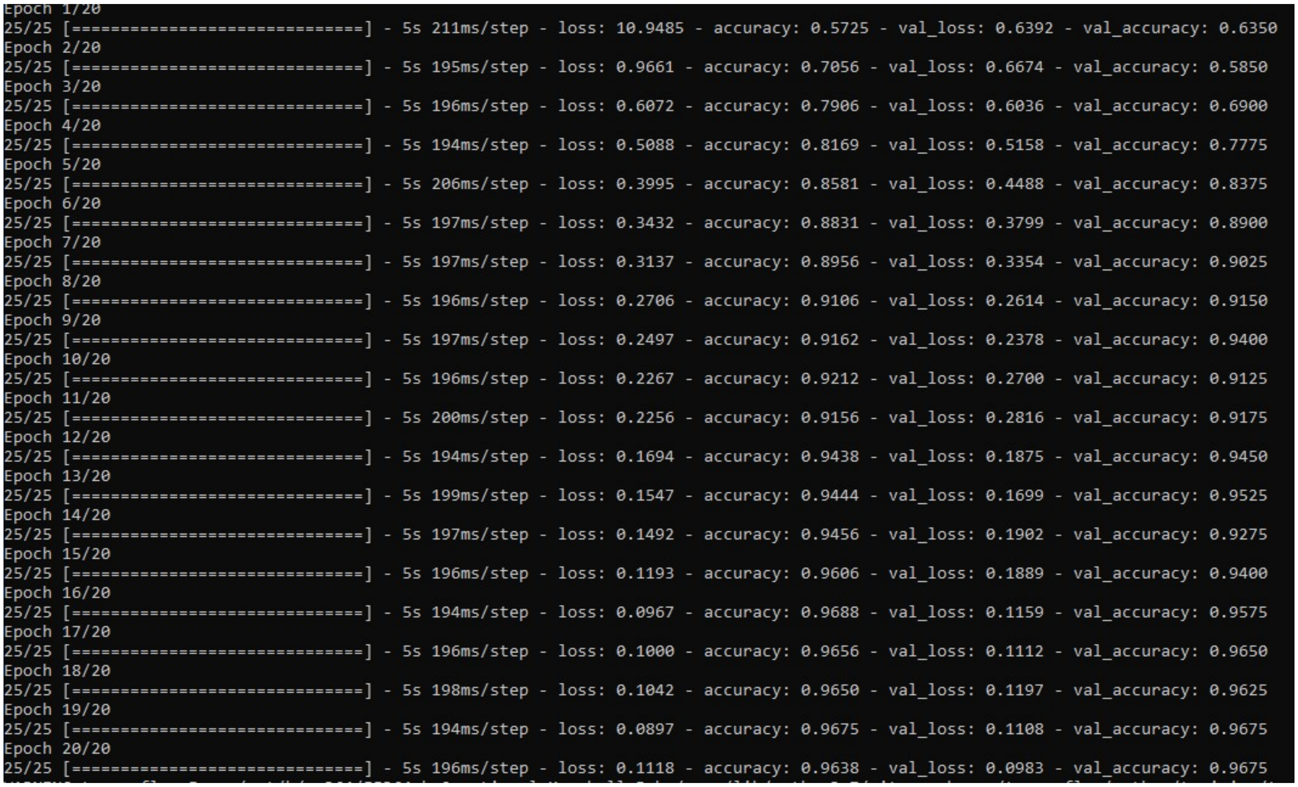
Method:

I utilized a deep learning technique called convolutional neural networks which specializes in image classification. CNN takes advantage of pattern in data to assemble complex patterns using simpler patterns. This technique was inspired by an animal's visual cortex.

