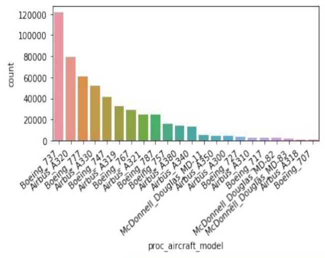
Convolutional Neural Networks for Aircraft Model Identification

Problem Statement + Strategy

- Using a **Convolutional Neural Network** to identify the make and model of an airplane from a picture of it
- Airplanes can be distinguished by the presence of very geometric features (winglets, size and distance between windows, wingspan, etc.) - these are enlarged and translated (due to different positions of the photographer) but never deformed since airplanes don't bend -- this is, theoretically a great real-world distribution to exercise a CNN on.

- We first validated the model on 30GB of data, yielding 85%/73% accuracy on train/val sets respectively.
- We determined that overfitting was controlled (and could be reduced even more by increasing dropout), so we had a bias problem. Given this, we scraped 20x more data, and re-trained with half of that.
- The best architecture trained on the bigger dataset yielded **82% test accuracy**
- **Note: test set evaluations were only run at the end, after all models evaluated. Factor refers to what value the resolution is divided by.**



Dataset information

- **700GB** of well-labeled images scraped off plane-spotting websites
  - For speed/accuracy tradeoff, used only 300GB
  - # of images: train **485k**, val 27k, test 27k (95% / 5% / 5%)
- Images vary in resolution and size, and all airplanes are "widescreen"
  - Images resized by different factors of the mean resolution in the dataset, which is (812 x 1200) divided by factor. Training takes approx. 4 hours per epoch (1 GPU)
- Significant infrastructure challenges with dataset of this size - scraping, preprocessing, and modeling required significant work.
- Note that classes are not equally represented, as shown to the right

Results

Model	Train acc.	Val acc.	Test acc.	# epochs
Guessing most common class	n/a	n/a	24%	n/a
Human expert (estimated)	n/a	n/a	88%	n/a
Bayes' error (estimated)			82%	
Arch. B (30GB)	85%	73%	n/a	15
Arch. B (300GB, factor=4)	***84%***	***82%***	***82%***	***4***
Arch. B (300GB, factor=6)	79%	80%	79%	5
Arch. B (300GB, factor=4, padding=same)	81%	80%	80%	5
Arch. B (300GB, factor=6, padding=same)	80%	80%	79%	5

Model Architecture

```
Conv2D(32, kernel_size=(3, 3), activation='relu',
             input_shape=img_dims2, padding='same'))
Conv2D(64, (3, 3), activation='relu', padding='same'))
Conv2D(64, (3, 3), activation='relu', padding='same'))
MaxPooling2D(pool_size=(2, 2)))

Dropout(0.5))
Conv2D(128, (3, 3), activation='relu', padding='same'))
Conv2D(128, (3, 3), activation='relu', padding='same'))
MaxPooling2D(pool_size=(2, 2)))

Dropout(0.5))
Flatten())
Dense(64, activation='relu'))
Dropout(0.5))
Dense(num_classes, activation='softmax'))
```

- Having re-trained with more data, our previous observation that we had a bias, not variance, problem is confirmed: changing nothing except more data,, validation accuracy increases 9 p.p. whereas train accuracy *decreases* 1 p.p.
- This decrease in training acc. is because of the large amount of data: the model only saw each image few times, making it less likely that it would overfit to the training set.
- In fact, our model achieves over 60% validation accuracy *after a single epoch*.
- Note that preserving the convolution volumes increases parameterization; these need additional epochs to reach the same performance.

Future work

- Three factors clearly help the CNN achieve greater accuracy:
  - More data - done, no use for more data
  - Greater image resolution - difficult to train with more than ¼ the original resolution due to memory problem
  - Preserving input volumes through convolution (through non-destructive padding)
- Overall, the biggest constraint was GPU memory. Even with parallelized training, we did not have access to >8GB per GPU which limited image size and model complexity.
- Future work: automatically detect and crop the airplanes in the image, so we can discard useless pixels and more of the image is informative
- Future work: investigate whether the model's performance could be increased using grayscale images so less memory is necessary

Error analysis

- To better understand our CNN's weaknesses and determine future work, we performed manual error analysis of about 30 images per class (on the model trained with 30GB data only)
- Model fails when *airplane is a small fraction of the image*, e.g. far away in the sky
- Model fails when input image is the cockpit of the airplane (already taken into account in Bayes' error)
- Model fails when several airplanes are in the image (common in airport photos)

Convolutional filter visualization

- We attempted to visualize the convolutional filters to better understand what information the CNN was relying on. This was done by using gradient ascent to find the input image that maximizes a given filter.
- Unfortunately, the filters are not particularly informative. We offer two hypotheses as to why:
  - The maximizing image is very dissimilar from the training distribution; the visualization may improve if an additional regularization is applied when creating it.
  - The varied positions of airplanes within the image result in overlapping, useful but uninterpretable templates.
- We also tried to find to perform *Class Activation Maximization*, but ran into difficulties due to the amount of data/time to train.

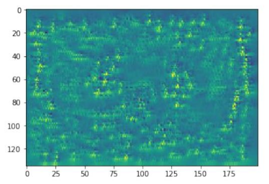


Fig. 1: Input image that maximizes one of the early convolutional filters (typical image - other filters look similar)

Guilherme Reis

CS230, Winter 2018