Toronto Metropolitan University
Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Number	AER 850
Course Title	Intro to Machine Learning
Semester/Year	Fall 2025
Instructor	Dr. Faieghi Reza
TA Name	Hailey Patel

Lab/Tutorial Report No.	2

Section No.	3
Submission Date	Wednesday, Nov 5, 2025
Due Date	Wednesday, Nov 5, 2025

Student Name	Student ID	Signature*
Minsu Kim	XXXX17044	M.K

(Note: remove the first 4 digits from your student ID)
*By signing above you attest that you have contributed to this submission and confirm that all work you have 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, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/content/dam/senate/policies/pol60.pdf

Table of Contents

List of Figures

1.0 Introduction

The objective of this project was to develop and compare two variations of a Deep Convolutional Neural Network (DCNN) to classify aircraft skin defects into three categories: 'crack', 'missing-head', and 'paint-off'.

2.0 Data Processing

All images were resized to (500, 500). The training set utilized ImageDataGenerator with augmentations (shear_range=0.2, zoom_range=0.2, horizontal_flip=True) to improve generalization. The validation set used only rescale=1./255 to provide a realistic evaluation, matching the test data protocol. Due to VRAM limitations(6GB) of the RTX 3060 Laptop GPU, a BATCH_SIZE of 8 was used instead of 32."

3.0 Neural Network Architecture Design

To address the multi-class classification problem of identifying aircraft defects, two variations of Deep Convolutional Neural Networks (DCNN) were designed: a baseline model (Model A) and a deeper complex variation (Model B).

```
--- Model A Architecture ---
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 498, 498, 32)      896

 max_pooling2d (MaxPooling2D  (None, 249, 249, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 247, 247, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 123, 123, 64)     0
 2D)

 flatten (Flatten)           (None, 968256)            0

 dense (Dense)               (None, 64)                61968448

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 61,988,035
Trainable params: 61,988,035
Non-trainable params: 0
_________________________________________________________________
```

Figure 1 : Model A Architecture

The figure above shows the architecture of Model A. It was designed as a relatively shallow network to serve as a baseline. It consists of two convolutional blocks followed by one fully connected hidden layer.

```
--- Model B Architecture ---
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 498, 498, 32)      896

 max_pooling2d_2 (MaxPooling  (None, 249, 249, 32)     0
 2D)

 conv2d_3 (Conv2D)           (None, 247, 247, 64)      18496

 max_pooling2d_3 (MaxPooling  (None, 123, 123, 64)     0
 2D)

 conv2d_4 (Conv2D)           (None, 121, 121, 128)     73856

 max_pooling2d_4 (MaxPooling  (None, 60, 60, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 460800)            0

 dense_2 (Dense)             (None, 128)               58982528

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 3)                 387

=================================================================
Total params: 59,076,163
Trainable params: 59,076,163
Non-trainable params: 0
_________________________________________________________________
```

Figure 2 : Model B Architecture

The figure above shows the architecture of Model B. It was designed to test the hypothesis that a deeper network could extract more complex features and achieve higher performance. It includes an additional convolutional block (with 128 filters) and a larger fully connected hidden layer (128 neurons).

4.0 Hyperparameter Analysis

- Activation Functions:
  - ReLU (Rectified Linear Unit): Applied to all hidden convolutional and dense layers. ReLU was chosen for its computational efficiency and its ability to mitigate the vanishing gradient problem during training of deep networks.
  - Softmax: Used in the final output layer. It is the standard activation function for multi-class classification problems as it converts the raw output scores into a probability distribution across the 3 classes.

- Optimizer: The Adam optimizer was selected. Adam is widely used due to its adaptive learning rate properties, which often lead to faster convergence compared to standard stochastic gradient descent.

- Loss Function: Categorical Crossentropy was chosen as the loss function. This is the appropriate objective function for multi-class classification tasks where targets are one-hot encoded.

- Filter and Neuron Sizes: The number of filters in the convolutional layers was designed to increase progressively (from 32 to 64, and to 128 in Model B). This allows the network to detect simple features in early layers and combine them into more complex, high-level features in deeper layers.

5.0 Results

<u>5.1 Model A</u>

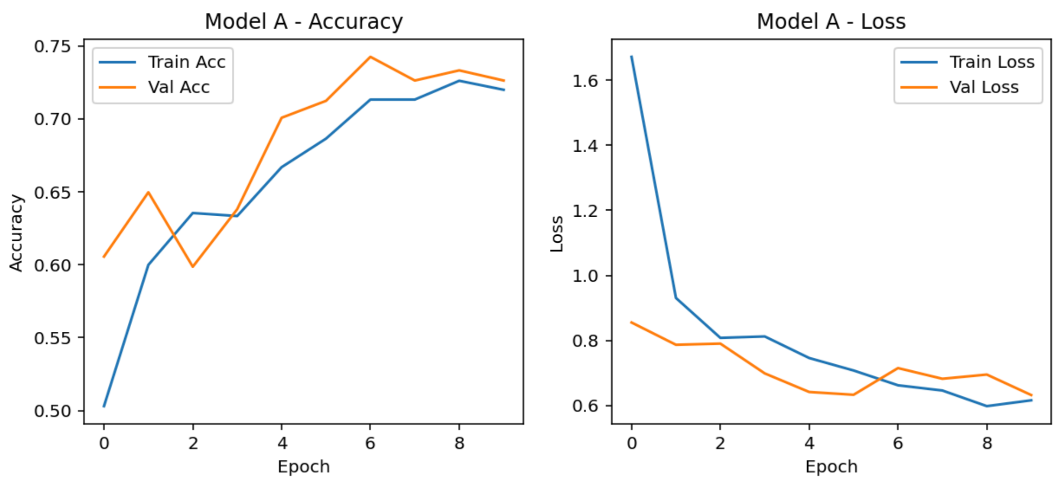

Figure 3 : Model A Performance

Model A achieved a peak validation accuracy of 74.28% at Epoch 6. The EarlyStopping callback correctly stopped the training at Epoch 9 as performance on the validation set began to decline, successfully preventing overfitting.

