without objects.

Classification Loss The classification loss assesses the accuracy of class probability predictions for cells containing objects:

$$\mathcal{L}_{\text{cls}} = \sum_{i=1}^{S^2} \mathbf{1}_i^{\text{obj}} \sum_{c \in C} (p_i(c) - \hat{p}_i(c))^2,$$

where C is the set of classes as defined in Figure 3.1, and $p_i(c)$ and $\hat{p}_i(c)$ are the ground truth and predicted probabilities for class c in cell i.

2.3.2 Performance Metrics

YOLO models make use of an array of robust metrics to properly capture model performance. In our work we focus on several primary metrics: Intersection over union, precision, recall, mAP50, and mAP50:95.

Intersection over union Intersection over union (IoU) is a measure of how well a model localizes target classes within an image. For a given prediction, the IoU is equal to:

$$IoU = \frac{|P \cap G|}{|P \cup G|}$$

where P is the set of pixels in the predicted region and G is the set of pixels in the ground truth region. IoU ranges from 0 to 1, with 1 indicating perfect localization and 0 indicating no overlap at all. For object detection, a prediction is typically considered correct if its IoU exceeds a predetermined threshold. For example, IoU@50 would only consider predictions above a threshold of 50% overlap.

Precision Precision quantifies the accuracy of positive predictions by measuring the proportion of true positive detections among all predicted detections. It is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

A true positive occurs when a predicted bounding box correctly identifies the object class and has an Intersection over Union (IoU) with a ground truth box exceeding a specified threshold. False positives arise from incorrect class predictions or insufficient IoU overlap with ground truth boxes.

Recall Recall measures the ability of the model to detect all relevant objects in an image. It is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

A false negative occurs when a ground truth object is missed or not detected by the model. High recall indicates that the model successfully identifies most objects, even if it produces some false positives. In practice, recall is often evaluated at different confidence thresholds to generate the precision-recall curve.

Mean Average Precision (mAP50) The mAP50 metric represents the mean Average Precision (AP) calculated at a fixed IoU threshold of 0.5. For each class, AP is determined as the area under the precision-recall curve:

$$\text{AP} = \int_0^1 p(r)\, dr$$

where $p(r)$ is the precision at a given recall level r. In practice, this integral is approximated by summing discrete points along the curve. The mAP50 is computed as:

$$\text{mAP50} = \frac{1}{|C|} \sum_{c \in C} \text{AP}_c^{50}$$

Here, C is the set of all classes, and AP_c^{50} denotes the AP for class c at an IoU threshold of 0.5.

mAP50:95 The mAP50:95 metric extends mAP by averaging the AP across multiple IoU thresholds, ranging from 0.5 to 0.95 in increments of 0.05. This metric provides a comprehensive assessment of detection quality, as it evaluates performance at varying levels of localization precision:

$$\text{mAP50:95} = \frac{1}{10} \sum_{t \in \{0.5, 0.55, \ldots, 0.95\}} \text{mAP}^t$$

where mAP_t represents the mean Average Precision at a specific IoU threshold t. By encompassing a broader range of thresholds, mAP50:95 rewards models capable of generating precise bounding boxes, even under more stringent conditions.

Collectively, these metrics provide an in depth evaluation of object detection performance.

Chapter 3

Methodology

3.1 Data

A defect dataset collated by Zhang et al. [24] was selected due to its relatively small size and high quality image content. The selected dataset contains a total of only 389 labeled images. Without augmentation of any kind, the selected dataset is insufficient for the prescribed task. Of the 389 available labeled images, 45 were randomly selected and set aside for testing, leaving a total of 344 labeled images for model training, roughly equivalent to a 90:10 train test split. The subject dataset contains four classes of common fuselage defects shown in Figure 3.1.

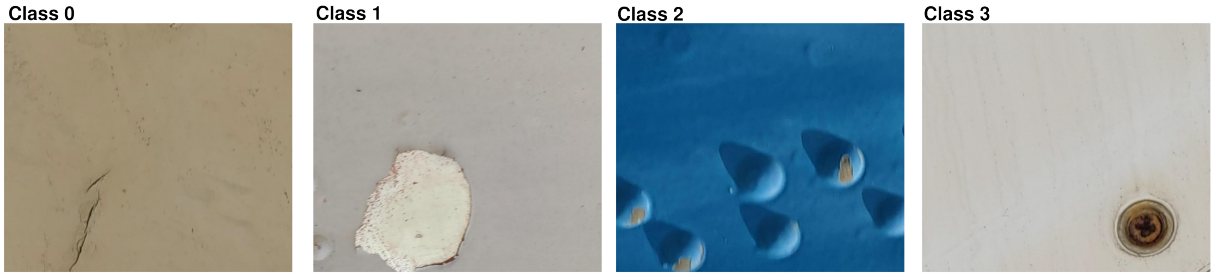
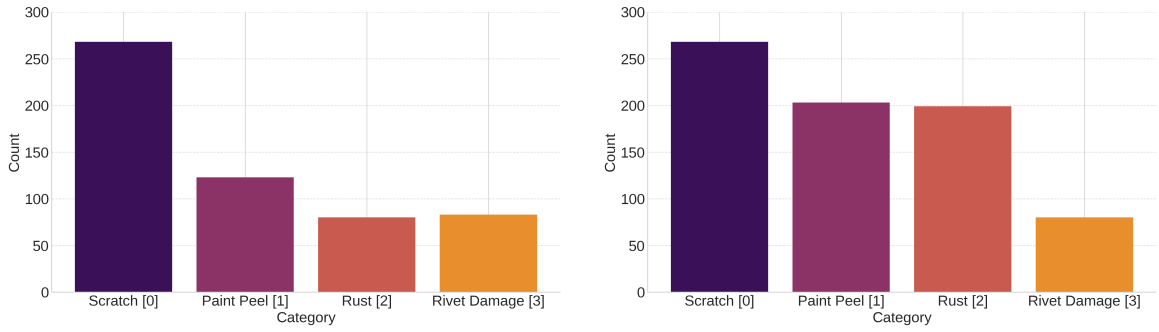


Figure 3.1: Example defects: Scratch, Paint-peel, Rust, and Rivet Damage. (left to right)

Our proposed workflow demonstrated varying effectiveness in forming realistic representations across different defect classes. Notably, we encountered limitations with Class 0 (Scratch) defects, where the creation of realistic samples proved challenging as defects were often only a few pixels across. Early attempts to model Class 0 yielded unstable and

unreliable performance indicating an inability for them to be represented meaningfully. As such they were disregarded for the purpose of this study, so that the effect of (easily) modeled defects could be studied in isolation. Interestingly, Class 3 defects demonstrated the highest performance in the absence of any augmentation. As such, it was decided that they be held out from the semi-synthetic dataset to study whether improved representations of Class 1 and 2 defects would effect the already reliable performance on Class 3.

