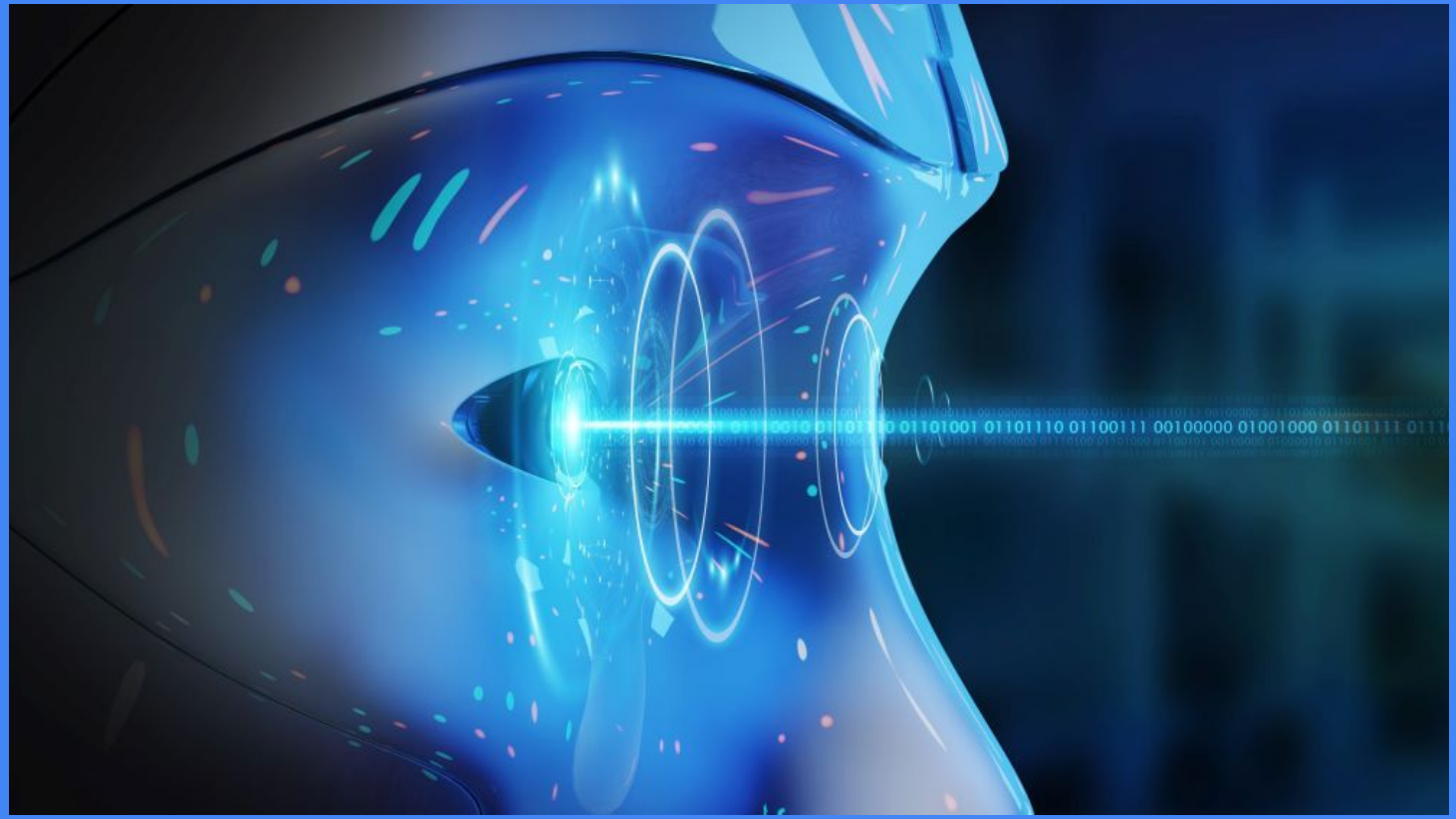
