Becca Dura
CS167 Project C

Problem

The problem I am trying to solve is how to best classify ten different species of monkeys. Being able to recognize these ten species of monkeys could be very useful in a variety of situations. Someone doing research on or observing monkeys in the wild could benefit from a monkey classifier like this if they put up cameras in the wild to observe the monkeys and wanted to identify them. A monkey classifier could also be useful at zoos, especially in their research labs, to help identify different species of monkeys. In other words, this monkey classifier would mostly be used for research purposes.

Data

My data set came straight from Kaggle and consists of a training and a validation set with ten species of monkeys. The common names of the ten species of monkeys in the data set are as follows: mantled howler, patas monkey, bald uakari, Japanese macaque, pygmy marmoset, white-headed capuchin, silvery marmoset, common squirrel monkey, black-headed night monkey, and nilgiri langur. Originally, the data set just had a number to identify each species, but I changed it so the monkeys were identified by their common names. In total, there were 1,097 pictures of monkeys in the training set and 272 pictures of monkeys in the validation set. Below are some of the pictures in the data set.

Bald Uakari

White-Headed Capuchin

Common Squirrel Monkey

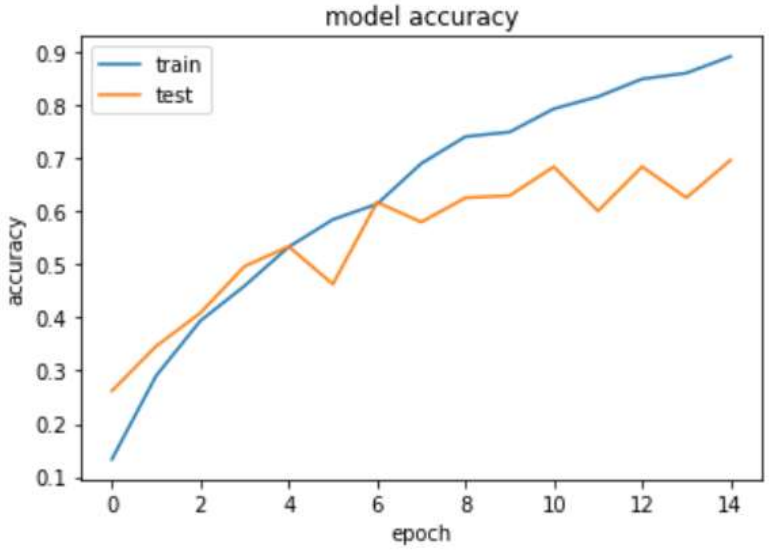
Silvery Marmoset

Becca Dura
CS167 Project C

First Model

For my first model, I took the second model in the example code and modified it to fit the problem at hand. To do that, I changed the model from binary to categorical, changed the sigmoid activation to softmax, and changed the dense layer to have ten units instead of just one. I made those changes because my model is trying to classify ten different species of monkeys instead of just two. The model has two convolutional layers, one with 32 feature maps and one with 64 feature maps, from regions that are 3x3 in the images. It also has two pooling layers, one after each of the convolutional layers, and gets flattened before the fully connected dense layer at the end. The model also has a dropout layer to help prevent overfitting.

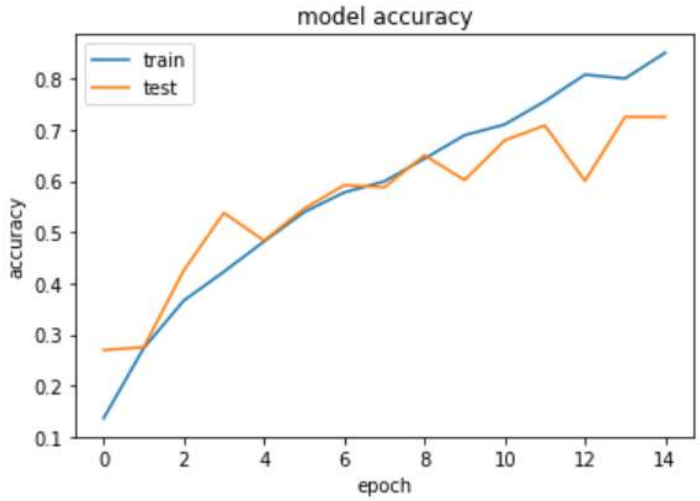
This model seemed to fit decently well, but there were some flaws with it. I trained the model with 15 epochs (approximately 30 minutes), but further training may have been beneficial because the accuracy hadn't been level very long. In other words, I am unsure that I reached the point where further training wouldn't do any good. The highest accuracy of the model was for the 15th epoch, which was the last one, so the accuracy could very well increase with further training. From the plot below, you can see that the training set's accuracy increased more steadily and at a faster rate than the testing set's accuracy. Since the accuracy of the training set is almost 20% higher than the accuracy of the testing set at the end, the model seems to suffer from overfitting. Overall, this model is not necessarily bad, but it could definitely be better.