The final semi-synthetic dataset contains 125 semi-synthetic images, consisting of both Class 1 and Class 2 defects in blended contexts. In total, the augmented dataset is approximately 36% larger than the baseline if all semi-synthetic images are included. A comparison of the frequency of each defect type in both the original and the augmented dataset can be seen in Figure 3.2.



(a) Base Dataset

(b) Augmented Dataset

Figure 3.2: Occurrences of defects broken down by class.

3.1.1 Data Generation

In this work we propose a novel form of semi-synthetic data augmentation tailored for aircraft defect detection tasks. By implementing a data generation workflow which leverages 3D assets in Unreal Engine coupled with existing defect features extracted from the parent dataset, we observe a more rapid improvement in performance in defect detection when compared to our baseline model. Specifically, the workflow enhances the speed at which surface defects such as rust, chipped paint, and rivet damage are able to be learned.

The process of data generation begins with sample preprocessing, where defects are selected from a limited training set based on image quality and defect type. While this human-in-the-loop defect selection process inevitably introduces bias to the training data—a common challenge in all 3D models with human creators—it also offers a

unique advantage. Through careful curation, human operators can effectively guide the model toward a better understanding of the target domain. This ability to impart semantic understanding through intentional sample creation, while likely infeasible for tasks with abundant training data, has been repeatedly demonstrated in Unreal Engine-based semi-synthetic modeling frameworks when working with sufficiently small datasets [5, 20, 21].

A second order consequence of the limited number of training samples in our dataset is that it precludes the possibility of training a segmentation algorithm for feature extraction. Thus defect features are manually segmented using open source photo editing software so that background context is removed. After defect preprocessing is complete, they can then be used to create augmented data in a workflow which they are combined with existing high quality 3D airplane assets available for use through the Unreal Engine (UE) marketplace.

The extracted defect features in .jpg format are ported to the Unreal Engine as 2D sprite objects, allowing them to be manipulated in the 3D environment, and subjecting them to the same lighting and environmental conditions as other objects in the scene. Once imported they can be used to texture existing 3D airplane assets (Figure 3.3), giving the illusion of material degradation.

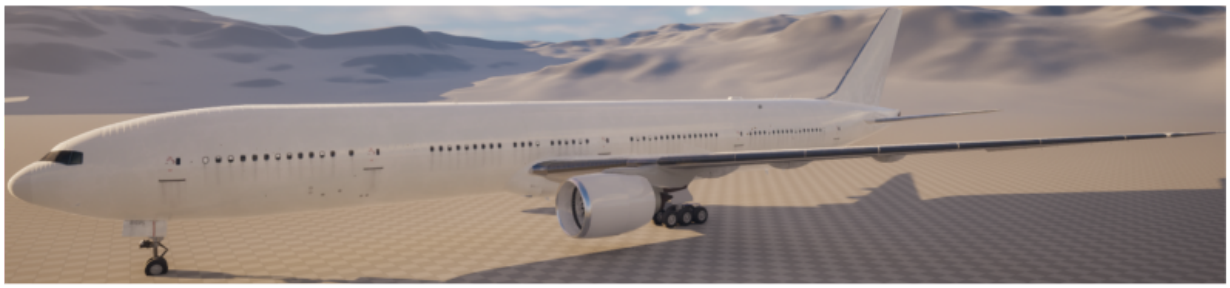


Figure 3.3: Example of high fidelity aircraft asset available in Unreal Engine.

Model texturing is done in a manner providing a wide range of patterns and combinations of realistic looking contextually relevant defects. Data is effectively augmented by varying defect scales, orientations and background context. Extending beyond conventional data augmentation, this semi-synthetic data generation workflow allows for the creation of unique training samples which could not otherwise be created as a result of applying any combination of conventional augmentations. This includes but is not limited to: scaling defects relative to their environment, increasing their number within a frame, or manipulating their orientation within a fixed background context. Once textured, the augmented vehicle assets are used to generate novel image data for training. This process mimics the way that data collection would occur in a real setting, with the defects be-

ing photographed within the development environment. This process is expedited using a modified camera actor asset in Unreal Engine (See Figure 3.4).

Figure 3.4: Camera actor positioned within the development environment, and what it sees.

A feature of this modified camera asset is that it is capable of automatically identifying the inserted defects based upon a user specified tag. This enables rapid creation of training samples, ensuring the correct bounding box with pixel precision, while generating both an image (.png) and label (.txt) file for export. By leveraging the ability to spatially navigate the environment within which the assets are contained, the user is able to capture many of the real phenomena that conventional data augmentation techniques attempt to mimic. The user is able to vary camera angles, zoom, rotation, lighting conditions, and background materials to create as many distinct training samples as necessary (Figure 3.5)

Figure 3.5: Example semi-synthetic samples.

This inherent modeling flexibility enables the creation of a dataset which is potentially much more thorough, encompassing broader distribution of potential defects otherwise unable be achieved through standard data augmentation techniques alone.

3.2 Model Design and Implementation

To evaluate the applicability of the proposed method and the synthetically augmented dataset, we employed a YOLOv5 architecture, a derivative model of the original YOLO single-stage detector. The YOLOv5 architecture utilizes a CSPDarknet53 backbone for feature extraction, with a Path Aggregation Network (PANet) for feature fusion across multiple scales. The model was implemented using the official YOLOv5 repository and tuned to address our four-class aircraft defect detection problem.

3.2.1 Strong Data Augmentation

The YOLO model family was selected because of its prevalence as a staple in defect detection literature, but also in large part because of its robust offerings in terms of built in data augmentation techniques. Augmentations serve a critical function in ensuring models learn robust and generalizable representations of the classes present in the training data. To evaluate our proposed augmentation technique, a series of strong augmentations is necessary to establish a performance baseline and ensure that observed gains are not attributed to learning simple invariances that could otherwise be achieved by augmenting the existing data. The applied augmentations include:

Mosaic Augmentation: Applied with a 100% probability, mosaic augmentation is one of the staple augmentations in the YOLO family of models. Mosaic works by randomly combining four images to form a composite training sample. A random crop sized for a standard input is taken from this composite to form a unique training sample, enhancing the model's ability to detect objects in diverse scenes. An example of one such sample is shown in Figure 3.6.

