ECE324 Final Report:
AER202: Aircraft Recognition

Daniel Ding – 1004015040
Frederick Boyd – 1004021435

Word Count: 1969
Penalty: 0%

December 3, 2019

1 Introduction

Plane spotting is a hobby involving the tracking, photography and identification of aircraft movements. Aviation enthusiasts are often found at airport perimeters, hoping to record unique airliner sightings in their logbooks. Unfortunately, it is difficult to spot the differences between various aircraft, especially when they are moving at speed or traveling at a distance. Spotters have traditionally relied on costly cameras and bulky recognition manuals; these factors make it difficult for newcomers to participate.

To make plane spotting more accessible, we present a neural network powered aircraft recognition tool. With increasingly advanced smartphone photography, it's now possible to capture detailed images right from the pocket. We envision plane spotters feeding images to our model to quickly identify features like aircraft type and airline.

Machine learning is critical, as neural nets may be able to identify subtle aircraft characteristics that humans struggle with. The difference between models can be very minor, such as the curve of a wing or shape of the engines. A powerful neural net can greatly increase identification accuracy and speed. Equally important is adaptability: With proper training, a neural net can make accurate predictions despite different situations such as varying aircraft orientation. This is perfect for plane spotting, since the hobby takes place entirely outdoors with little control over environmental conditions. Finally, extensive aircraft photo databases can be found online with labelling, easing the data collection process.

2 Illustration/Figure

The diagram below is a high-level overview of our project:

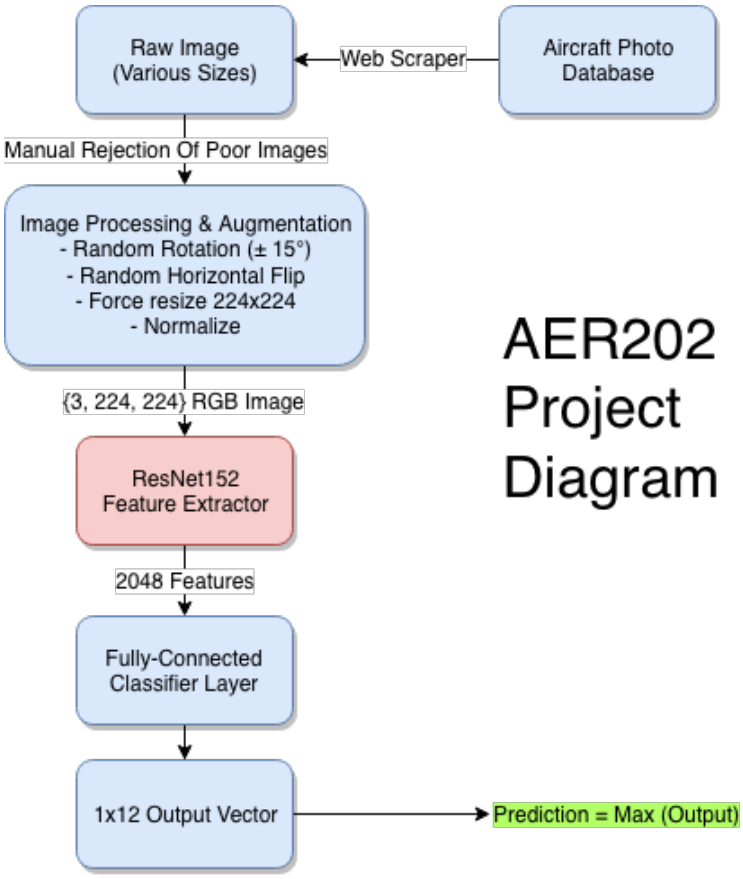


Figure 1: High-level overview of project structure.

3 Background & Related Work

An example of a similar project that has been done using neural networks is car classification. The description and source code for this project has been published on GitHub [1]. The project's architecture can be found in Figure 2. At a high level, there are two main aspects of this architecture that stand out:

- Transfer learning with multiple pre-trained models (ResNet50, Inception, Xception)
- Long short-term memory (LSTM), a type of RNN architecture

Transfer learning is used for feature extraction. There are two fully connected layers after the pre-trained models whose weights are updated during training in order to make decisions on which features are relevant to each car.