Training:

Batch size: 64

Epochs: 20



```
Epoch 1/20
25/25 [==============================] - 5s 211ms/step - loss: 10.9485 - accuracy: 0.5725 - val_loss: 0.6392 - val_accuracy: 0.6350
Epoch 2/20
25/25 [==============================] - 5s 195ms/step - loss: 0.9661 - accuracy: 0.7056 - val_loss: 0.6674 - val_accuracy: 0.5850
Epoch 3/20
25/25 [==============================] - 5s 196ms/step - loss: 0.6072 - accuracy: 0.7906 - val_loss: 0.6036 - val_accuracy: 0.6900
Epoch 4/20
25/25 [==============================] - 5s 194ms/step - loss: 0.5088 - accuracy: 0.8169 - val_loss: 0.5158 - val_accuracy: 0.7775
Epoch 5/20
25/25 [==============================] - 5s 206ms/step - loss: 0.3995 - accuracy: 0.8581 - val_loss: 0.4488 - val_accuracy: 0.8375
Epoch 6/20
25/25 [==============================] - 5s 197ms/step - loss: 0.3432 - accuracy: 0.8831 - val_loss: 0.3799 - val_accuracy: 0.8900
Epoch 7/20
25/25 [==============================] - 5s 197ms/step - loss: 0.3137 - accuracy: 0.8956 - val_loss: 0.3354 - val_accuracy: 0.9025
Epoch 8/20
25/25 [==============================] - 5s 196ms/step - loss: 0.2706 - accuracy: 0.9106 - val_loss: 0.2614 - val_accuracy: 0.9150
Epoch 9/20
25/25 [==============================] - 5s 197ms/step - loss: 0.2497 - accuracy: 0.9162 - val_loss: 0.2378 - val_accuracy: 0.9400
Epoch 10/20
25/25 [==============================] - 5s 196ms/step - loss: 0.2267 - accuracy: 0.9212 - val_loss: 0.2700 - val_accuracy: 0.9125
Epoch 11/20
25/25 [==============================] - 5s 200ms/step - loss: 0.2256 - accuracy: 0.9156 - val_loss: 0.2816 - val_accuracy: 0.9175
Epoch 12/20
25/25 [==============================] - 5s 194ms/step - loss: 0.1694 - accuracy: 0.9438 - val_loss: 0.1875 - val_accuracy: 0.9450
Epoch 13/20
25/25 [==============================] - 5s 199ms/step - loss: 0.1547 - accuracy: 0.9444 - val_loss: 0.1699 - val_accuracy: 0.9525
Epoch 14/20
25/25 [==============================] - 5s 197ms/step - loss: 0.1492 - accuracy: 0.9456 - val_loss: 0.1902 - val_accuracy: 0.9275
Epoch 15/20
25/25 [==============================] - 5s 196ms/step - loss: 0.1193 - accuracy: 0.9606 - val_loss: 0.1889 - val_accuracy: 0.9400
Epoch 16/20
25/25 [==============================] - 5s 194ms/step - loss: 0.0967 - accuracy: 0.9688 - val_loss: 0.1159 - val_accuracy: 0.9575
Epoch 17/20
25/25 [==============================] - 5s 196ms/step - loss: 0.1000 - accuracy: 0.9656 - val_loss: 0.1112 - val_accuracy: 0.9650
Epoch 18/20
25/25 [==============================] - 5s 198ms/step - loss: 0.1042 - accuracy: 0.9650 - val_loss: 0.1197 - val_accuracy: 0.9625
Epoch 19/20
25/25 [==============================] - 5s 194ms/step - loss: 0.0897 - accuracy: 0.9675 - val_loss: 0.1108 - val_accuracy: 0.9675
Epoch 20/20
25/25 [==============================] - 5s 196ms/step - loss: 0.1118 - accuracy: 0.9638 - val_loss: 0.0983 - val_accuracy: 0.9675
```

Test loss: 0.09825825691223145

Test accuracy: 0.9674999713897705

Results:

When running the model on all 2000 images in combined dogs and cats folder, I was able to get an accuracy of 97.4%. When training the data, I was always around a 95% for final test accuracy. To decrease overfitting when training, I added a dropout threshold to each of my 3 convolutional layers. When testing with new images I downloaded and edited, I was able to achieve over 85% accuracy. This 15% difference is probably due to the small dataset as well as a bit of overfitting that was not dropped.