Random Horizontal Flipping: Images were flipped horizontally with a 50% probability to help the model learn invariances along the horizontal axis. This simple augmentation prevents the model from overfitting to object orientations ensuring robustness to reflection.

HSV Adjustments: Colors were transformed in HSV (Hue-Saturation-Value) color space, where hue represents the color type, saturation the color intensity, and value the brightness level. Specifically, hue was varied by $\pm 1.5\%$, saturation by $\pm 70\%$, and brightness by $\pm 40\%$. These adjustments enable the model to generalize across a broad spectrum

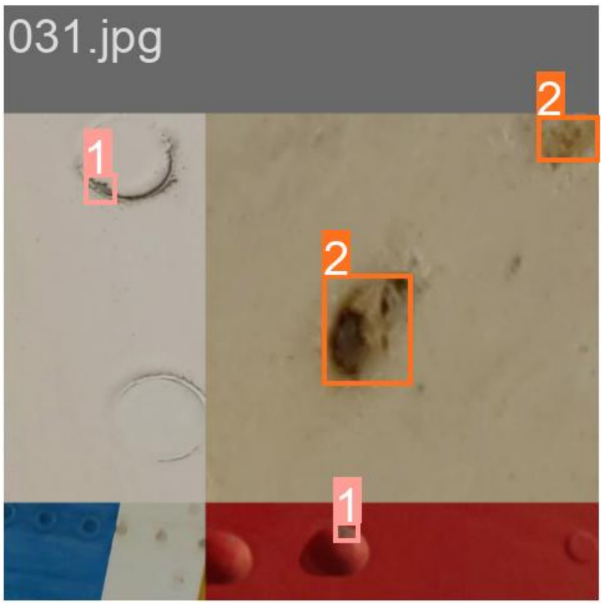


Figure 3.6: Example of Mosaic Augmented Training Sample.

of color and lighting variations, making it less sensitive to environmental conditions during inference.

Translation and Scaling: Random translations of ±10% were applied to simulate positional shifts, while scaling factors between 90% and 110% were used to mimic size variations. These transformations expose the model to changes in object positioning and scale, ensuring it learns spatially robust features.

Mixup Augmentation: Combines two images with a 10% probability by overlapping them to create a blended training sample. This augmentation introduces additional complexity by creating scenarios where features from multiple classes are present in the same image, forcing the model to learn to distinguish objects in challenging conditions.

Overall, this comprehensive augmentation strategy ensures the model is trained on a diverse and challenging dataset, promoting better generalization to unseen data and establishing a strong baseline for performance evaluation.

3.2.2 Model Configuration

To establish an optimal baseline model while adhering to computational constraints, hyperparameters were tuned on the baseline data. All experiments were executed on a single NVIDIA RTX 3070 GPU equipped with 8GB of VRAM under CUDA 12.4. The YOLOv5m architecture, selected as the foundation, operates at 47.9 GFLOPs during inference processing. Training utilized SGD optimization, implementing a 0.01 initial learning rate coupled with 0.937 momentum, while overfitting was mitigated through a weight decay factor of 0.0005.

Initial Learning Rate	Batch Size
0.001	4
0.01	8
0.1	**10**

Table 3.1: Experimented learning rates and batch sizes. Note: bold indicates optimal result.

The architectural design incorporates the established YOLO backbone framework, where input traverses through a series of convolutional layers integrated with feature pyramid networks. The model's detection mechanism was specifically adapted for 4-class classification, implementing a tri-scale detection system with variable anchor box configurations. The architecture combines conventional convolution operations with C3 bottleneck blocks and SPPF modules, systematically processing 640×640 pixel inputs through progressively deeper feature maps, scaling from 48 to 768 channels.

Model evaluation occurred through periodic assessment on an isolated test dataset, ensuring unbiased performance metrics throughout training. To maintain experimental integrity, both the baseline and synthetic-augmented variants were initialized with identical hyperparameter configurations, thereby ensuring that performance variations could be directly attributed to the incorporation of semi-synthetic training data.

3.3 Experimental Setup

The first step in our study involved establishing a robust performance baseline. For this, we conducted extensive training of models using the standard dataset exclusively. To determine the number of epochs necessary to train each baseline model to a performance

plateau, we performed exploratory training runs, closely monitoring the validation loss and mean average precision (mAP) metrics across epochs. The point of convergence was defined as the epoch at which mAP50 exhibited no further improvement over 5 epochs. For the baseline model, this plateau was established at 60 Epochs. This iterative process ensured that all baseline models were trained optimally while avoiding unnecessary overfitting while remaining cognizant of computing constraints.

To assess the relative performance of models trained using the augmented dataset, we performed a two stage validation:

- **Full Dataset Augmentation (100%)**: In the first phase, the standard dataset was augmented with the complete semi-synthetic dataset. This resulted in a training set consisting of 469 images (344 base, 125 semi-synthetic). This was done to explore whether increasing the volume of semi-synthetic samples provides additional benefit in terms of convergence speed and model performance.
- **Fractional Dataset Augmentation (50%)**: In the second phase, the standard dataset was augmented with a randomly selected 50% partition of the available semi-synthetic data amounting to 406 images (344 base, 62 semi-synthetic). This was done to assess the effects of reducing the number of semi-synthetic samples on model performance.

In total, 30 uniquely seeded and distinct models were trained: 10 to establish a robust baseline, 10 with complete access to both the baseline and available semi-synthetic data, and 10 with both the baseline data and 50% of the full semi-synthetic dataset.

To quantify the impact of semi-synthetic data on convergence rates, we measured the number of epochs and total number of samples were processed for each model to reach optimal performance. These metrics were averaged across the 10 models in each configuration, with 95% confidence intervals computed to assess variability. In depth comparisons of key performance metrics (precision, recall, mAP50, and mAP50:95) were conducted across model configurations to confirm the significance of observed phenomena (See Tables 4.3,4.4).