After the fully connected layers, there are multiple LSTM units. The goal behind this RNN architecture was to provide the neural network with more context when evaluating the photo. The author noted that the features they look at when identifying a car is similar to a sequence. For example, they might first look at the shape of the doors, then the headlights, and finally the radiator grill [2]. They noted that implementing LSTM units increased their accuracy from 55% to 72% compared to just using transfer learning.

Another interesting technique employed was the use of ImageAI. A YOLO object identification model was used to identify the car's location so that part of the background could be removed. An example of this being applied to an airplane can be found in Figure 3. The goal behind this was to remove unnecessary information before presenting it to the neural network [2]. This increased the author's accuracy by an additional 10% [2].

4 Data Processing

A dataset has been built by scraping Airliners.net, an aviation photo database (Figure 4). Airliners.net has been a mainstay of the plane spotting community since the late 90s. The website contains a database of over 3 million labelled aircraft photos. There are photographs of nearly every aircraft type ever built and every airline. Importantly, all Airliners.net photos have undergone manual review prior to posting. The strict standards mean that images are more sterile; there is minimal editing and creative flair, which is good for constructing a reasonable training set.

A web scraper was built using Python. We were able to identify HTML elements of the website that linked to subsequent images, along with the elements that contained the actual jpg file. By using BeautifulSoup, an HTML parser library, we were able to extract these elements from the rest of the website code. This process of extracting and downloading images was looped. By adjusting the range of the loop, we controlled the number of images downloaded. The `scraper.py` file can be seen on our GitHub.

The final dataset contains 12 classes (aircraft types). There are 300 images per aircraft type for a total of 3600 images. Where needed, we supplemented our data via augmentation (rotation, cropping, flipping). To reach 300, approximately 600 raw images were downloaded per aircraft from Airliners.net. These were then manually assessed: They had to be good side profiles, in good lighting, with the full aircraft body visible. Poor images were rejected. Images were scaled down to a resolution of 500×500 for efficiency. The data was split between 64% training, 20% validation and 16% test. Some examples of good/poor images are shown in figure 5. Some class samples are shown in figure 6.

Our approach to data collection serves more as a proof-of-concept, so there are limitations. Since our images are mostly side profiles or slightly angled, the network has not been trained to identify aircraft/airlines from extreme orientations. In addition, there is currently no ability for identification during night or adverse weather. As a future goal, it would be interesting to collect more diverse data and expand capabilities.

