Alexandra Schoenberg

Final Report:
Sharks + Cats & Dogs Image Classification

Problem Statement

The initial goal of this project was to answer the question: Can a model correctly identify a shark's species, with 14 options, from its image?

Image recognition is a key function of deep machine learning. Throughout this project, the datasets used consisted of animals, sourced from the internet and identified by species. At the outset, every image was of a shark, and across the different species there were inherent similarities which the model struggled to differentiate. For example, while a hammerhead shark is very distinctive and recognizable, the difference between a whitetip and blacktip shark might be more difficult for a model to understand.

Upon continued investigation and modification of the model and project goals, I ultimately decided to transition to a new dataset consisting of images of cats and dogs, narrowing the image identification categories from 14 to 2 and asking a seemingly simpler task of the model, as cats and dogs have far fewer similarities than closely related shark species.

For the final iteration of this project, I used a wide variety of images of cats and dogs, which I obtained from Kaggle.

Data Wrangling

The raw dataset from Kaggle contained a total of 697 images. This was split into two groups: training and testing. In the training set, there were 278 images of dogs and 279 images of cats, totaling to 557 images. This was further split to create a validation set of 140 images, leaving 417 in the training set. In the test set, there were 70 images each of both dogs and cats, totaling to 140 images. This amounts to a train-test split of approximately 80%-20%.

The dataset was well organized and compiled from the outset. There were no formatting issues and all images were properly imported. As the file naming convention for each of the images was "(cat/dog)_(number).jpg", I was able to write a simple import code to not only append each image to a training or testing data array, but also to one-hot encode a label for each.

Exploratory Data Analysis

In order to confirm that the images were properly imported and labeled for the model, I performed spot checks on the training and testing data arrays. I showed both the image and the one-hot encoded label, confirming that in each case they were a match. In the training and validation sets, a cat was labeled as [1 0] and a dog was labeled as [0 1]. In the testing set, cats and dogs were simply labeled by their species names, as the model would not receive the label information anyway. In every case checked, across both the training and testing sets, the image was a match to its label.

```
[1 0]
<matplotlib.image.AxesImage at 0x7f08662744f0>
```

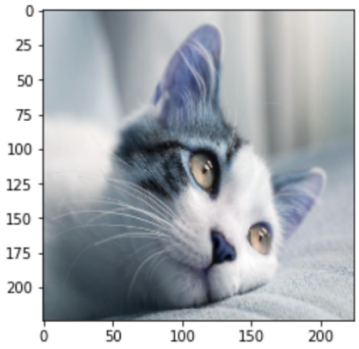

Figure 1: Example Image of Cat, Correctly Labeled [1 0]

I also performed a check on the split of training and validation images, comparing the percentage makeup of each species. As expected and desired, each species made up roughly half of both the training and validation sets.

Modeling

For this image classification project, I decided to use a Convolutional Neural Network model. I used one Sequential model that I had created from scratch and one pre-existing model, the MobileNetV2 in Keras.

I began by instantiating a Sequential Neural Network Model (Model 1) with a defined random seed of 42 for reproducibility, then adding 2D Convolution, Batch Normalization, 2D Max Pooling, Flattening, and Dense layers. I made many adjustments, additions, and subtractions to the model and its parameters, assessing performance and modifying as needed with each of dozens of runs, most ranging from 10 to 50 epochs. I also built early stop and learning rate reduction callbacks into the model, used to make automated adjustments as the model was running if performance metrics were not improving after a defined number of relatively stagnant epochs. In the final iteration of this model, each of the layers listed above, with the exception for the last Dense layer (which would be providing the model output), was assigned a 'relu' activation type. The final Dense layer was assigned a 'softmax' activation type with 2 output categories, corresponding to cats and dogs. The model was compiled with the 'Adam' optimizer and assessed for performance based on accuracy.

Model 1 was set to run for 50 epochs, though ultimately the provided callbacks typically halted its performance after 16. The final version of the model ran for 23 epochs and produced an accuracy of 0.65 for cats and 0.68 for dogs, based on the validation dataset.

Model 2 was a MobileNetV2 model, using example code from a Kaggle competition to identify real versus fake faces. This model included a Global Average Pooling layer, multiple Dense layers, a Batch Normalization layer, and multiple Dropout layers. As with Model 1, Model 2 was compiled with the 'Adam' optimizer and assessed for performance based on accuracy. I also applied the same callbacks to adjust and halt model training as necessary. In the final

iteration of Model 2, there were 16 epochs and the model achieved an accuracy of 0.76 for cats and 0.71 for dogs.

As the validation set results for Model 2 were superior to Model 1, I chose to pursue further analysis of Model 2. I created a graph of the training versus validation accuracy for the model, as shown in Figure 2 below, as well as using the model to predict labels for the testing dataset and outputting a classification report. The final results of the classification report were relatively successful – values for precision, recall, and F-1 score for both cats and dogs all averaged to 0.62.

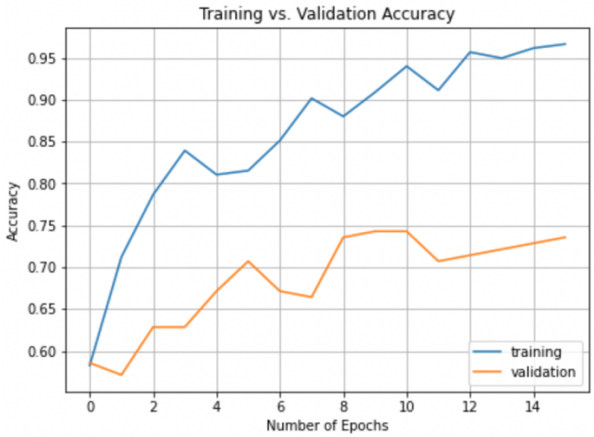

Figure 2: Line Graph of Training vs. Validation Accuracies

Takeaways and Future Steps

Based on the results of Models 1 and 2, this project has been a good start, but does have room for improvement. It is possible to expect to be able to train a model to a higher accuracy than 0.7, and some methods to improve performance could include data augmentation, further training and/or model parameter tweaking, transfer learning from more complex image classification models, or some combination of all these techniques.

Overall, this project was a very solid introduction to the world of image recognition/classification and machine learning in general. Though I was not successful in the original goal of training a model to correctly identify different species of sharks, I had to make many iterative models and identify resultant changes from my adjustments, learn about many different layers and compilers within convolutional neural networks, apply the concept of transfer learning and rewriting existing code or retraining existing models, practice working outside of a Jupyter notebook (the primary tool with which I was previously familiar), and ultimately learn when to shift focus and try something new. The cats and dogs model was much more successful this time, but in the future I hope to be able to get closer to what I originally envisioned with the sharks.