By training multiple models with unique random seeds and analyzing both aggregate metrics and statistical variability, we investigated how semi-synthetic data volume impacted trends in learning dynamics, convergence rates, and final performance outcomes. Through comparing different semi-synthetic data volumes, we observed that metrics improved at a similar rate using just 50% of the available semi-synthetic data, suggesting that relatively few semi-synthetic samples are needed to achieve improved convergence

Chapter 4

Results

4.1 Complete Semi-Synthetic Dataset

Initial experiments using the full enhanced dataset indicated the presence of interesting learning dynamics in the enhanced model. Figures 4.1 and 4.2 demonstrate that while ultimately achieving comparable final performance to the baseline model, models trained with full access to semi-synthetic data required fewer epochs to converge. This observation in improved convergence speed prompted further investigation into whether similar convergence results could be achieved with a reduced fraction of the synthetic dataset. To explore this, we opted to train a series of models using approximately 50% of the available semi-synthetic data.

4.2 Data Efficiency

After observing accelerated convergence dynamics with the full semi-synthetic dataset, we investigated whether similar benefits could be achieved with fewer synthetic samples. By training models with approximately 50% of the available semi-synthetic data (62 images) we found that improved convergence dynamics persisted without degradation. As shown in Figures 4.3 and 4.4, both models take a similar amount of time to match the optimal performance of the baseline model.

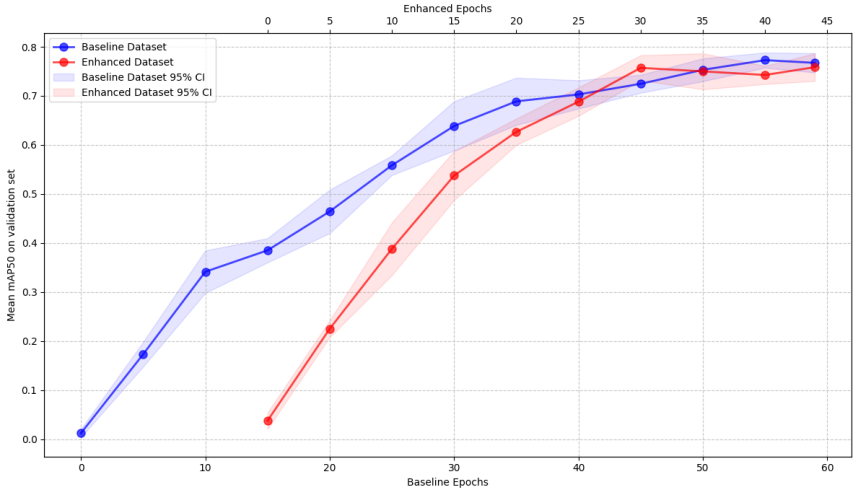


Figure 4.1: Validation mAP50 for both the Baseline and Enhanced Dataset, demonstrating accelerated learning dynamics for the enhanced model.

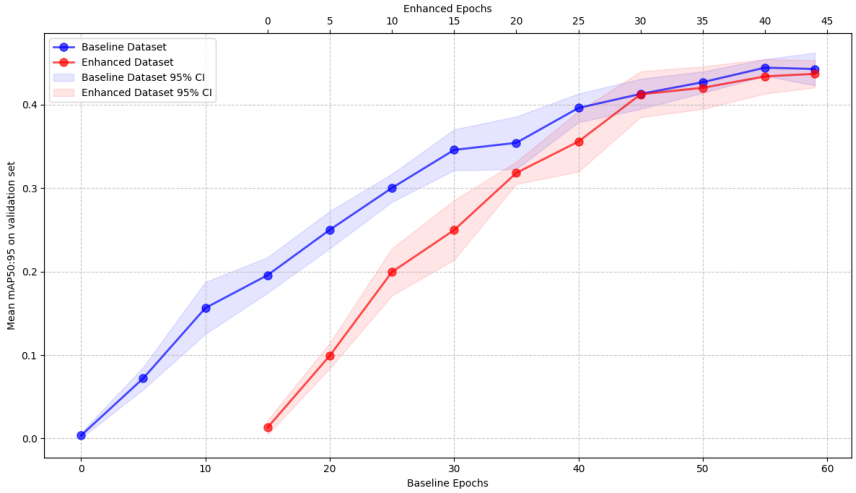


Figure 4.2: Validation mAP50:95 for both the Baseline and Enhanced Dataset, further demonstrating accelerated learning dynamics associated with the enhanced model.

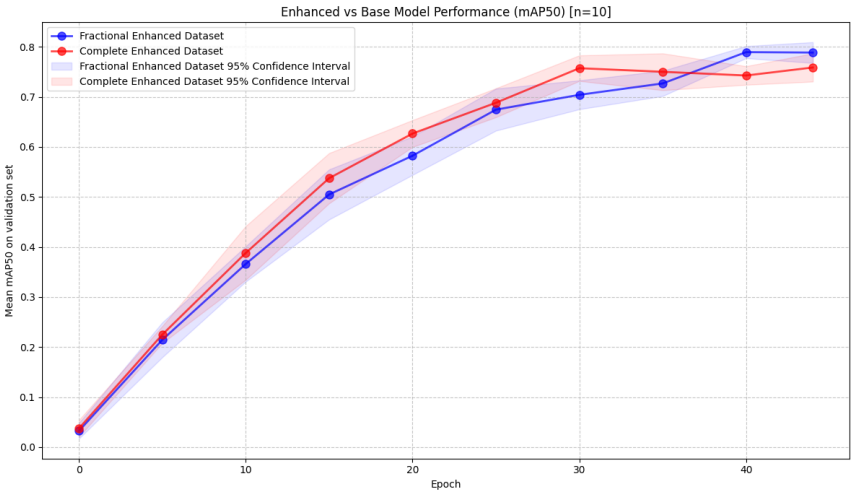


Figure 4.3: Validation mAP50 for both the Fractional and Fully Enhanced Dataset, demonstrating comparable learning dynamics with reduced data.