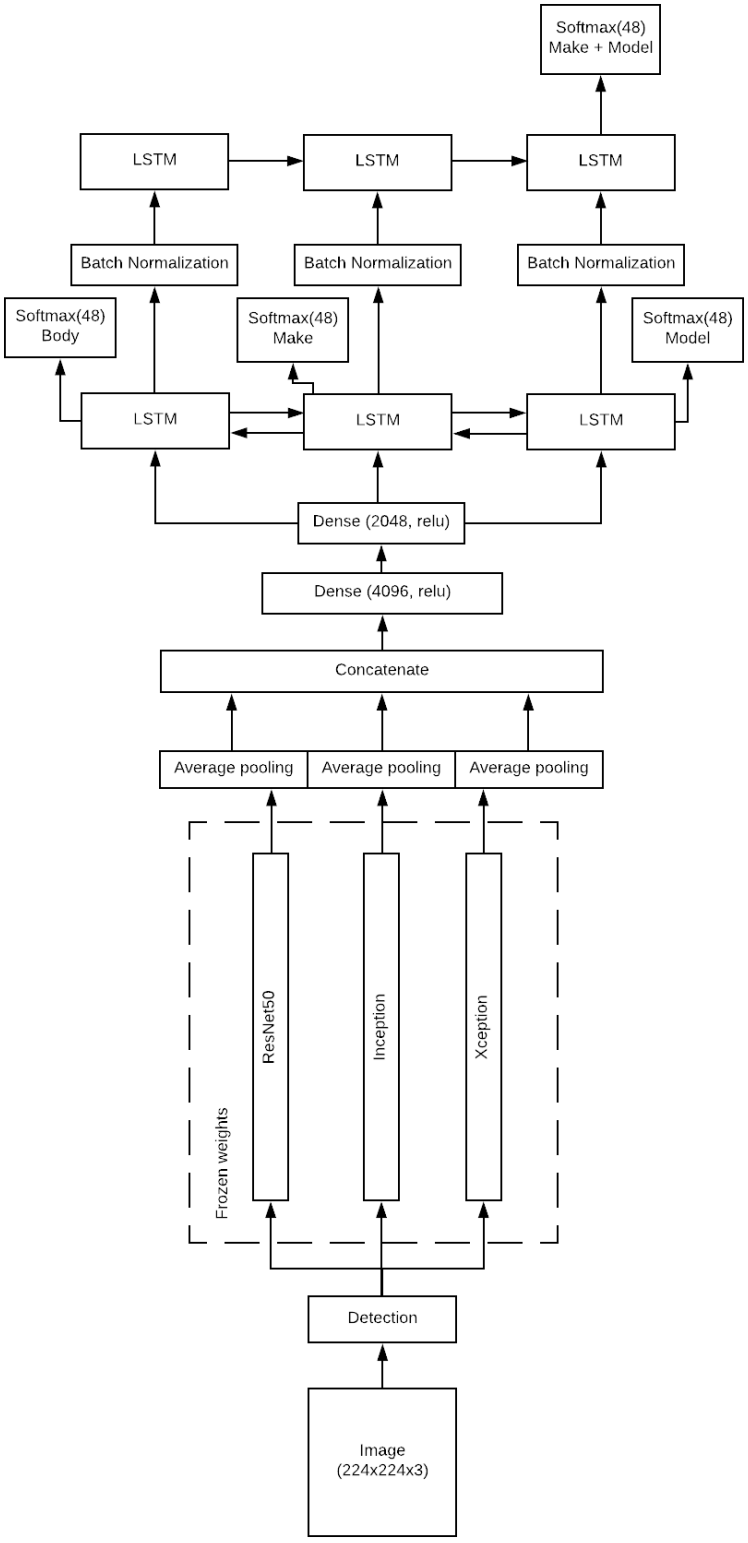


Figure 2: Example of a neural network architecture that has been used to classify cars by their model and manufacturer [1].

Figure 3: Example of a YOLO object identification model being used to intelligently crop the background out of photos as a data pre-processing step. The purpose of this is to remove unnecessary information before feeding the image into the neural network.

Figure 4: A typical Airliners.net image page, illustrating labels (Airline, Manufacturer, Model)

Figure 5: Examples of acceptable v.s. poor images.

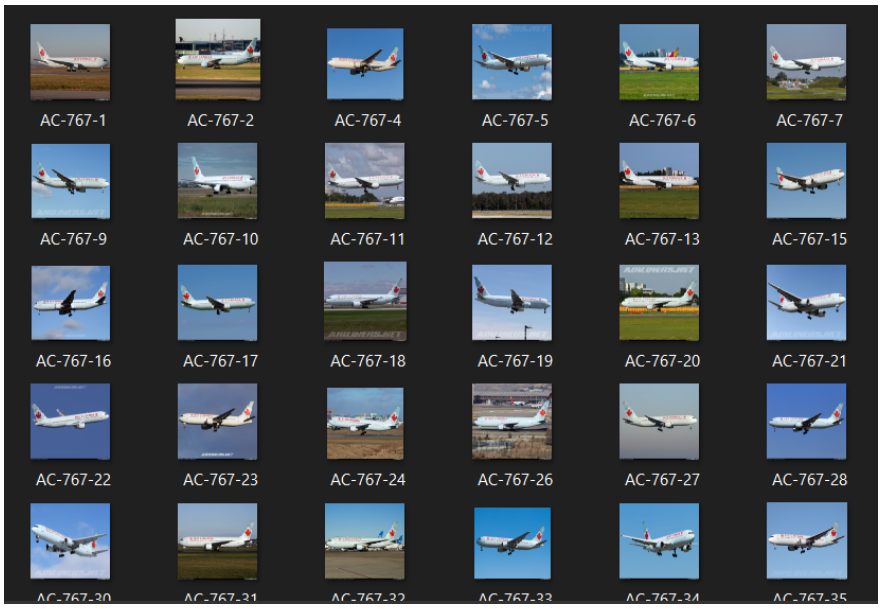


Figure 6: Some image samples from the final Air Canada 767 class.