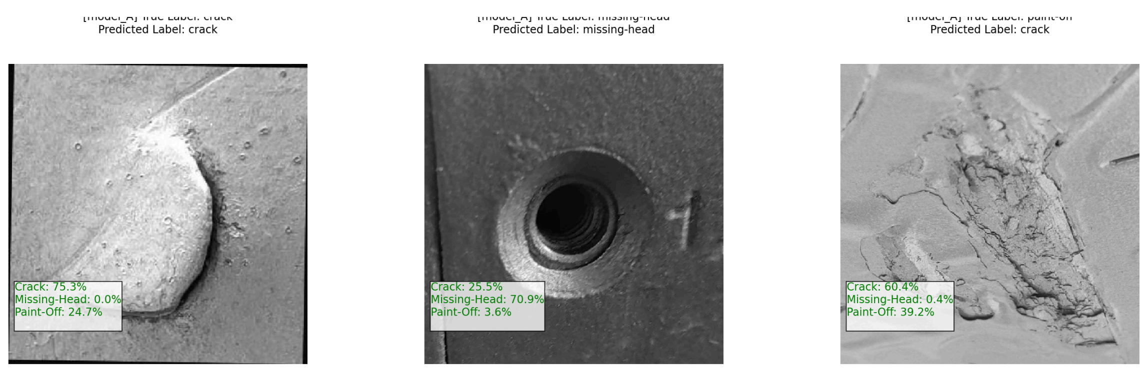

Figure 4 : Model A Testing Predictions

5.2 Model B

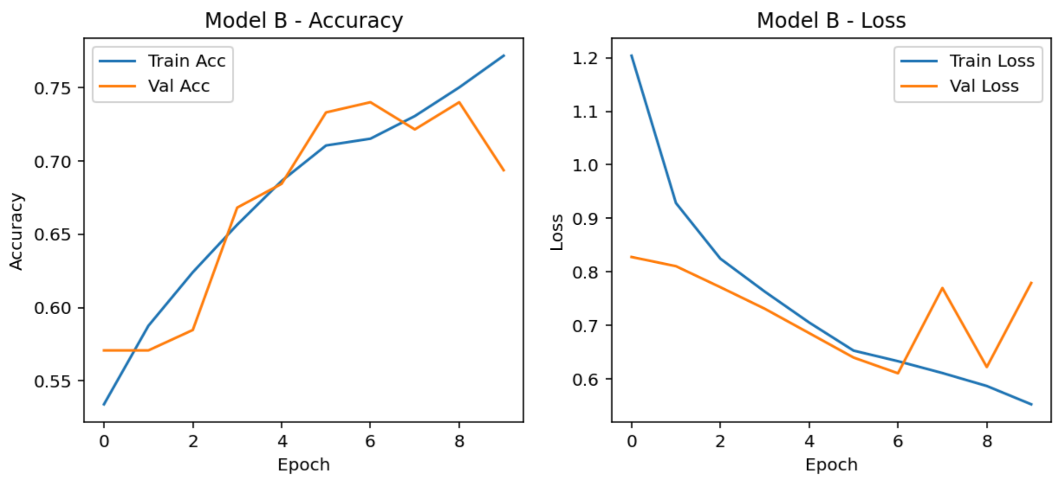

Figure 5 : Model B Performance

Model B achieved a lower peak validation accuracy of 74.01% at Epoch 6. The model stopped early at Epoch 9.

Figure 6 : Model B Testing Predictions

5.3 Discussion

Both models were evaluated on the three unseen test images. The results below show that both models correctly classified all images, but as it can be seen at figure 4 and 6, Model B generally showed higher prediction confidence.

- Cautious Prediction: Interestingly, for the 'crack' image, Model A had higher confidence (75.3%) where Model B was 66.5%. Model B assigned a notable probability (33.1%) to the 'paint-off' class for this image, suggesting it might be more sensitive to ambiguous features that could belong to either class, leading to a more cautious but still correct prediction.

- Higher Confidence in Model B: For the 'missing-head' class, Model B showed significantly higher confidence (88.8%) compared to Model A (70.9%), indicating a more robust feature representation for this distinct defect type.

- Superior Classification by Model B: The most critical finding is that Model A misclassified the 'paint-off' test image as a 'crack' with 60.4% confidence. In contrast, the deeper Model B correctly identified it as 'paint-off', 59.0%. This shows that Model B's deeper architecture enabled it to learn more discriminative features to distinguish between similar-looking defects.

6.0 Conclusion

This project successfully developed and compared two DCNN variations for aircraft defect classification. The baseline Model A achieved a slightly higher validation accuracy (74.28%) than the deeper Model B (74.01%) during training. However, Model B proved to be the superior model in practical testing. Model B correctly classified all three test images, whereas Model A failed to identify the 'paint-off' defect. These results confirm the hypothesis that a deeper network architecture can learn more complex and robust features, leading to better generalization on unseen data, even if training metrics might suggest otherwise.

References

1. https://github.com/davidsukim/AER850-Project-2.git