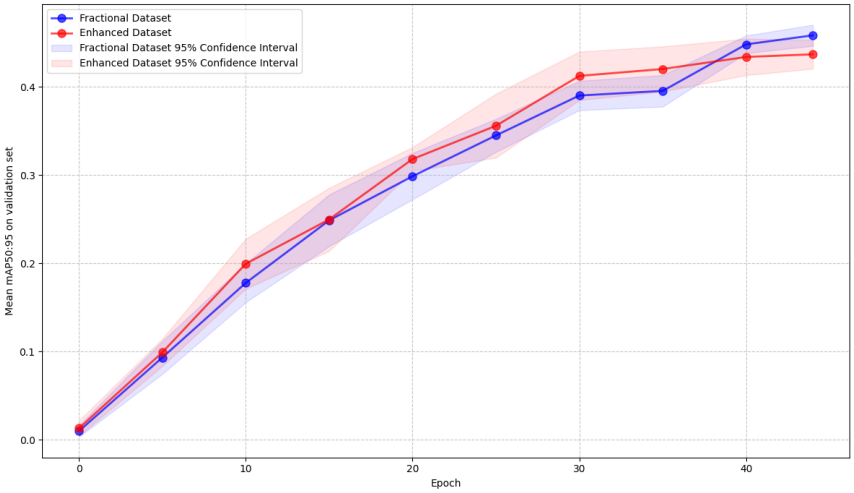


Figure 4.4: Validation mAP50:95 for both the Fractional and Fully Enhanced Dataset, demonstrating comparable learning dynamics with reduced data.

4.2.1 Convergence Rate

To assess the convergence rate of our fractional enhanced models, we established a benchmark based on the optimal performance of each individual model in the sample group. For this purpose we define convergence as the point at which the model achieves its best mAP50 result. By calculating the mean number of epochs required to reach this threshold, we can compare the rate at which respective model approaches its optima. Our convergence assessment, shown in Table 4.1 reinforces what is observed in our validation curves, with our enhanced model converging more rapidly than the base model by 24.1% measured in epochs, and 10.4% in terms of total number of images processed.

	Base Model	**Enhanced Model**	**Difference (%)**
Epochs to Optima	53.9 ± 2.69	40.9 ± 3.31	-24.1%
Total Samples Processed	74168	66408	-10.5%

Table 4.1: Convergence analysis of models showing epochs and samples required to reach optimal performance

The stark difference in convergence speeds indicates some interesting properties of our semi-synthetic data. In the presence of semi-synthetic samples, our models need to see the true data 24.1% less to learn a comparable representation. Even if we take all images as being equal, in terms of the total number of data observed our enhanced models still need nearly 8000 fewer images on average to match or exceed the performance of baseline models. This enhanced rate of convergence can be further visualized by comparing the validation metrics of both the standard and fractionally enhanced models in terms of their mAP50 and mAP50:95 respectively. Figures 4.5 and 4.6 both demonstrate the same accelerated rate of convergence observed in the fully enhanced model.

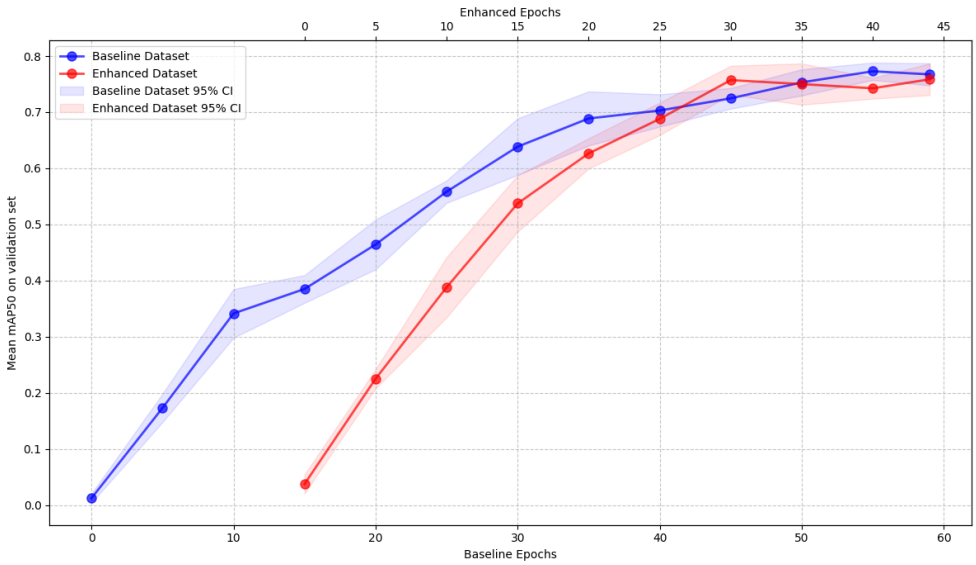


Figure 4.5: mAP50 convergence for both the enhanced and base model.

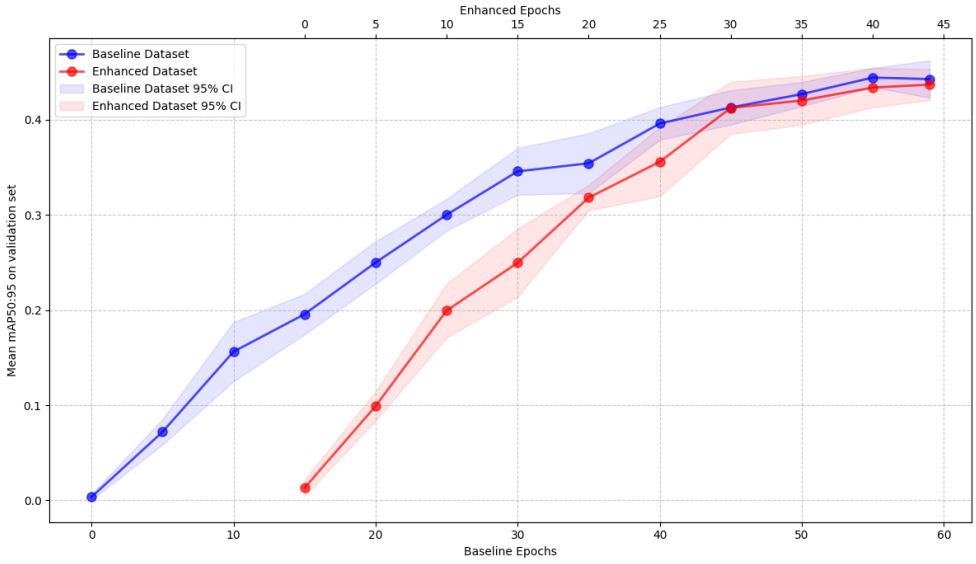


Figure 4.6: mAP50:95 convergence for both the enhanced and base model.