5 Architecture

We are leveraging transfer learning with a pre-trained CNN. We decided to use transfer learning since we have limited time and data; we wanted the network to generalize as quickly as possible. We chose ResNet152 for its depth (allowing for better feature representation), speed of training and extensive documentation. ResNet features residual connections that reintroduce outputs from previous layers, compensating for the vanishing gradient problem. We have implemented ResNet152 along with several augmentation techniques (rotating, cropping, flipping) to achieve around 99% validation accuracy. A screenshot of the `torchsummary` printout can be seen in figure 7. The network is 152 layers deep, 800+ megabytes in size, and has 50 million+ parameters. The network is run via Google Colab and takes advantage of GPU acceleration.

For training, the weights of the network are frozen except for a final fully connected layer (classifier). This is the only layer that is trained. The earlier convolutional layers therefore serve as feature extractors. We use cross entropy loss and the SGD optimizer. We also use the `ReduceLROnPlateau` scheduler, decreasing learning rate dynamically when validation accuracy stagnates. This is based on a 'patience' value - the amount of time we are willing to tolerate for stagnation. The model with the best accuracy is continuously saved during training, so that it can be loaded afterwards.

Final Hyperparameters:

- Batch Size: 64
- Loss Function: CrossEntropyLoss
- Optimizer: SGD
- Learning Rate: 0.01 (Reduced on Plateau, Patience = 3, Threshold = 0.9)
- Momentum: 0.9
- Epochs: 25
- Seed: 1

```
Total params: 58,168,396
Trainable params: 58,168,396
Non-trainable params: 0
----------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 606.58
Params size (MB): 221.89
Estimated Total Size (MB): 829.05
----------------------------------------
```

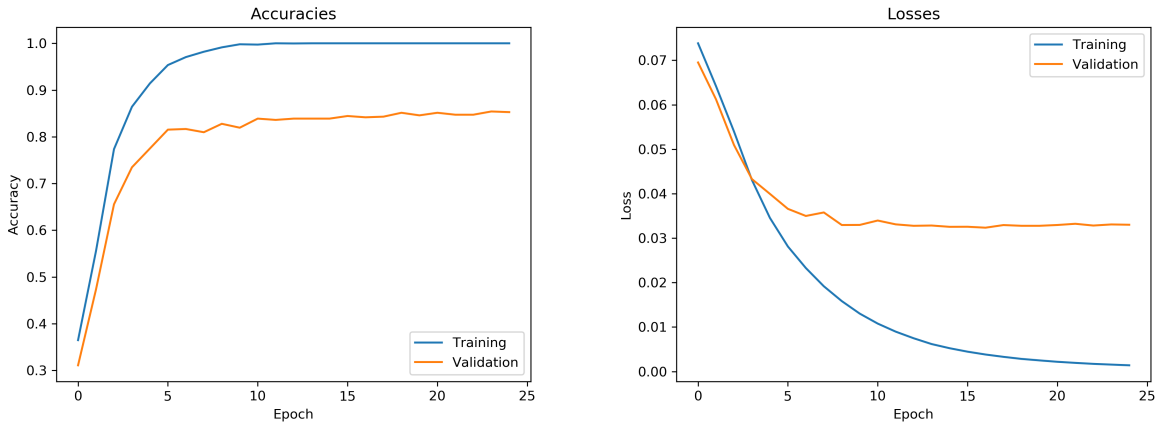
Figure 7: Model summary screenshot

6 Baseline Model

An overview of the baseline model used can be found in Figure 9. There are two convolution layers, each followed by a max pool and ReLU activation function. Two fully connected layers provide the final output. Plots of accuracies and losses can be found in Figures 8a and 8b.

- Convolution Kernel Size: 3×3
- Convolution Stride: 1

- Max Pool Kernel Size: 2×2
- Max Pool Stride: 2



(a) Baseline model accuracies.

(b) Baseline model losses.

Figure 8: Baseline model results.