Second Model

For the second model, I added a third convolutional and pooling layer. The third convolutional layer has 128 feature maps from 3x3 regions in the images. I added this layer to help my model extract more features from the images to help in classifying the monkeys correctly. However, I know adding more layers can lead to overfitting sometimes. Since the accuracy of my model at each epoch seemed to increase from the first to the second model, adding the convolutional and pooling layer was beneficial. Therefore, the additional layers helped my model extract more meaningful features from the images.

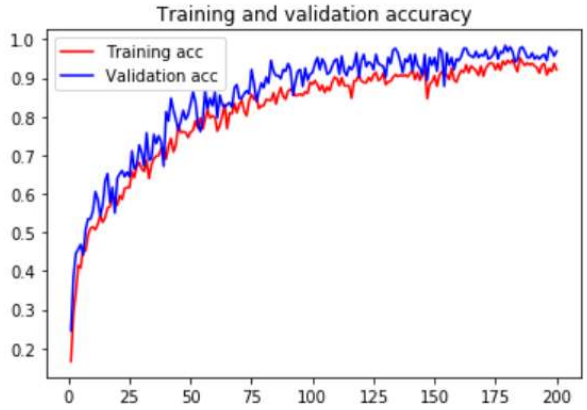
As seen in the plot below, the second model fits much better than the first. Although the highest accuracy (0.725) is only 3% higher than the highest accuracy in the first model (0.695), the second model does not suffer from overfitting as much as the first model. The lack of overfitting can also be seen by the lines for the accuracies of the training and testing sets below. The lines increase at a more similar rate in the second model than in the first model, so they are closer together and there is less overfitting of the model. For this model, I also only trained it with 15 epochs (approximately 30 minutes), but I do think the model would become more accurate with more epochs because the last epoch had the highest accuracy and the accuracies do not seem like they have leveled off yet. I do not believe I reached the point where more epochs would do no good. In other words, my second model with a third convolutional and pooling layer is a better fit than my first model.



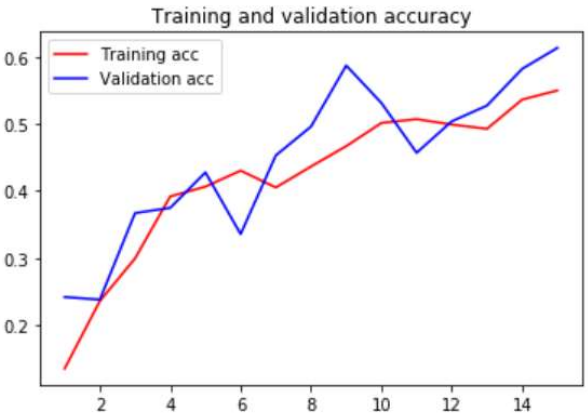
Exploration Part

For the exploration part, I experimented with transfer learning. I pulled a model that a Kaggle user had created for the dataset and tried to make it better. The model I pulled reached 92.53% accuracy on the validation set; however, that was with 200 epochs. The model most likely needed so many epochs because there were so many different species on monkeys to classify and not a lot of features on each money image to help differentiate the monkeys. Since I didn't have the time to run that many epochs, I just looked at the accuracy on the 15th epoch (61.33%) and tried to do better than that. Since I didn't run 200 epochs, it is hard to tell if I actually improved the model or not. The plots for the model I pulled from Kaggle when running 200 and 15 epochs are below.

200 Epochs (Original Model)



15 Epochs (Original Model)



First, I tried adjusting the rate in the dropout layers. The dropout layer is supposed to help overfitting by setting a portion of the input units to zero at each update while training the model. The rate, which I was adjusting, is the fraction of input units set to zero and can be any number between zero and one. I tried adjusting the dropout layers at 0, 0.25, 0.5, and 0.75; however, none of the various combinations of rates that I tried using in the dropout layer increased the accuracy. The model that came the closest to the original model's accuracy had only 60.70% accuracy; however, it could have ended up with a higher accuracy than the original model after 200 epochs.

Next, I tried adjusting the units, or dimensions of the output space, in the dense layer. However, both increasing and decreasing the units make the model significantly worse, meaning the 512 units in the dense layer produces the most accurate classifications of the monkey images.

Overall, neither adjusting the rate in the dropout layers or adjusting the units in the dense layer helped improve the model I pulled from Kaggle within the first 15 epochs. However, more epochs may have shown different results, since the accuracy didn't seem to get high until after 15 epochs for the original model, as shown in the plot above. This most likely means that more epochs does not necessarily lead to the model overfitting, instead it seems to help the model better classify the monkey images as their respective species. That makes sense because the monkey images do not have a lot of different

features that make the various species easily differentiable, so going through more epochs would help to better identify them. In other words, although I was not able to make the model I pulled from Kaggle better through transfer learning based off the accuracy at 15 epochs, the model may have been improved at further epochs.