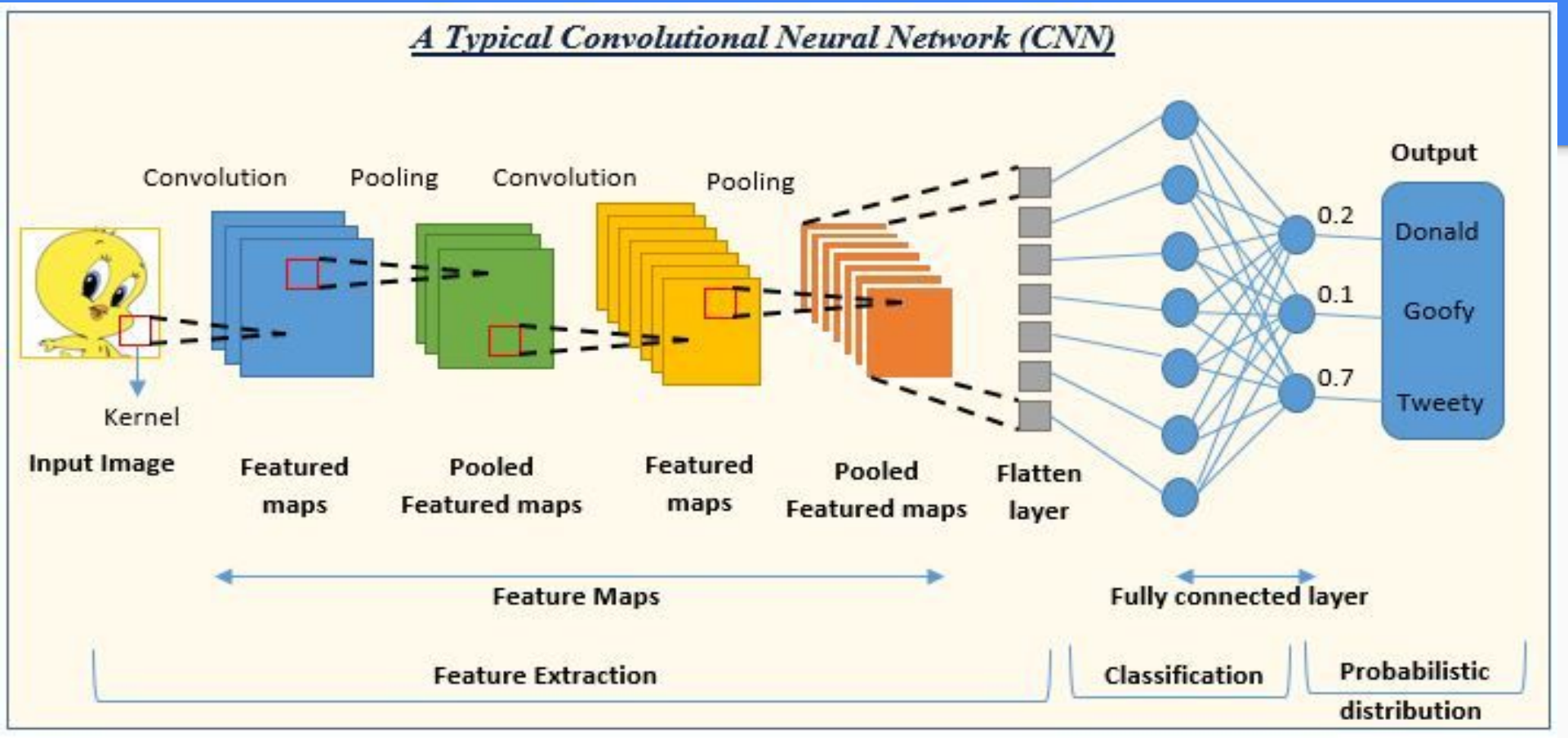
Computer Vision