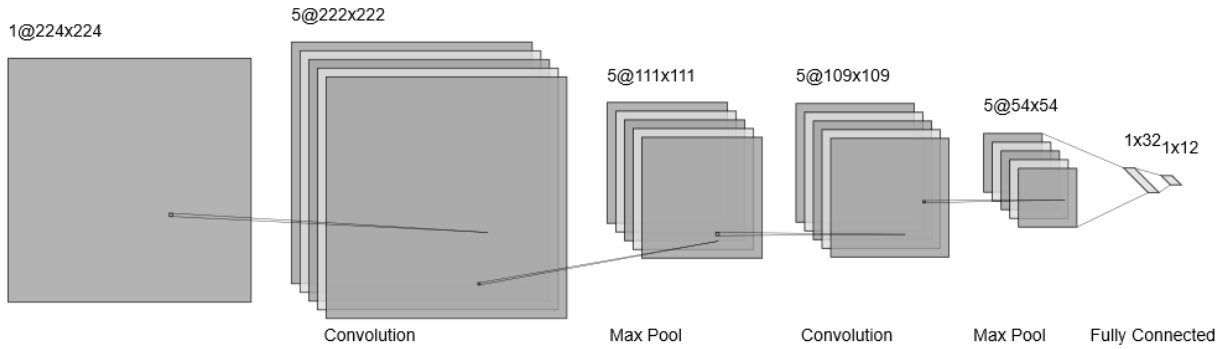


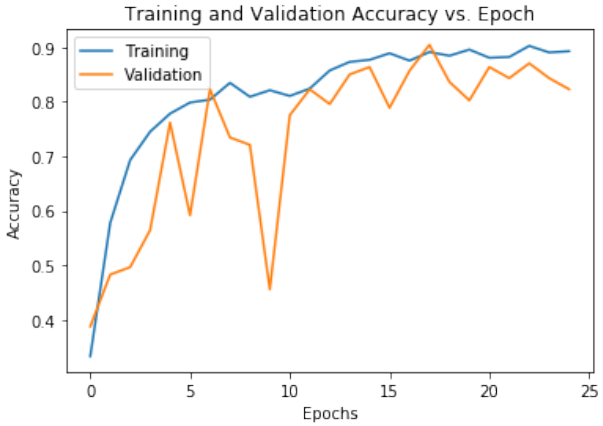
Figure 9: Overview of the CNN baseline model used as a reference. The model is similar to the one used in Assignment 4.

7 Quantitative Results

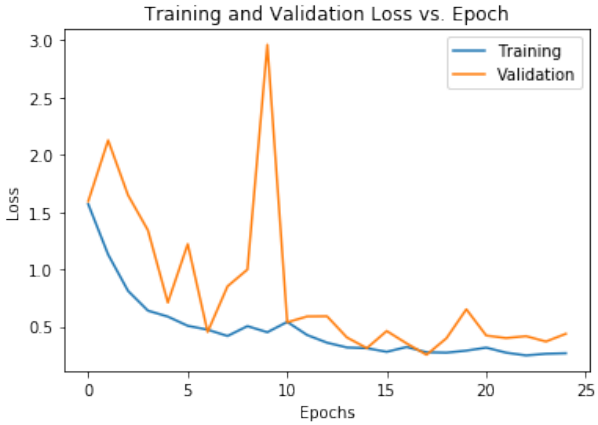
Our initial ResNet tests yielded validation accuracies of approximately 70-80%. Training was unstable (figure 10a, figure 10b), took nearly two hours on Colab and performed worse than expected. We improved performance by implementing:

- Adaptive Learning Rate (reduced on plateau)
- Reduced augmentation aggressiveness (dramatic transformations were unrealistic and didn't reflect real aircraft orientations)
- Forced resizing instead of cropping to preserve full view of aircraft and resolution
- Increased batch size to 64 (limited by VRAM)

Following this, we achieved good (99%+) results across training, validation and test sets. See figures 11a, 11b and 12. Training generally took around 30 minutes. A confusion matrix of our test set (576 images) illustrating strong performance can be seen in figure 13. We discuss some of these results in the qualitative section.

