Alison Au
2/27/22

MSDS 422 - Module 8 Assignment
Dogs vs. Cats – Convolutional Neural Networks

Description of Criterion

The Dogs vs. Cats data set contains images of both animals with 25,000 in the training (labeled) and 12,500 in the test set (unlabeled). The goal of this assignment is to build a convolutional neural network (CNN) that correctly identifies each image as a dog or cat. The Kaggle competition scores the networks based on the log loss, which considers the probabilities predicted from the models. I configured the networks to be evaluated based on accuracy given its easier interpretability.

The images are varied with different dog and cat breeds, lighting, angles, sizes, positioning of the animals, and backgrounds. Each image is interpreted by the computer as a set of pixels, with each pixel represented by 3 digits ranging from 0 to 255 that correspond to RGB values. EDA shows the training set is evenly divided into dogs and cats (12,500 each).

Model Development

To prepare for model fitting, the data was split 80/20 into training and validation sets. Each set was also scaled using min/max scaling so that all values ranged between 0 to 1. Scaling is necessary for neural networks, otherwise there may be a lack of convergence or a bigger risk of landing on a local optimal. Images were resized to 130x130 pixels and read in batches of 32 at a time. To expose and train the model to different orientations, the following image augmentation was done: rotated images between -30 and 30 degrees, distorted (shear range 0.2), and zoomed (zoom range 0.2).

After some initial exploration, 3 models were tested. The first is a custom architecture of 3 convolutional layers, two with 32 filters of 3x3 then 64 filters of 3x3. All three use ReLU for activation. It is followed by 2 dense layers and a 0.5 dropout, one with 64 nodes and ReLU activation and the last outputting the prediction with a sigmoid function. Optimizing was done with RMSprop, binary cross entropy for loss, and binary accuracy as the optimization metric.

The other two models were based on VGG16. Both used global average pooling and a 0.2 dropout for regularization prior to its dense layer, which classified the images with a sigmoid function. Stochastic gradient descent was used for optimization, binary cross entropy for loss, and binary accuracy as the optimization metric. The first VGG16 model was configured with 20 epochs and early stopping based on the validation's binary accuracy. However, 20 seemed insufficient as accuracy and loss were still improving. The second VGG16 model thus extended this to 30 epochs, and early stopping was set based on validation loss to align with the Kaggle evaluation metric. Step size between was also specified in this second model.

Model Evaluation & Results

A summary of the models' performance is below.

Model	Layers	Parameters	Epochs	Time to Train	Train Accuracy	Test Loss
Custom	3 conv, 2 dense	831,585	20	0:40:22	0.85149	0.35201
VGG16 v1	13 conv, 1 dense	14,715,201	20	1:16:60	0.93161	0.16747
VGG16 v2	13 conv, 1 dense	14,715,201	30	3:18:24	0.94275	0.18493