4.2.2 Learning Dynamics

Analysis of the validation metrics revealed interesting patterns in how synthetic data influenced different aspects of model learning. Most notably, the models trained on synthetic samples converged more quickly to their optima, indicating that the presence of synthetic samples improved the rate at which models were able to learn accurate representations of the included classes.

The objectness loss, responsible for measuring the model's ability to distinguish objects from background, and the box loss, evaluating localization precision, both remained consistent between the base and enhanced models. This absence of change indicates that our observed improvements were not a result of performance changes in these categories. This consistency suggests that synthetic data neither improved nor impaired the model's fundamental ability to locate defects and separate them from their surroundings.

Interestingly, the classification loss showed a notable difference between the models. The enhanced model demonstrated overall a marginally lower classification loss, indicating an improved ability to discriminate between classes present in the dataset. This improvement is particularly interesting given the way that our dataset was constructed. Although our semi-synthetic samples used the same defect instances as the original dataset, merely placing them in different contexts, this contextual variety appears to have enhanced the model's learning. While the semi-synthetic samples may not have captured novel characteristics defects outside the distribution present in the training set, the variety of contextually relevant backgrounds and mixing of defects appears to have helped the model develop more robust class representations. To illustrate these findings, Figures 4.7 4.8 & 4.9 present the mean values for each metric across each of the 10 models trained on the fractionally enhanced and the baseline dataset, accompanied by their 95% confidence intervals.

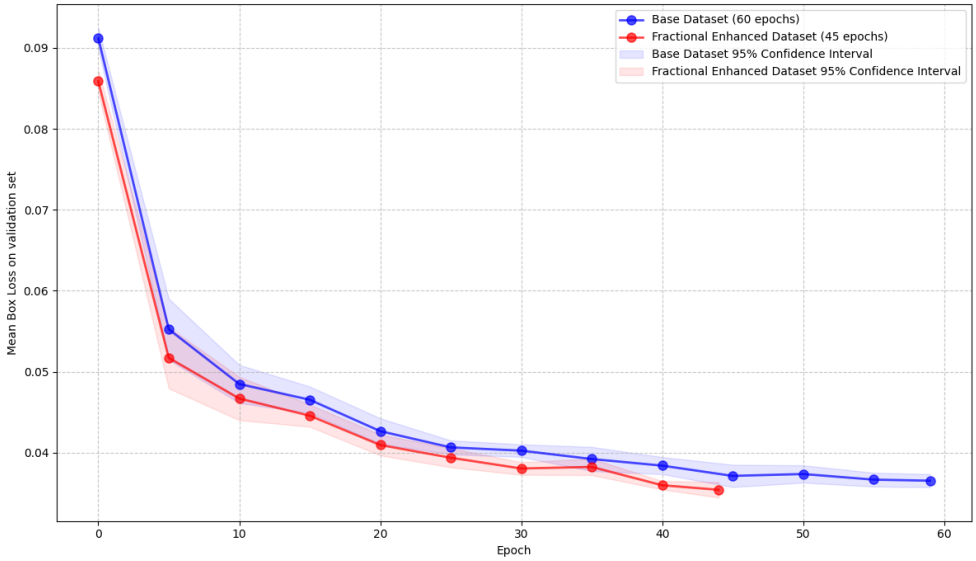


Figure 4.7: Mean bounding box regression loss measured across across base and fractional datasets.

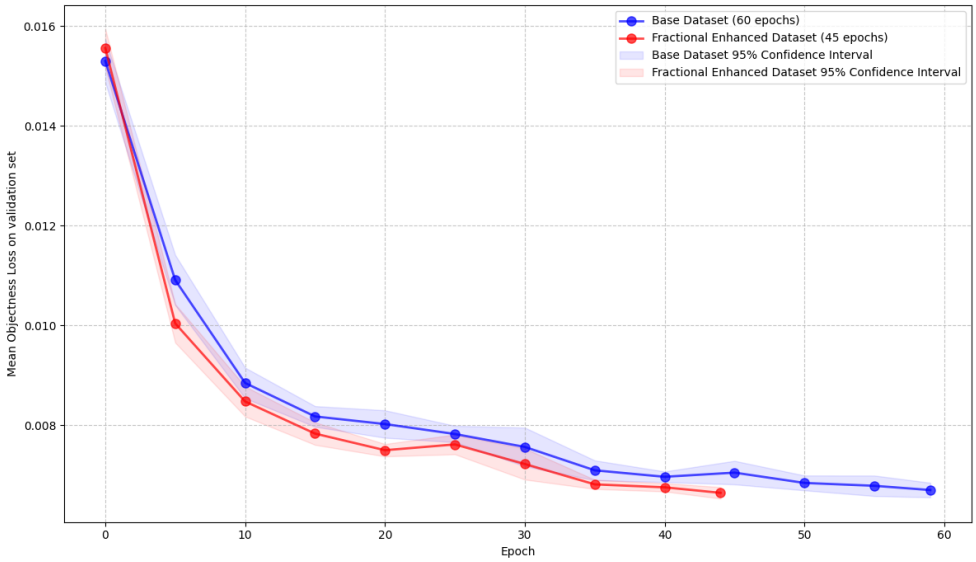


Figure 4.8: Mean objectness loss measured across across base and fractional datasets.

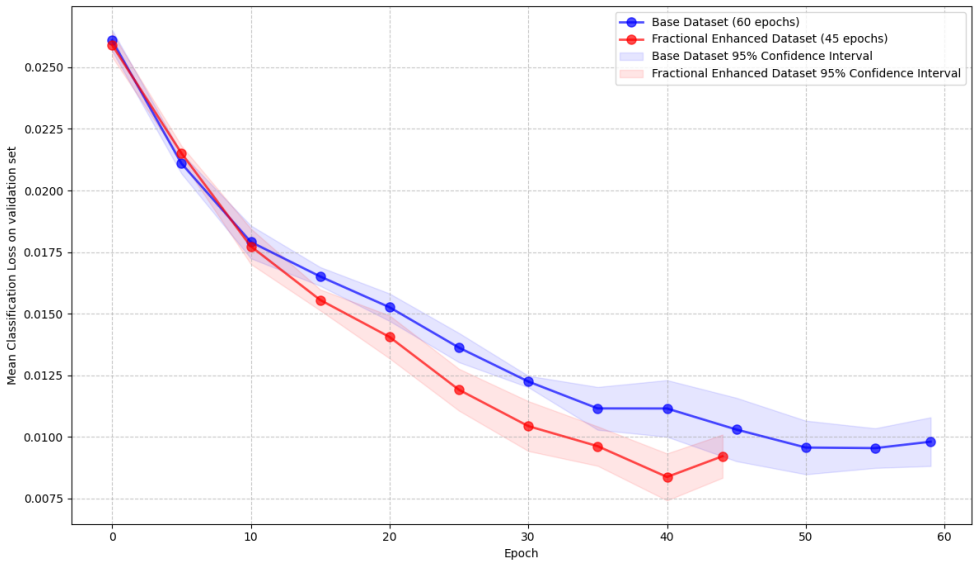


Figure 4.9: Mean classification loss measured across base and fractional datasets.