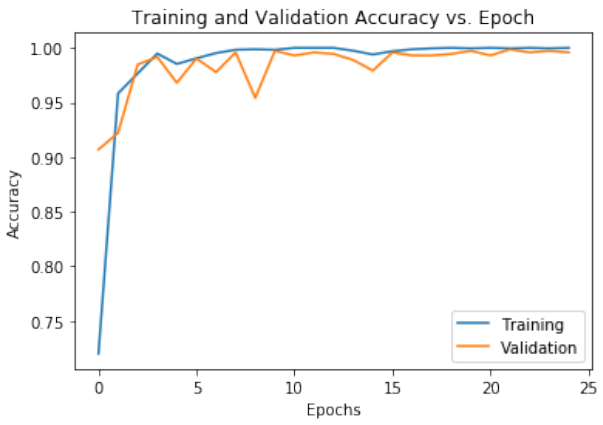


(a) Initial accuracy during early trials

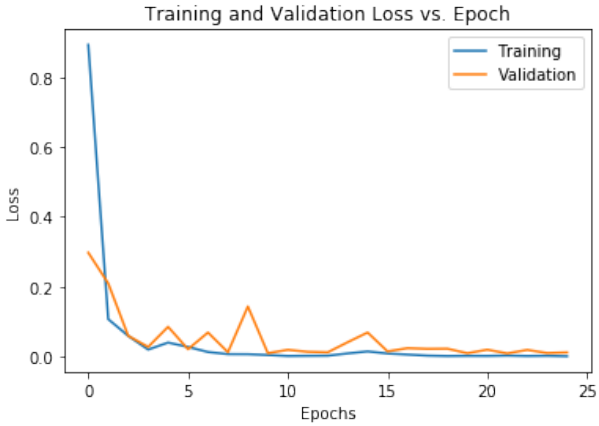


(b) Initial loss during early trials

Figure 10: Initial results



(a) Final accuracy



(b) Final loss

Figure 11: Final results

```
Epoch 25/25
----------
train Loss: 0.0005 Acc: 1.0000
valid Loss: 0.0115 Acc: 0.9958
test Loss: 0.0050 Acc: 0.9983

Training complete in 26m 48s
Best val Acc: 0.998611
```