Dog Breed Classification

Goal: Build a model that will tell us the breed of our furry friends

Convolutional Neural Network



Issues With Our Data

- There were 20 and a half thousand images evenly distributed among the 120 breeds (not enough examples for the network!)
- Image data contained noise that needed filtering

Classification for 120 Breeds

We first started experimenting with our CNN on the entire dataset of 120 breeds.

This was a naive approach as the model produced very lackluster results.

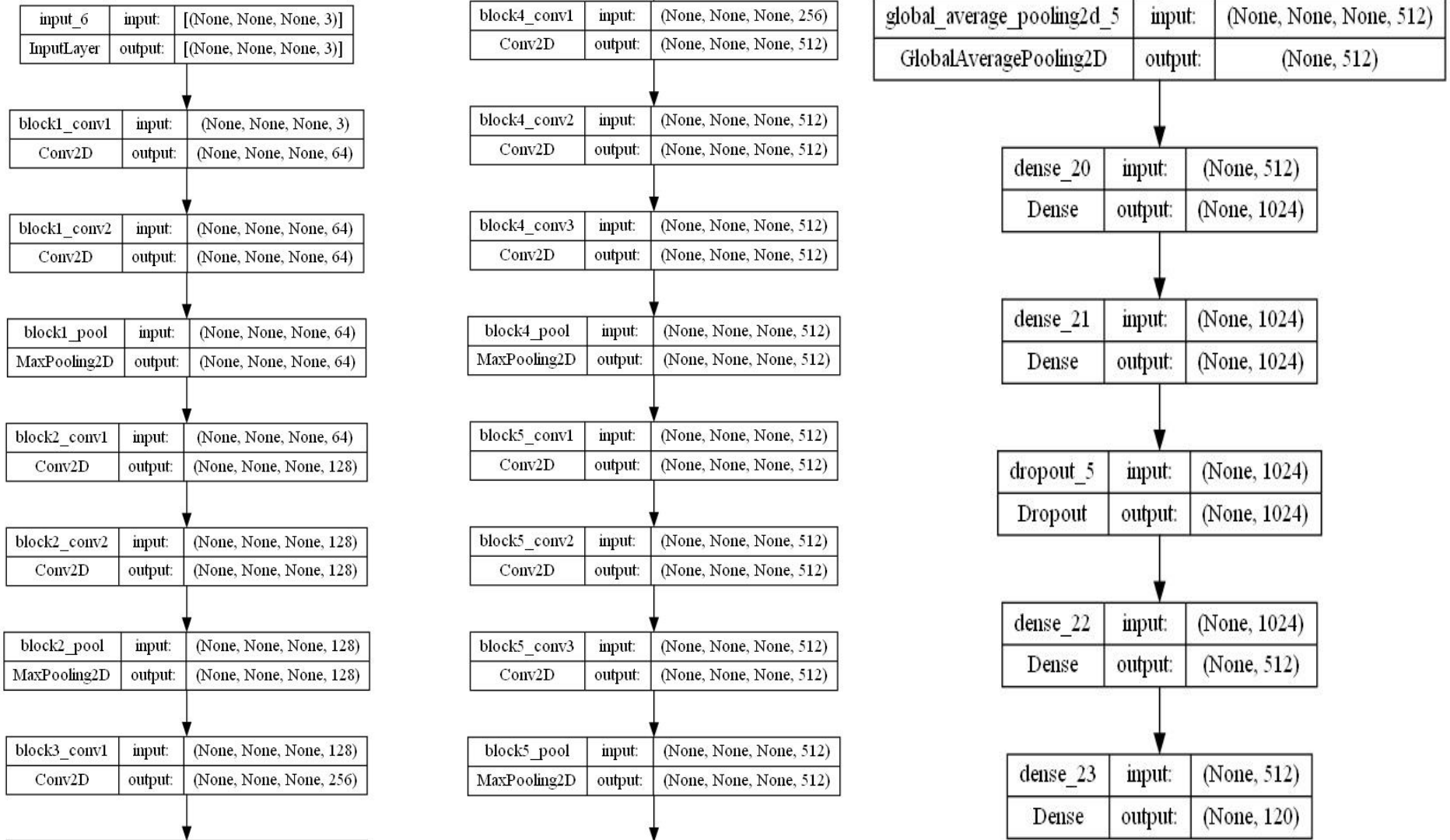
Our accuracy was about 30% for the training dataset

Transfer Learning

Transfer Learning is a technique where you can build upon pre-trained models that have been tested on millions of samples and combine them with your own

This allows us to have a solid foundation for our own model and hopefully allow faster convergence.

We used the VGG16 package (a CNN) from Keras and the ImageNet training set of over 16 million different picture.

