


Semester (Term, Year)	Fall, 2024
Course Code	AER 850
Course Section	01
Course Title	Intro to Machine Learning
Course Instructor	Dr. Reza Faieghi
Submission	Project Report 2
Submission No.	1
Submission Due Date	November 17 th 2024
Title	Project 2 – Deep Convolution Neural Networks
Submission Date	November 17 th 2024

Submission by (Name):	Student ID (XXXXXX1234)	Signature
Kiran Patel	501024568	

By signing the above you attest that you have contributed to this submission and confirm that all work you contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, and "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Academic Integrity Policy 60, which can be found at www.torontomu.ca/senate/policies/

Table of Contents

1. GitHub Submission Link	2
2. Introduction	2
3. Results and Discussions	3
4. Conclusion	7

List of Figures

Figure 1. Training and Validation Plots of Accuracy and Loss.....	3
Figure 2. Code Stopping at 14 out of the 50 epochs set	5
Figure 3. Crack Image Classification Test Result	5
Figure 4. Missing Screw Head Classification Test Result.....	6
Figure 5. Paint-off Classification Test Result.....	6

1. GitHub Submission Link

The Following GitHub Link can be used to access the code for the project:

https://github.com/kirankiran22/AER850_Project2_Kiran

2. Introduction

This project explores the use of Deep Convolutional Neural Networks (DCNNs) to automate the detection of typical aircraft surface defects, including cracks, missing screw heads, and paint degradation. Regular inspections are essential and mandatory to maintain the structural integrity and safety of aircraft, but manual inspection methods can be time-consuming, have human error, and be unsafe. By applying machine learning, this project aims to improve the efficiency and accuracy of defect detection and reduce the need for human involvement in the maintenance and inspection process.

The development process involved five key steps. First, preparing the dataset by applying data augmentation and organizing it into training, validation, and test sets. This was done by resizing the test images and using Keras's image preprocessing methods to format the dataset for the CNN model to train from. In the second step of the project, a custom Deep Convolution Neural Network was created to learn features from the images. In the third step, optimizing hyperparameters such as filter sizes, and kernels were done to enhance the CNN model performance. Then the model was evaluated by analyzing accuracy and loss metrics to minimize overfitting to the trained dataset. Finally, testing was done on the trained model by passing unseen images to assess its performance in identifying a crack, a missing screw head, and paint scratched off a surface.

3. Results and Discussions

This section summarizes the results obtained from the trained CNN and challenges faced in developing the Machine Learning Algorithm. First the Training and Validation Accuracy of the model is provided alongside the Training and Validation Loss plots are provided below in Figure 1. The plots visualize the training and validation performance results over 14 epochs. It should be noted that in the GitHub code, it will be seen the CNN was supposed to be run for 50 epochs, however, due to hardware limitations, and the long epoch running times, the code ended early without running for all 50 epochs. This issue can be resolved in the future by using Google Colab to run the intensive processing power required for a computer to iterate over a large dataset as used in the project. Figure two below shows the console output of the program stopping after 14 epochs. It can also be seen that each epoch took on average about, 1 hour and 40 minutes to run, which mean the model was training for about 22 hours.