Figure 12: Final model statistics

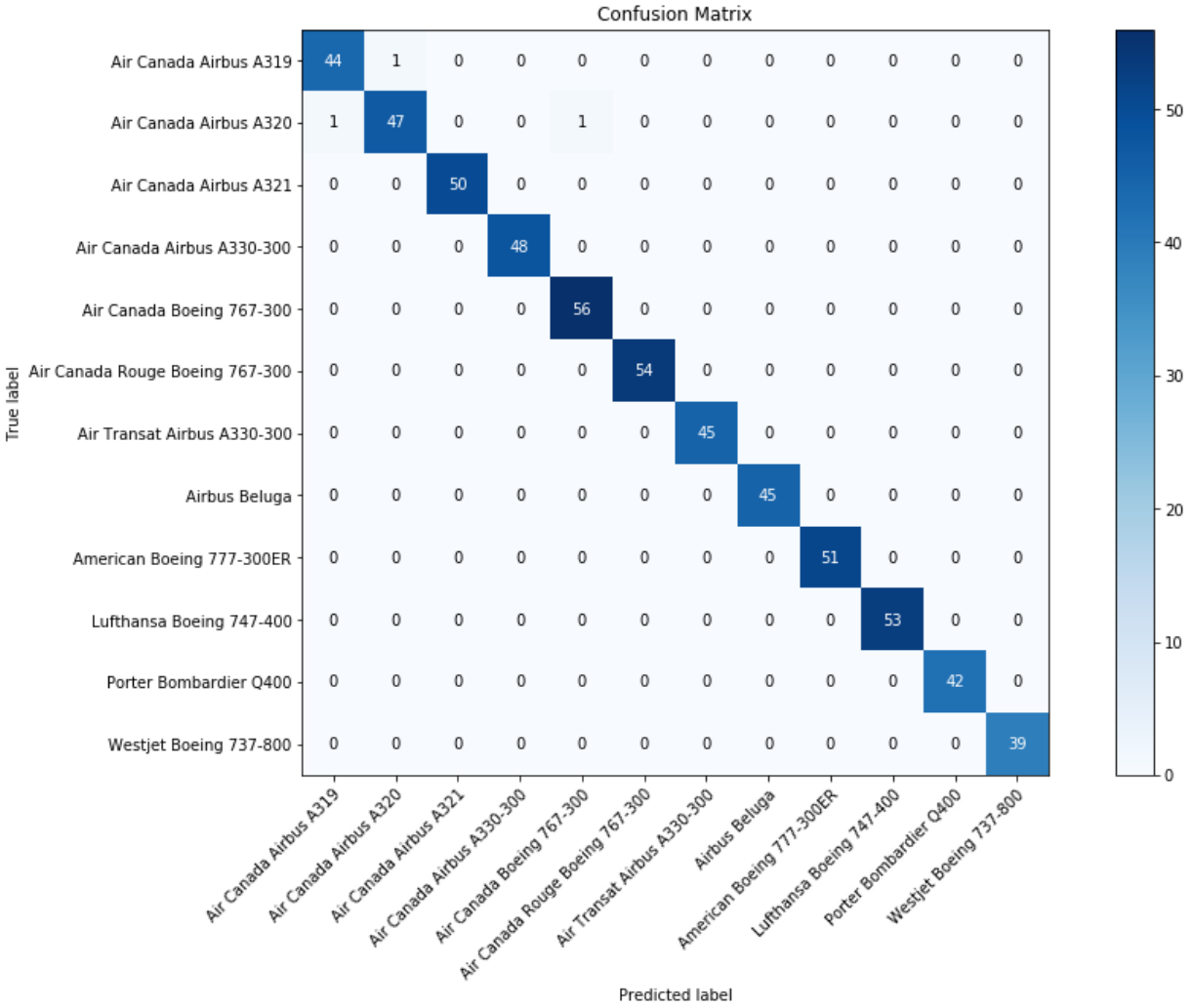


Figure 13: Test set confusion matrix

8 Qualitative Results

A random sampling of test set predictions and labels can be seen in figure 14. The model was able to achieve nearly perfect accuracy under different aircraft orientations, lighting conditions and backgrounds. Since we achieved good performance with our original dataset, we decided to test more challenging cases. Figure 15 illustrates some interesting results, using non-dataset images. The top row on the left shows extreme angles and even watermarks, but the model still performs well. The bottom row features aircraft that the model was never trained on. Interestingly, the Boeing Dreamlifter is predicted as an Airbus Beluga, which is a highly similar cargo aircraft in terms of dimensions and function. Likewise, the Ryanair 737 is predicted as a Westjet 737; despite not being trained on the airline, the network is still able to identify the type. Clearly there is a high degree of generalization in the model. The Norwegian 737 is misidentified as an Air Canada Rouge 767; this is likely due to similarity between the strong red colouring. On the right, the most common mistake in the confusion matrix is shown; this was to be expected, as these two aircraft are so similar that even experienced spotters have difficulty.

