Aircraft Fuselage Defect Detection Using CNN

Individual Final Report
Adam O'Connor

ECEN 5060: Deep Learning
Dr. Hagan
Due Date: May 6, 2025

1. Introduction

This report outlines my individual contributions to the group project titled "Aircraft Fuselage Defect Detection Using CNN" completed for ECEN 5060: Deep Learning, under the supervision of Dr. Hagan. The goal of the project was to build a computer vision model capable of identifying defects in aircraft fuselage images using a convolutional neural network (CNN). While the project was a collaborative effort, this report focuses on the tasks and technical responsibilities I completed independently.

2. Description of My Individual Work

2.1 Masked Dataset Loading

My main individual contribution was customizing the dataset loading process to support masked image training. The aim was to ensure that only relevant, labeled regions were shown to the model during training to reduce false negatives caused by unlabeled but visible defect classes in the same image. To achieve this, I extended the PyTorch `Dataset` class by overriding the `__init__()` and `__getitem__()` methods in our `CustomDataset`. I used the bounding box annotations provided in the COCO-style JSON files to mask out everything in the image except the labeled regions. The masking was implemented by blacking out the image and selectively copying over the bounding box areas from the original image. Each time a training image was loaded, the bounding box was randomly expanded and its position slightly altered to prevent the network from learning bounding box shapes rather than class-specific features.

2.2 Augmentation Pipeline

We applied aggressive augmentations such as random rotation up to 90°, resized cropping, and color jittering. All classes were masked consistently, even those like 'scratch' that appeared cleanly labeled, to avoid the model learning to associate unmasked images with specific classes. This masking and augmentation technique proved essential in improving model generalization, particularly for underrepresented classes like 'paint_peel'.

2.3 Visualization Tool

In addition to preprocessing and model development, I implemented a utility function named `view_image()` that allowed us to visualize how PyTorch was interpreting and transforming our dataset during training. This function sampled an image after it had passed through our full augmentation and masking pipeline, overlaid the image ID and associated label(s), and saved the result as `sample_image.png`. This let us confirm that the masked bounding regions were correctly applied, rotations and crops varied randomly, color and aspect ratio augmentations were visible, and label assignments matched the visual data. This visualization tool helped us debug and validate the pipeline and ensured the final model inputs retained class-discriminative features.

3. Detailed Contributions

Contribution	Description
CNN architecture (`model.py`)	Designed and implemented the core CNN model
Dataset loader (`dataset_loader.py`)	Customized dataset loading with bounding box-based masking
Data augmentation	Developed rotation, cropping, and color jittering strategies
Hyperparameter tuning	Tuned model settings and managed dataset balancing
Debugging and analysis	Investigated class imbalance, improved performance on 'paint_peel'

4. Results

Below is a visual showing how our preprocessing pipeline was applied to an original image containing the 'paint_peel' defect. The left side shows the original image, while the right displays several randomized, masked, and augmented versions of the same image. These transformations helped train the model to recognize 'paint_peel' across a wider variety of appearances, camera angles, and contexts.

When we applied this masked augmentation strategy in combination with the final model architecture—which included three parallel convolutional kernels at the input layer—we saw a noticeable improvement in the prevalence and confidence of 'paint_peel' detections when labeling previously unlabeled data. This suggests the combination of masked input data and diverse feature extraction helped the model generalize more effectively to this difficult class.

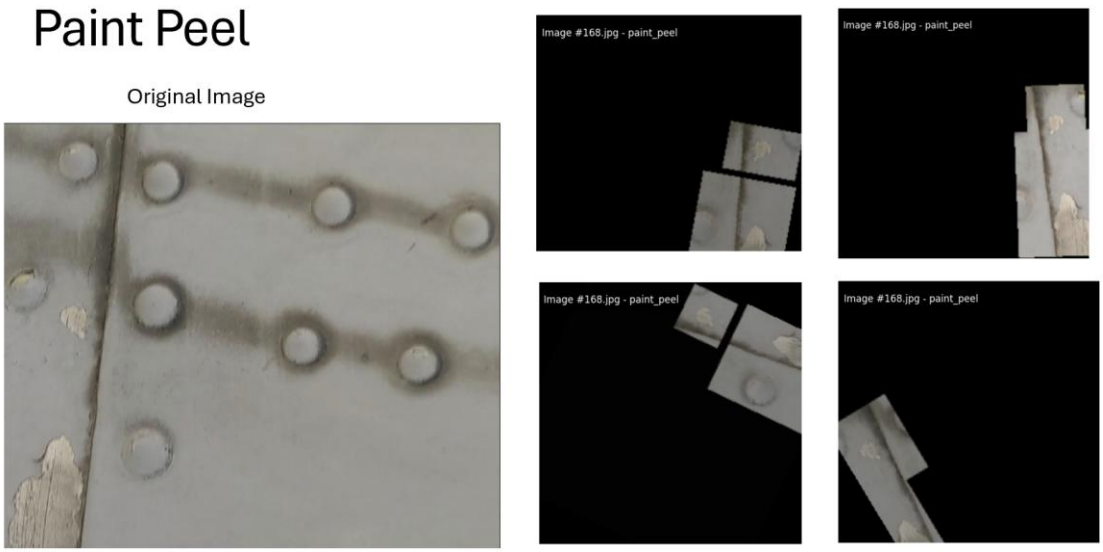

Figure 1 Augmented Image for Paint Peel

5. Summary and Conclusions

Through this work, I gained valuable experience in designing custom dataset pipelines and augmentations in PyTorch. The dynamic masking approach helped balance label quality and training accuracy, and the layered augmentations ensured robustness.

Key Takeaways:

- Custom masked preprocessing improved detection of rare defect classes

- Visualization helped debug and confirm preprocessing quality

- Augmentation strategy prevented overfitting to spatial patterns

Future Improvements:

- Explore semi-supervised learning from unlabeled regions

- Introduce bounding box regression for localization accuracy

6. References

- PyTorch Documentation – https://pytorch.org/docs/stable/index.html
- ECEN 5060 Image Recognition Competition 2 Starter Code (Spring 2025)