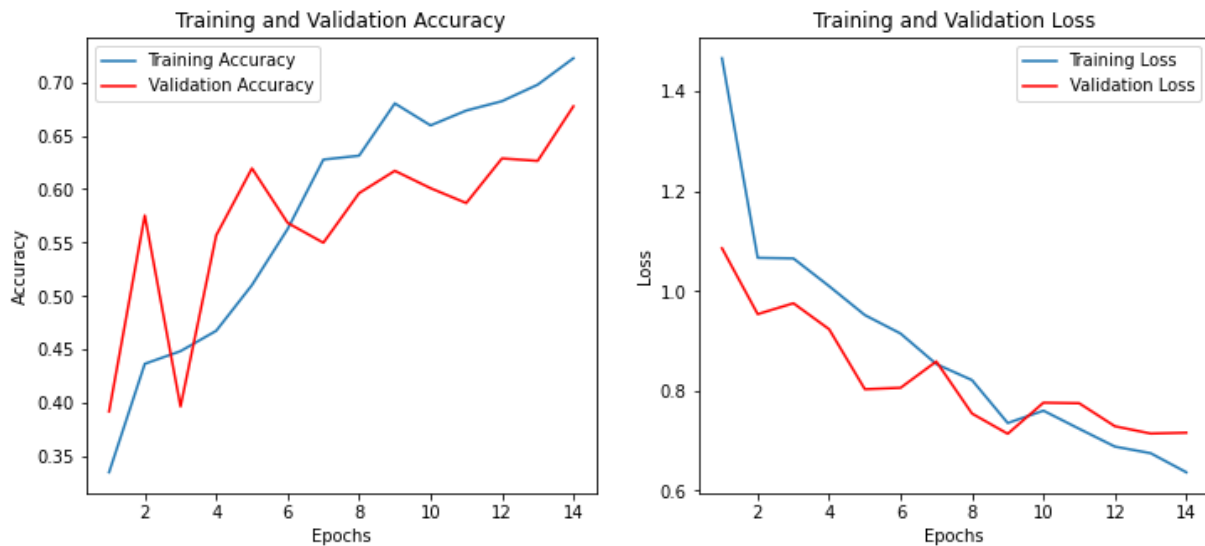


Figure 1. Training and Validation Plots of Accuracy and Loss

For the results obtained in figure 1, it can be seen in the left-hand plot, The training accuracy started around 35% accuracy and increased to over 70%. This line indicated the model performance with the training data. It gradually improved its ability to make correct predictions over the 14 epochs to have a satisfactory predictive accuracy. The red Validation accuracy line shows how well the model created was able to generalize the validation dataset. There was some fluctuation in the first 8 epochs however, trended upwards reaching about 65% accuracy by the last epoch. Overall, while the results are satisfactory, it can be seen the hyperparameters were highly tuned and if the code could have run for all 50 epochs the results would most likely have trended towards more accurate results. However, the current model is still acceptable and was used to classify the test images as shown below.

In the second plot showing Training and Validation loss, the blue Training loss line Shows the reduction in training loss over epochs, starting high around 1.4 and steadily decreasing to below 0.5. The Red Validation loss line Starts around 1.4 and steadily decreases, with occasional fluctuations, reaching below 0.6 by the 14th epoch. The Training loss represents the error the model makes on the training data while Validation loss represents the error the model makes on the validation data. The decreasing training loss indicates that the model is learning and improving its predictions on the training data. It also indicates there is no overfitting, which would have happened if the training loss decreases, but the validation loss starts increasing (which means the model is memorizing the training data rather than generalizing). Since, both losses decrease together, no significant overfitting occurred.

Overall, the gap between training and validation accuracy/loss is relatively small, for both plots, indicating that the model generalizes reasonably well to unseen data and that there is no major overfitting from the training data. To see further improved results, better hardware is

required or using software like Google Colab to conduct Machine learning is required as the hardware used for this project was unable to let the program run for all 50 epochs based on the initial parameters set (128 filters, 5x5 Kernels, etc.).

```
Epoch 10/50
61/61 _____ 6098s 100s/step - accuracy: 0.6755 - loss:
0.7076 - val_accuracy: 0.6009 - val_loss: 0.7765
Epoch 11/50
61/61 _____ 5807s 95s/step - accuracy: 0.6917 - loss:
0.6978 - val_accuracy: 0.5870 - val_loss: 0.7754
Epoch 12/50
61/61 _____ 5776s 95s/step - accuracy: 0.6691 - loss:
0.6938 - val_accuracy: 0.6288 - val_loss: 0.7293
Epoch 13/50
61/61 _____ 5771s 95s/step - accuracy: 0.7063 - loss:
0.6664 - val_accuracy: 0.6265 - val_loss: 0.7147
Epoch 14/50
61/61 _____ 5681s 93s/step - accuracy: 0.7118 - loss:
0.6575 - val_accuracy: 0.6775 - val_loss: 0.7163
Model: "sequential_1"
```

Layer #	Layer (type)	Output Shape	Param
---------	--------------	--------------	-------