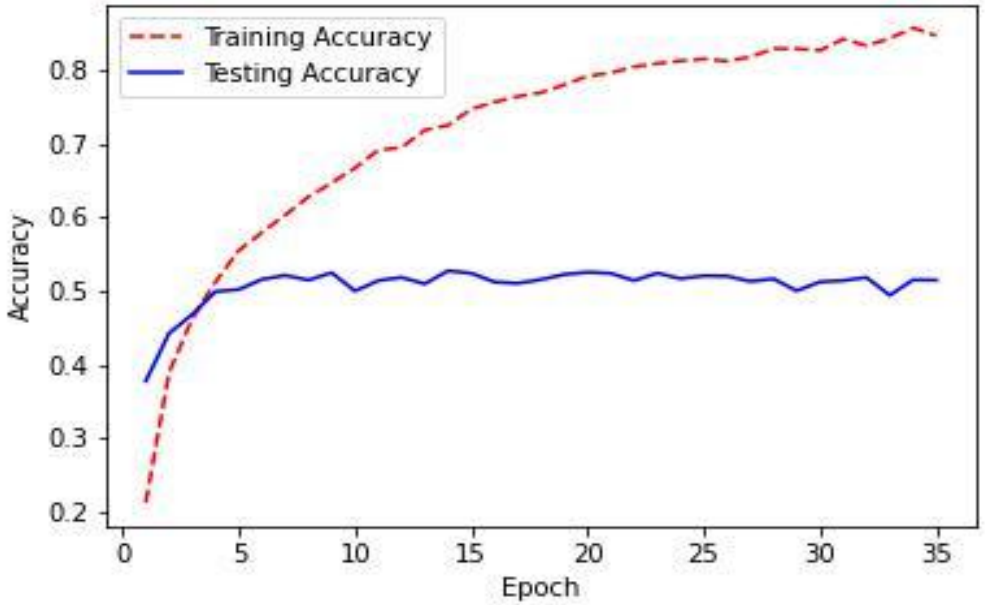

input_6 | input: | [(None, None, None, 3)]
InputLayer | output: | [(None, None, None, 3)]
block1_conv1 | input: | (None, None, None, 3)
Conv2D | output: | (None, None, None, 64)
block1_conv2 | input: | (None, None, None, 64)
Conv2D | output: | (None, None, None, 64)
block1_pool | input: | (None, None, None, 64)
MaxPooling2D | output: | (None, None, None, 64)
block2_conv1 | input: | (None, None, None, 64)
Conv2D | output: | (None, None, None, 128)
block2_conv2 | input: | (None, None, None, 128)
Conv2D | output: | (None, None, None, 128)
block2_pool | input: | (None, None, None, 128)
MaxPooling2D | output: | (None, None, None, 128)
block3_conv1 | input: | (None, None, None, 128)
Conv2D | output: | (None, None, None, 256)
block4_conv1 | input: | (None, None, None, 256)
Conv2D | output: | (None, None, None, 512)
block4_conv2 | input: | (None, None, None, 512)
Conv2D | output: | (None, None, None, 512)
block4_conv3 | input: | (None, None, None, 512)
Conv2D | output: | (None, None, None, 512)
block4_pool | input: | (None, None, None, 512)
MaxPooling2D | output: | (None, None, None, 512)
block5_conv1 | input: | (None, None, None, 512)
Conv2D | output: | (None, None, None, 512)
block5_conv2 | input: | (None, None, None, 512)
Conv2D | output: | (None, None, None, 512)
block5_conv3 | input: | (None, None, None, 512)
Conv2D | output: | (None, None, None, 512)
block5_pool | input: | (None, None, None, 512)
MaxPooling2D | output: | (None, None, None, 512)
global_average_pooling2d_5 | input: | (None, None, None, 512)
GlobalAveragePooling2D | output: | (None, 512)
dense_20 | input: | (None, 512)
Dense | output: | (None, 1024)
dense_21 | input: | (None, 1024)
Dense | output: | (None, 1024)
dropout_5 | input: | (None, 1024)
Dropout | output: | (None, 1024)
dense_22 | input: | (None, 1024)
Dense | output: | (None, 512)
dense_23 | input: | (None, 512)
Dense | output: | (None, 120)

120 Breed Model

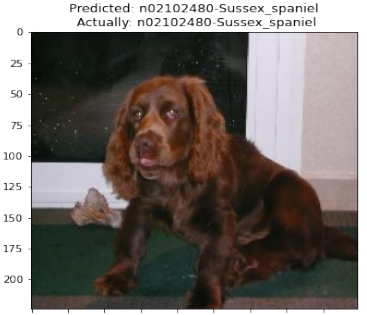
Using the transfer learning technique we were able to get the following accuracies:

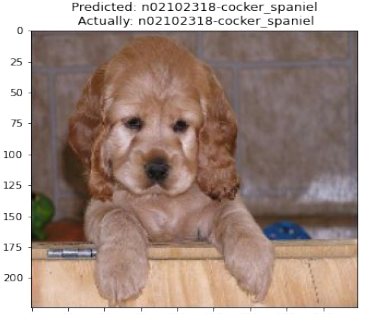
84.66% on the training set

51.5% on the testing set.

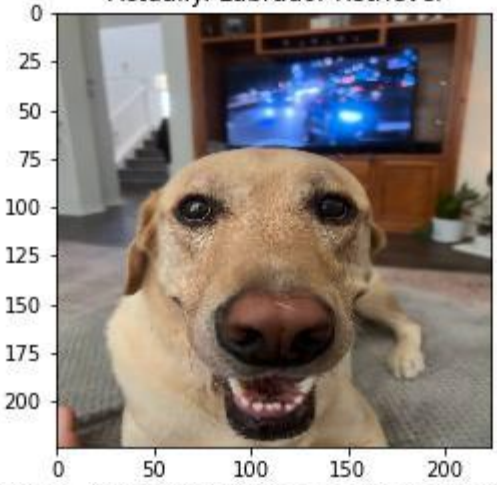


Here are some test predictions from the dataset.

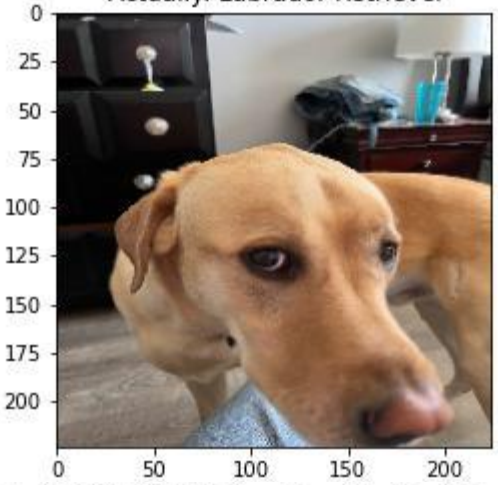




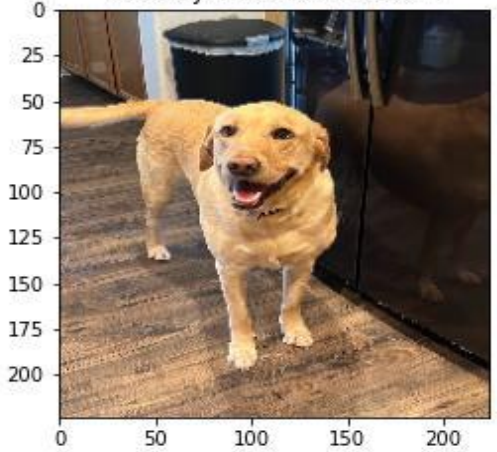
Predicted: ['n02092339-Weimaraner']
Actually: Labrador Retriever



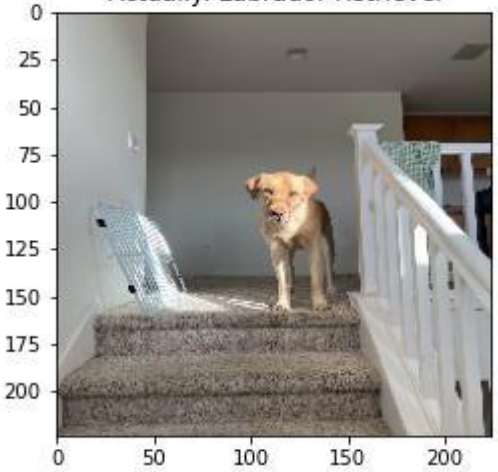
Predicted: ['n02092339-Weimaraner']
Actually: Labrador Retriever



Predicted: ['n02099849-Chesapeake_Bay_retriever']
Actually: Labrador Retriever



Predicted: ['n02093428-American_Staffordshire_terrier']
Actually: Labrador Retriever



How to Improve?