4.3 Comprehensive Results

4.3.1 Early Stage Performance Advantage

Since the enhanced model was trained on a larger dataset (as it included semi-synthetic data) comparing models based on the number of epochs alone would not provide an ideal assessment of performance on a value per sample basis. Instead, to evaluate the impact of incorporating semi-synthetic data, we controlled for the total number of training samples seen by each model. We trained ten base models using the same number of total training samples as the enhanced model at its optimal performance level. These early stage performance results (shown in Table 4.2) indicate that the inclusion of our semi-synthetic data may present a more readily learnable feature space for our problem.

4.3.2 Optimal Performance Comparison

At optimal performance (Table 4.3) the enhanced model achieved 79.10% precision (+3.2 percentage points). The mAP50 showed a small improvement of 0.4 percentage points in favor of the enhanced model (79.5% vs 79.1%), and both Recall and mAP50:95 were higher for the base model (72.7% vs 70.5% & 47.0% vs 46.6% respectively).

Metric	Base Model	Enhanced Model	Diff
P	70.39 ± 3.89	**79.10** ± 4.80	+8.71
R	67.51 ± 7.42	**70.50** ± 7.40	+2.99
mAP50	73.90 ± 4.32	**79.50** ± 2.40	+5.60
mAP50 : 95	41.64 ± 3.08	**46.60** ± 1.20	+4.96

Table 4.2: Overall performance comparison with early stopping. Note: bold indicates optimal results.

Metric	Base Model	Enhanced Model
mAP50	79.1 ± 1.7	**79.5 ± 2.4** (+0.4)
mAP50:95	**47.0 ± 1.9**	46.6 ± 1.2 (-0.4)

Table 4.3: Overall performance comparison at optima.

Class-specific analysis (Table 4.4) revealed varied performance across defect types. For paint peel defects, the enhanced model achieved higher precision (81.4% vs 74.3%, +7.1 percentage points) and mAP50 (77.1% vs 75.1%, +2.0 percentage points). Similarly, for rust defects, the enhanced model showed improved recall (81.4% vs 75.4%, +6.0 percentage points) and mAP50 (86.5% vs 83.1%, +3.4 percentage points). For the remaining defect categories, scratch and rivet damage, the baseline model achieved higher performance across most metrics.

Category	**Metric**	**Base Model**	**Enhanced Model**
Scratch	P	**63.3**	61.9 (-1.4)
	R	**56.9**	49.3 (-7.6)
	mAP50	**62.2**	59.9 (-2.3)
	mAP50:95	**32.9**	31.4 (-1.5)
Paint Peel	P	74.3	**81.4** (+7.1)
	R	**64.5**	62.5 (-2.0)
	mAP50	75.1	**77.1** (+2.0)
	mAP50:95	49.5	**50.6** (+1.1)
Rust	P	**81.6**	81.5 (-0.1)
	R	75.4	**81.4** (+6.0)
	mAP50	83.1	**86.5** (+3.4)
	mAP50:95	40.6	**41.1** (+0.5)
Rivet Damage	P	84.2	**91.8** (+7.6)
	R	**93.9**	88.8 (-5.1)
	mAP50	**96.1**	94.6 (-1.5)
	mAP50:95	**65.0**	63.2 (-1.8)

Table 4.4: Performance comparison of base and enhanced models by defect type.

Chapter 5

Discussion

Analysis of the enhanced and baseline models provided significant insight into how semi-synthetic data of this nature impacts training dynamics. Interestingly, the enhanced model shows significantly improved early-stage performance, with improvements in precision (+8.71 percentage points), mAP50 (+5.60 percentage points) and mAP50:95 (+4.96 percentage points) when models are compared after having processed the same number of training samples.

Both models ultimately achieve similar performance in aggregate, though the model trained on semi-synthetic data required fewer training iterations to do so. Performance in the enhanced models varied across defect types. The enhanced model performed particularly well with paint peel detection (precision +7.1 percentage points) and rust identification (recall +6.0 percentage points), while showing slightly decreased performance in scratch detection across all metrics. Notably, the enhanced models demonstrated substantially increased precision for rivet damage defects, suggesting that contextually relevant semi-synthetic samples of both rust and paint peel defects led to more discriminative learning. This increased precision came at a small cost to recall, resulting in minor decreases in mAP50 and mAP50:95 for this class.

The empirical evidence for enhanced learning when given access to semi-synthetic samples is compelling: models trained with additional samples just 18% of the original dataset size were able to achieve performance equivalent to our benchmark whilst requiring 24.1% fewer epochs and 10.5% fewer total samples. This accelerated convergence suggests that our semi-synthetic approach may create a more structured and navigable feature space compared to traditional strong augmentation techniques alone.

Class specific performance dynamics indicate that classes able to be effectively repre-

sented in the semi-synthetic dataset reaped the most significant gains, further supporting our enhanced learnability hypothesis. Substantial early gains in precision specifically suggest that our contextually relevant augmentation manages to distill essential class features responsible for making classes more readily distinguishable.

Chapter 6

Conclusion

Our work demonstrates that integrating contextually appropriate semi-synthetic data into aircraft defect detection models offers clear benefits, including accelerated learning and enhanced class-specific detection capabilities while preserving overall performance.

By blending real defect features with high-fidelity 3D assets in Unreal Engine, our hybrid dataset uniquely introduced contextually relevant sample variety capable of improving a broad scope of learning dynamics. By demonstrating both fewer epochs and samples required to reach optimal performance, we give insight into how our unique augmentation strategy not only introduces valuable sample variety improving class-specific performance, but also creates a more learnable class representation capable of facilitating faster and more efficient model optimization.