Figure 2. Code Stopping at 14 out of the 50 epochs set

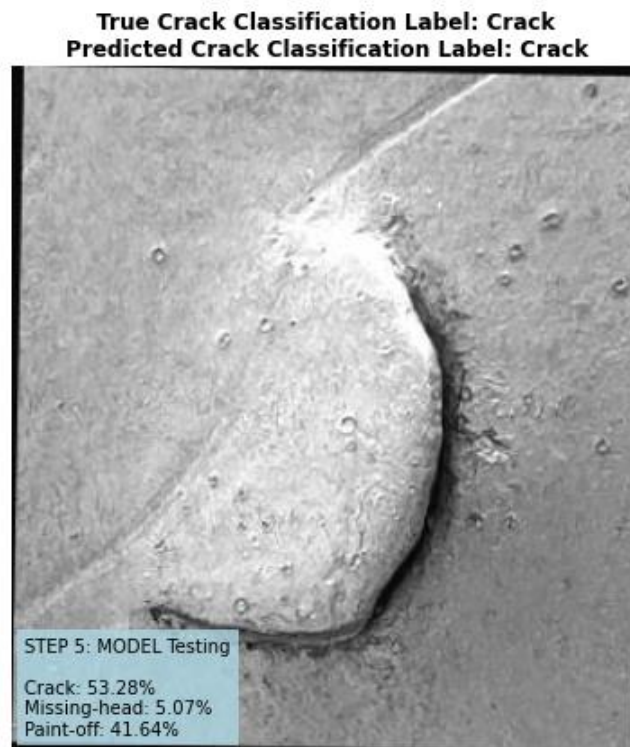


Figure 3. Crack Image Classification Test Result

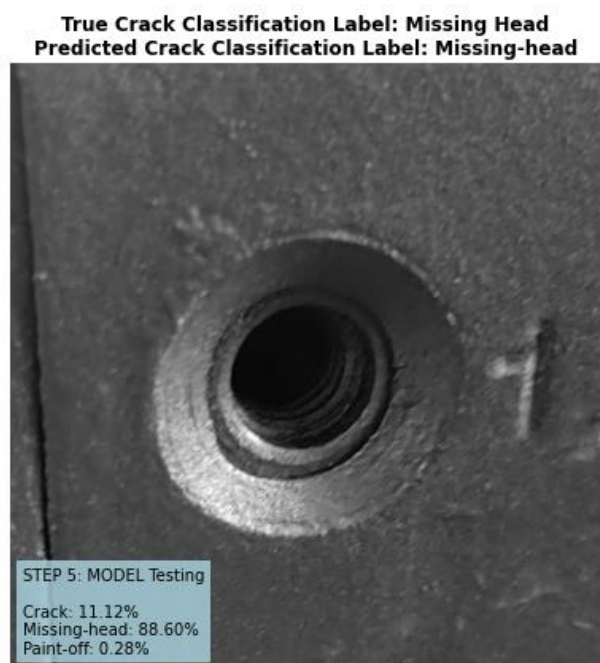


Figure 4. Missing Screw Head Classification Test Result

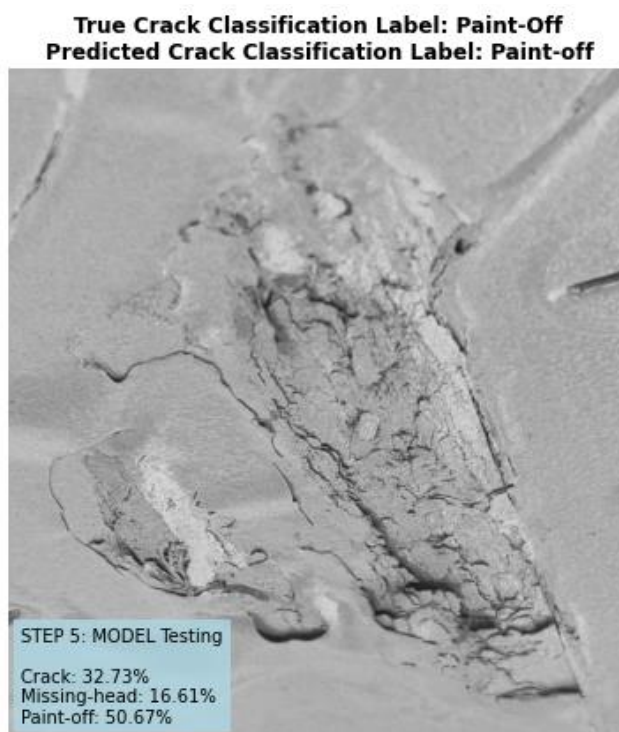


Figure 5. Paint-off Classification Test Result

From Figures 3-5 above, despite the incomplete model training, the CNN model was still able to correctly classify the 3 images. The predictive results are provided as well to show that the Crack image in figure 3, is the only image it had some issues with identifying the image. There was a 41.6% probability leaning towards Paint-off, but it was still able to marginally correctly pick crack with a 53% prediction. While improvements can be made to make the predictive percentages larger, a full 50 epoch run would have created the desired higher predictive percentage results.

4. Conclusion

In summary, the project was successful in creating a Deep Convolutional Neural Network to classify aircraft surface defects, such as cracks, missing screw heads, and paint degradation. Despite hardware limitations that constrained the model training to 14 epochs instead of the planned 50, the CNN achieved satisfactory performance. The training accuracy improved from 35% to over 70%, while validation accuracy reached approximately 65%, indicating reasonable generalization. Both training and validation losses decreased steadily without significant overfitting, reflecting the model's ability to learn effectively from the training data. The model's performance on test images further validated its performance capabilities, as it correctly classified all three test images. However, some predictions, such as the crack classification, showed marginal confidence levels of only 53%, which highlighted areas for potential improvement. Running the model for the full 50 epochs and utilizing more powerful hardware or cloud-based resources, like Google Colab, could significantly enhance these results.