There appears to be overfitting in our model, considering the discrepancy between the testing and training set.

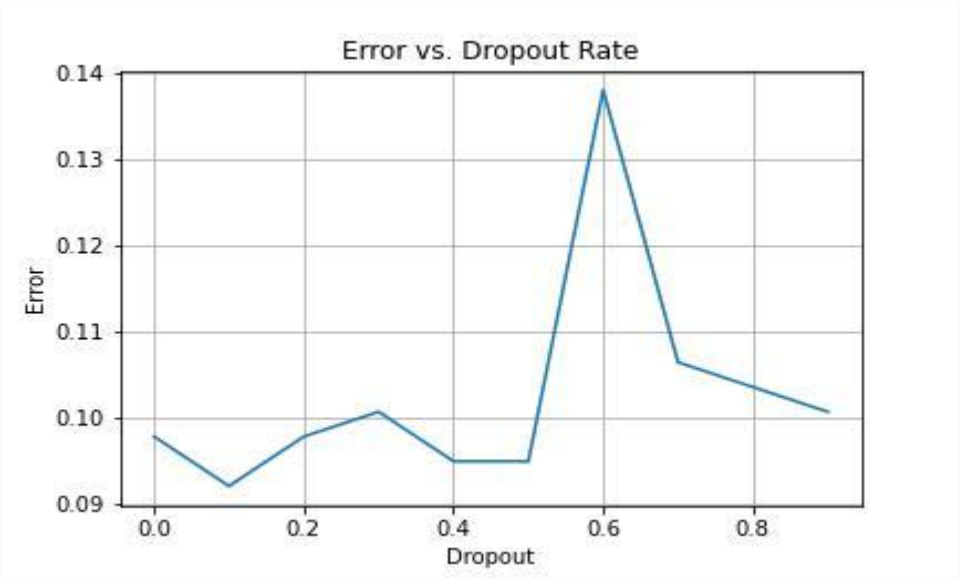
One possible way is to use the augmented data that John had created

The problem is, the data was 11 magnitudes larger than the original so it was not possible to run on our 120 breed classifier

10 Breed Classifier

Using the same methods as before, we used the augmented data and instead only ran the model using 10 different breed.

We experimented with different dropout rates and ended up with using the metric of 0.1