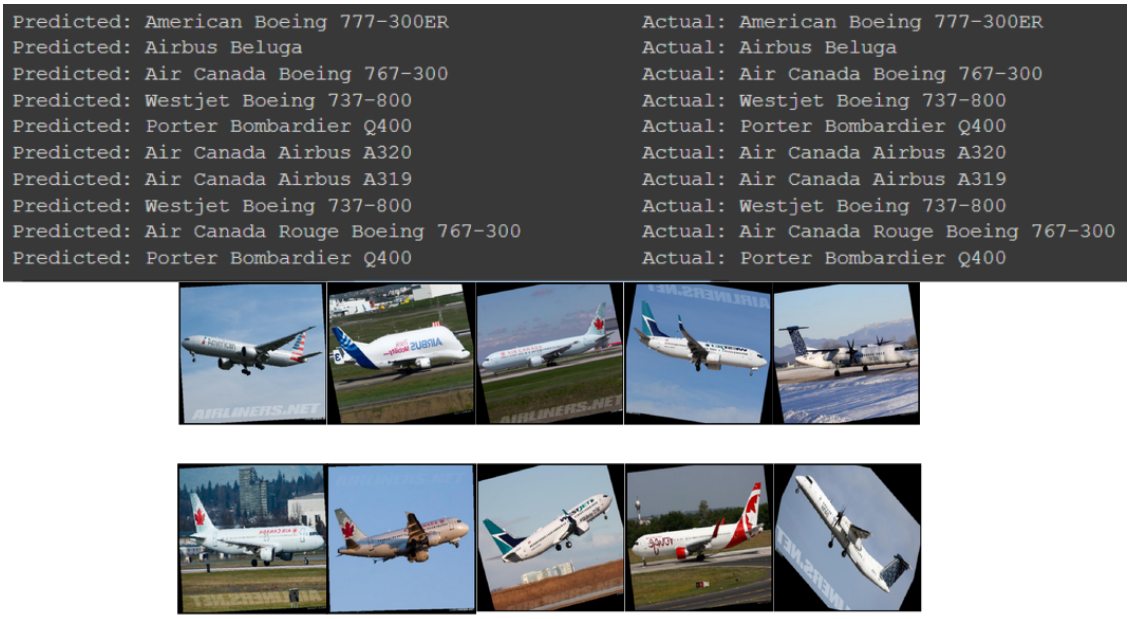


Figure 14: Highly accurate test set predictions vs. labels

Figure 15: L: Noteworthy results from non-standard, non-dataset images. R: Confusion matrix error

9 Discussions & Learnings

The key takeaway from this project was the strength of transfer learning. We obtained results that were far better than what we expected, in a far shorter time (compared to training models from scratch). Pre-trained networks are often demoed on basic classification problems (e.g. apple vs orange, house vs car), so we did not expect them to perform so well in discerning highly similar machines. We were also surprised at how the model adapted to non-dataset images (figure 15). If we had known this level of performance was possible prior to finalizing our dataset, we may have been more lenient with the data cleaning and/or included more diverse/challenging images.

We found data collection to be the most challenging aspect, as opposed to building the actual model. Image downloading was bottlenecked by internet speed. Cleaning and processing data took significant effort and time. Storing and moving the data was also difficult; we had issues with huge file sizes and services like Google Drive deleting images. We came to appreciate the power of cloud computing services such as Google Colab; training and debugging would have been a long and arduous process if we used the integrated GPUs on our laptops.

We also had another data pre-processing method ready to implement in the event that our accuracies were not ideal. We had a working script that used YOLO object recognition to identify the location of an aircraft in an image and intelligently crop the photo so that just the aircraft remained. However, given the results we had when just using transfer learning, we decided the additional potential accuracy was not worth the effort and performance impact of integrating the script.

With more time, we would have liked to explore additional classes to see how much complexity our model can handle. Finding ways to further automate data processing (particularly rejection of poor images) would be highly beneficial. We also wanted to explore images from night time or poor weather, and expand to a prototype application or website. However, given this project's time constraints and the significant manual effort required to curate photos, we were unable to do this.

10 Ethical Framework

We see beneficence and justice being relevant to the proposed application. This largely affects aviation enthusiasts, who benefit from faster and easier access to aircraft identification. The hobby as a whole is more accessible (and just), since those without high-end equipment or prior knowledge can now participate. Data collection and privacy concerns are not highly relevant, since aircraft data has been available online for decades and does not contain personal information. Non-maleficence could be an issue, particularly when it comes to governmental and military organizations. The US Air Force is actively investigating AI aircraft recognition [3]. Even with our rudimentary testing we have seen promising results; professionals could greatly extend these capabilities. There could be applications in weapon targeting systems, which raise heavy concerns about accuracy. If a network were to incorrectly discern between friend vs. foe, or military vs. civilian, there could be severe implications.

References

[1] I. Bohaslauchyk. `car_identification`. [Online]. Available: https://github.com/iPhaeton/car_identification

[2] ——. Neural network for car recognition. [Online]. Available: https://dashbouquet.com/blog/artificial-intelligence/neuro-network-for-car-recognition

[3] R. Mash, N. Becherer, B. Woolley, and J. Pecarina, "Toward aircraft recognition with convolutional neural networks," 07 2016.

Permissions

1. Permission to post video: Yes
2. Permission to post final report: Yes
3. Permission to post source code: Yes

Fun fact: The word count is a reference to the year of first flight for the Boeing 747.