Looking ahead, there remain several opportunities for further refinement. Provided a larger base dataset, automating defect extraction through segmentation models with a human in the loop could reduce the upfront cost of manual preprocessing and improve both scalability and the ability to rapidly prototype. Exploring further techniques in the domain of 3D modeling to specifically address challenging defect types (e.g. scratches) may further boost model performance.

Our results demonstrate promising potential for leveraging semi-synthetic data for aircraft defect detection. Though this work represents just an initial step toward addressing the broader challenges of data scarcity in safety-critical domains, the observed improvements in both training efficiency and class-specific performance suggest potential directions for future research employing hybrid assets in data scarce domains

References

[1] Nicolas P. Avdelidis, Antonios Tsourdos, Pasquale Lafiosca, Richard Plaster, Anna Plaster, and Mark Droznika. Defects recognition algorithm development from visual UAV inspections. *Sensors*, 22(4682), 2022.

[2] Marie-Anne Bauda, Alex Grenwelge, and Stanislas Larnier. 3d scanner positioning for aircraft surface inspection. In *ERTS 2018*, Toulouse, France, 2018.

[3] Wenbin Chen and Siew Huang. Human reliability analysis for visual inspection in aviation maintenance by a bayesian network approach. *Transportation Research Record*, 2449(1):105–113, 2014.

[4] I. L. Davis and M. Siegel. Automated nondestructive inspection of aging aircraft. *Measurement and Control in Intelligent Instruments*, pages 190–201, 1993.

[5] César Roberto de Souza, Adrien Gaidon, Yohann Cabon, Naila Murray, and Antonio Manuel López. Generating human action videos by coupling 3D game engines and probabilistic graphical models. *International Journal of Computer Vision*, 2019. Special Issue on Generating Realistic Visual Data of Human Behavior.

[6] B. S. Dhillon. Human reliability and error in transportation systems. In *Springer series in reliability engineering*. Springer, London, 2007.

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *1st Conference on Robot Learning (CoRL 2017)*, Mountain View, United States, 2017.

[8] Matteo Fabbri, Guillem Brasó, Guglielmo Maugeri, Onur Cetintas, Riccardo Gasparini, Aljoša Ošep, Simone Calderara, Laura Leal-Taixé, and Rita Cucchiara. Motsynth: How can synthetic data help pedestrian detection and tracking? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10849–10859, 2021.

[9] José Luis Salazar González, C. Zaccaro, Juan Antonio Álvarez-García, Luis Miguel Soria Morillo, and Fernando Sancho Caparrini. Real-time gun detection in cctv: An open problem. *Neural Networks*, 132:297–308, 2020.

[10] Anand K. Gramopadhye and Colin G. Drury. Human factors in aviation maintenance: how we got to where we are. *International Journal of Industrial Ergonomics*, 26(2):125–131, 2000.

[11] Bin Huang, Yan Ding, Guoliang Liu, Guohui Tian, and Shanmei Wang. ASD-YOLO: An aircraft surface defects detection method using deformable convolution and attention mechanism. *Measurement*, 238:115300, 2024.

[12] Adam Kortylewski, Andreas Schneider, Thomas Gerig, Bernhard Egger, Andreas Morel-Forster, and Thomas Vetter. Training deep face recognition systems with synthetic data, 2018.

[13] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*. NVIDIA, 2017.

[14] Hyunwoo Lee, Jaeyoung Jeon, Daewon Lee, Chulwoo Park, Jonghun Kim, and Donghyun Lee. Game engine-driven synthetic data generation for computer vision-based safety monitoring of construction workers. *Automation in Construction*, 155:105060, 2023.

[15] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[16] Yuanpeng Liu, Jingxuan Dong, Yida Li, Xiaoxi Gong, and Jun Wang. A UAV-based aircraft surface defect inspection system via external constraints and deep learning. *IEEE Transactions on Instrumentation and Measurement*, 71:5019315, 2022.

[17] Ka Man and Javaan Chahl. A review of synthetic image data and its use in computer vision. *Journal of Imaging*, 8(11):310, 2022.

[18] Reddy Mandati, Vladyslav Anderson, Po-Chen Chen, Ankush Agarwal, David Barnard, Michael Finn, Jesse Cromer, Tatjana Dokic, Andrew McCauley, Clay Tutaj, Neha Dave, Bobby Besharati, Jamie Barnett, and Timothy Krall. Integrating artificial intelligence models and synthetic image data for enhanced asset inspection and defect identification. 2024.

[19] P. Martinez-Gonzalez, Sergiu Oprea, A. Garcia-Garcia, Alvaro Jover-Alvarez, S. Orts-Escolano, and J. Garcia-Rodriguez. Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. *Virtual Reality*, 24:271–288, 2018.

[20] Augusto J. Peterlevitz, Mateus A. Chinellatto, Angelo G. Menezes, Cézanne A. M. Motta, Guilherme A. B. Pereira, Gustavo L. Lopes, Gustavo M. de Souza, Juan Rodrigues, Lilian C. Godoy, Mario A. F. Koller, et al. Sim-to-real transfer for object detection in aerial inspections of transmission towers. *IEEE Access*, 2023.

[21] Thomas Pollok, Lorenz Junglas, Boitumelo Ruf, and Arne Schumann. Unrealgt: Using unreal engine to generate ground truth datasets. In *ISVC 2019, LNCS 11844*, pages 670–682. Springer, 2019.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2016.

[23] Yuri D. V. Yasuda, Fabio A. M. Cappabianco, Luis Eduardo G. Martins, and Julio A. B. Gripp. Aircraft visual inspection: A systematic literature review. *Computers in Industry*, 141:103695, 2022.

[24] Xiang Zhang, Jing Zhang, Jian Chen, Ruipeng Guo, and Jun Wu. Aircraft_fuselage_det2023: An aircraft fuselage defect detection dataset, 2023. IEEE Dataport.

[25] Xiaoyu Zhang, Jinping Zhang, Jiusheng Chen, Runxia Guo, and Jun Wu. A semisupervised aircraft fuselage defect detection network with dynamic attention and class-aware adaptive pseudolabel assignment. *IEEE Transactions on Artificial Intelligence*, 5(7):3551, July 2024.