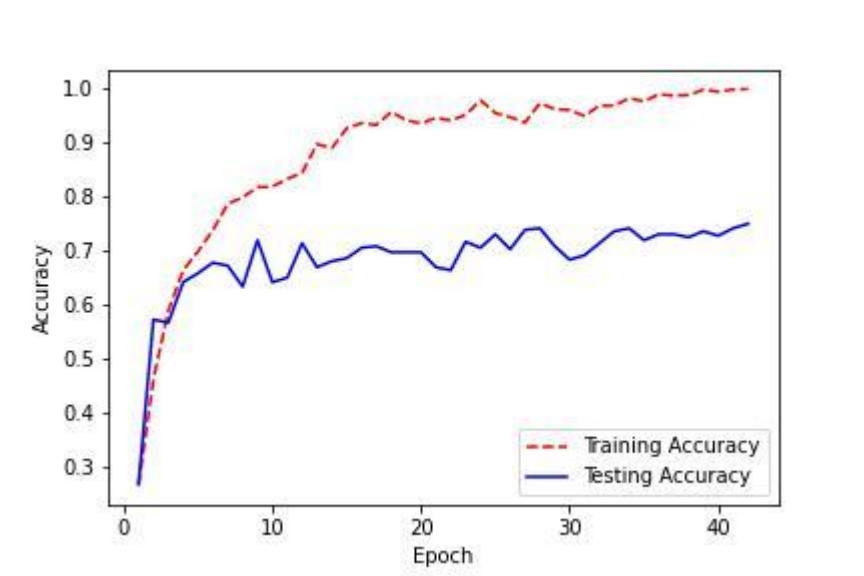


10 Breed Classifier

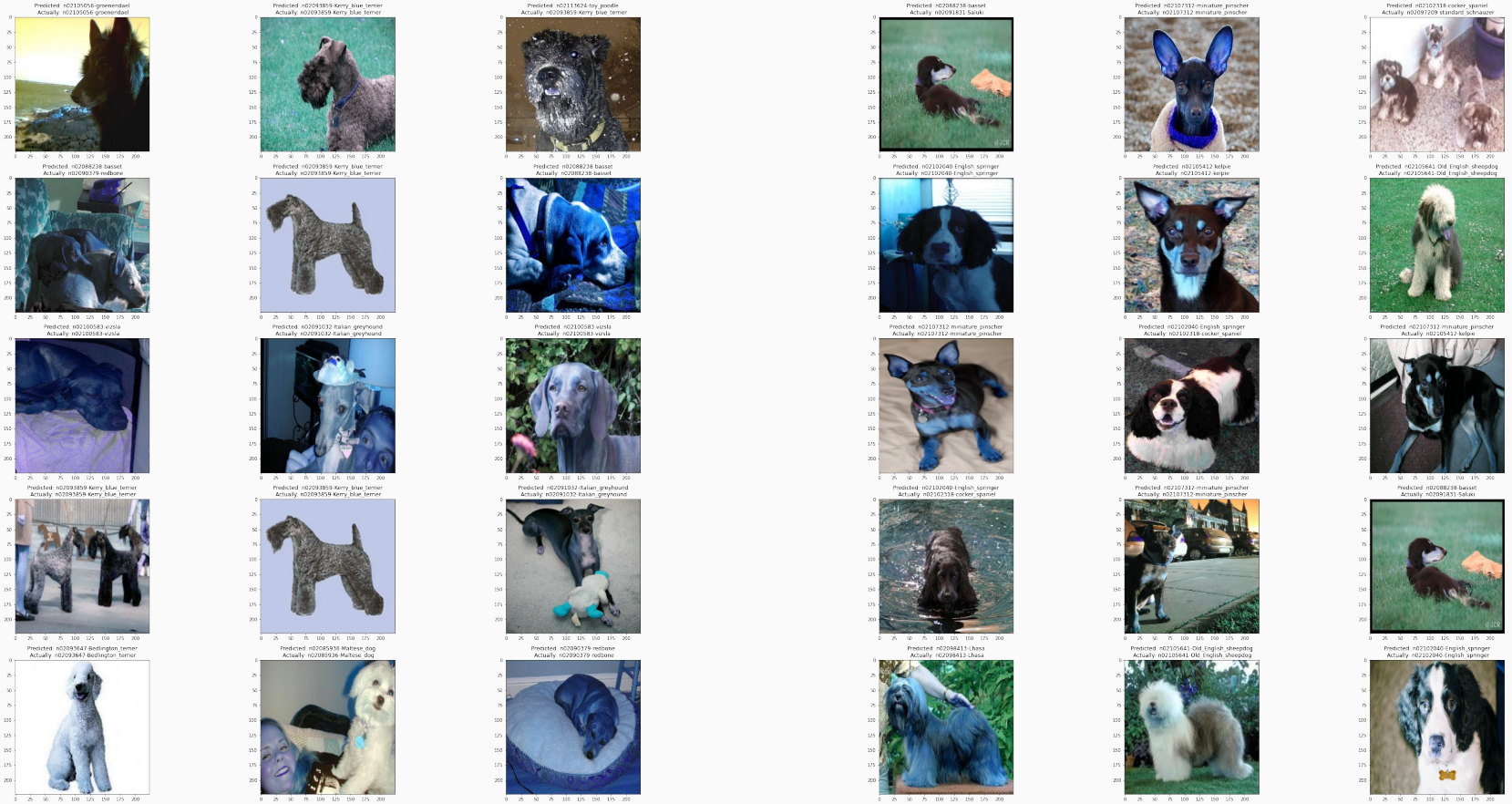
This model performed rather well and by using transfer learning and the augmented data we were able to get the following accuracies:

100% on the training set

75.1% on the testing set.



Some predictions using the 10 breed classifier



Summary

If we were able to use more memory for our model, we could certainly get a lower error rate for our 120 breed classifier. It is simply unfeasible to attempt it without additional GPU processing power.

A 25% error rate for a problem as complex as this is not so bad. Classifying dog breeds in pictures from real world data is a monumental task. Still, there are even more methods we could use to improve our model such as filtering for the images and changing the parameters of the Neural Network.

Questions?