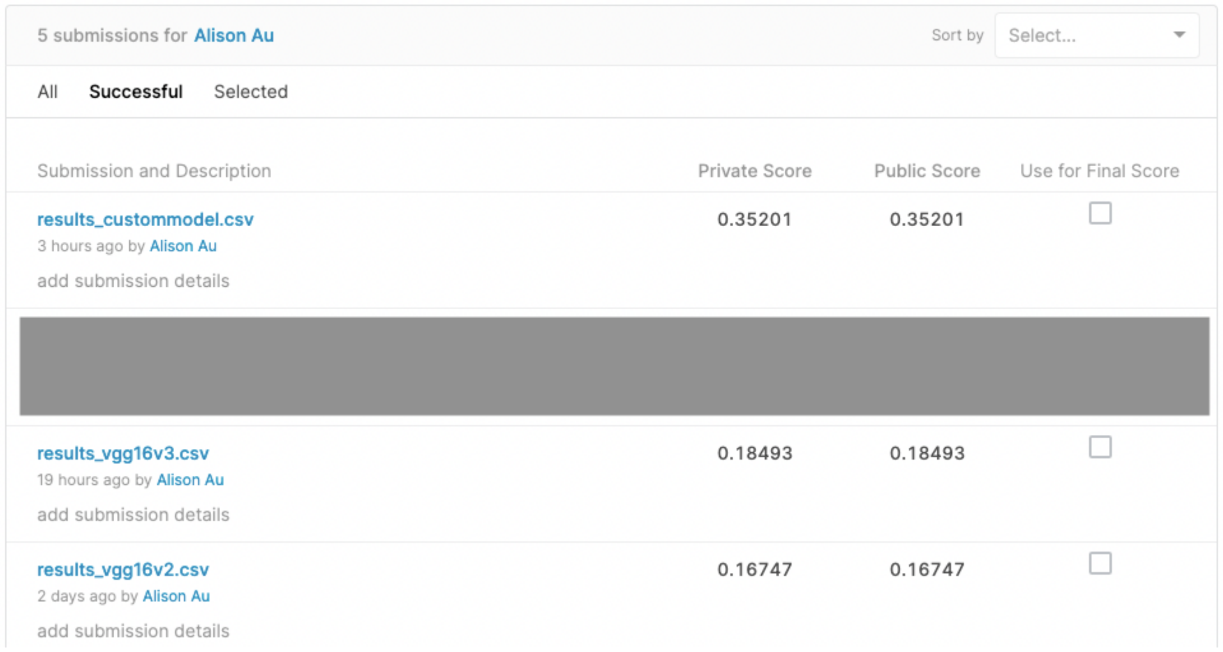
The VGG16 models performed significantly better than the custom model. This is not surprising given that they contain several more convolutional layers and have been pre-trained on the ImageNet dataset. They produced a training accuracy of about 0.93 and 0.17 log loss on the test. In contrast, the custom model had 0.85 accuracy and 0.35 loss. The VGG16 models did take more time to run, though the first one took only about double the time of the custom model even though it had about 3 times the number of layers (both ran for 20 epochs). Having pre-trained weights likely helped with this. Ultimately, the additional processing time is worth the improved performance.

In contrast to my expectations, the first VGG16 model performed slightly better than the second one. The VGG16 v2 model seemed to improve beyond 20 epochs. Indeed, accuracy for both train and validation increase a percentage point. However, log loss on the test degraded by 0.02 even with early stopping set based on the validation loss. This did fluctuate slightly with each epoch, so perhaps the CNN was overfitting. It also took more than double the time to train with just 10 more epochs, though specifying the step size also may have extended the time.

The best model VGG16 v1 had high 97% precision identifying dogs, though recall was weaker at 89% of dogs correctly classified. On the flip side, precision identifying cats was 90% while recall was 97%. The second VGG16 model had a somewhat opposite performance with higher recall (97%) than precision (92%) for classifying dogs. In other words, it classified more of the images as dogs so captured more of them but had more false positives.

Kaggle Results

Results submitted to Kaggle are shown below under username Alison Au. The best model was VGG16 v1 with 20 epochs, producing log loss of 0.16747.



5 submissions for Alison Au

Sort by Select...

All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
results_custommodel.csv 3 hours ago by Alison Au add submission details	0.35201	0.35201	☐
[redacted]			
results_vgg16v3.csv 19 hours ago by Alison Au add submission details	0.18493	0.18493	☐
results_vgg16v2.csv 2 days ago by Alison Au add submission details	0.16747	0.16747	☐

Insights

In general, CNNs are evidently very powerful tools in image recognition. They can differentiate between cats and dogs over 90% of the time despite the variance in images, e.g. breeds, angles, positioning. The ability to use existing architecture pre-trained on other data sets through transfer learning is also highly beneficial. Rather than starting from scratch, these models have prior learning which improve performance and make it more efficient to build new networks.

From these results, I would learn towards using existing CNN architectures in the future. More layers also seem beneficial. However, too many could lead to overfitting, similar to what was seen with running more epochs which did not improve log loss. Changing hyperparameters while using the same architecture also did not seem to impact results much, at least based on testing two versions of VGG16. However, there are a plethora of parameters to adjust from optimization methods, filters, dense layers, image augmentation, and more. Testing more CNNs would be beneficial to explore, including tuning the hyperparameters and other CNNs like ResNet. The largest restriction is computational power/time. Even when using GPU, these models can take an hour or more to train. As computers become increasingly powerful, CNNs will only become stronger in their capabilities and performance.