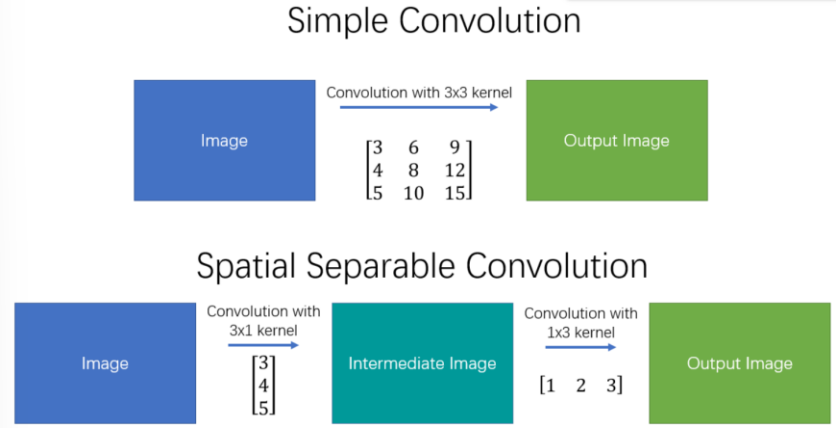
Assignment 2– Mukhtar, 2022



Splitting the Dataset to 1000 for training and 500 for validation, and 500 for testing

I have split the Dataset into 1000 for training and 500 for validation, and 500 for testing and built my model

Model Building:

In building the model, I chose separable convolution layers. For selecting this model, we assume that different channels are highly independent while the intermediate activations are highly correlated. I also assume that there is a possibility of overfitting. As a panacea, I both dropout layers at the end and also used data augmentation preferred techniques. The batch size is now changed to 64 from 32.

The parameters are now reduced:

```
=================================================================
Total params: 508,584
Trainable params: 508,322
Non-trainable params: 262
_________________________________________________________________
```

Testing the base model and then improving the model by changing the training size, Data augmentation, and using a pre-trained model:

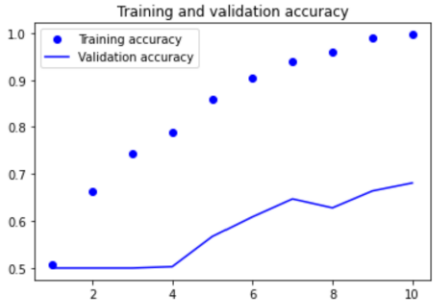
The optimizer that I now use is "adam."

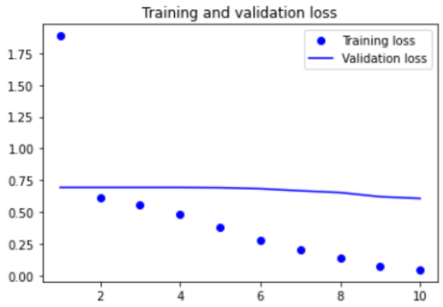
I now Display the shapes of the data and labels yielded by the Dataset that results in ;

data batch shape: (64, 180, 180, 3) and labels batch shape: (64,). Using ten epochs to train the model resulted in the validation accuracy and validation loss dropping with each epoch which suggests the model isn't overfitting. With more epochs, the model would continue improving.

Displaying curves of loss and accuracy during training gives us the following:

```
Epoch 9/10
32/32 [==============================] - 14s 407ms/step - loss: 0.0761 - accuracy: 0.9885 - val_loss: 0.6200 - val_accuracy: 0.6640
Epoch 10/10
32/32 [==============================] - 14s 407ms/step - loss: 0.0451 - accuracy: 0.9965 - val_loss: 0.6064 - val_accuracy: 0.6810
```



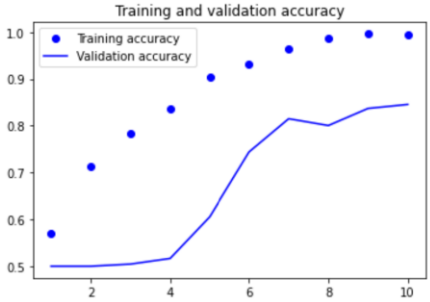


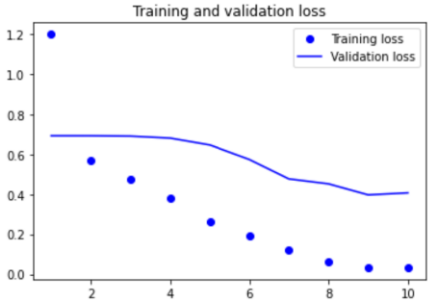
We now evaluate the model on the test set. Model accuracy is right in line with the validation accuracy of model prediction, about 2/3 being predicted correctly. More epochs would increase model performance since it hasn't overfit yet. Ideally, with the right computing power, I'd set the number of epochs and graph validation accuracy to see if the model has overfitted, then set the optimal number of epochs

Increasing the training dataset to 1500 per assignment instructions and re-training the model:

We increased the training dataset to 1500 per assignment instructions and re-training the model. Expanding the training sample size drastically increased validation accuracy from 68% to 84% and reduced validation loss with the same number of epochs; however, accuracy on the test data set only increased from 68% to 69%, as depicted below.

```
32/32 [==============================] - 5s 104ms/step - loss: 0.6325 - accuracy: 0.6880
Test accuracy: 0.688
```



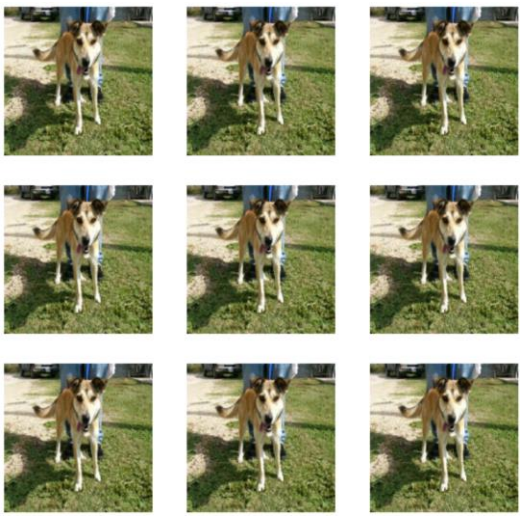


Increasing the size of the training sample drastically increased validation accuracy from 68% to 84% and reduced validation loss with the same number of epochs. However, accuracy on the test data set only increased from 68% to 69%

Using a pre-trained convolution model and using data augmentation as per assignment instructions

I now repeated steps 1-3 and used a pre-trained convolution model and data augmentation as instructed in the assignment. Thereafter, I added a data augmentation and a classifier to the convolution base.

The image below indicates augmented images of "Dog."



To get the best performance, I now limited the model to 10 epochs in training the regularized convnet to compare it with the model trained from scratch.

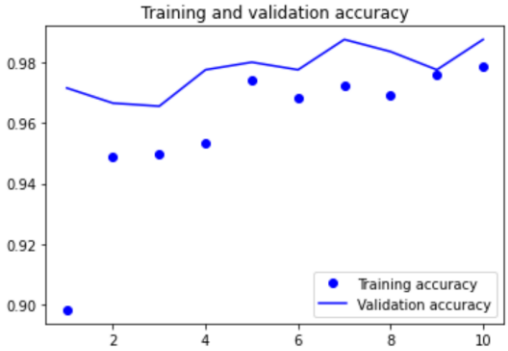
The outcome indicated an improved accuracy of 0.9753 from 0.9245 and a validity accuracy of 0.975 from 0.8360. It's almost 0.98, close to the testing accuracy of 0.975. Given the result, we may conclude that it's not overfitting.

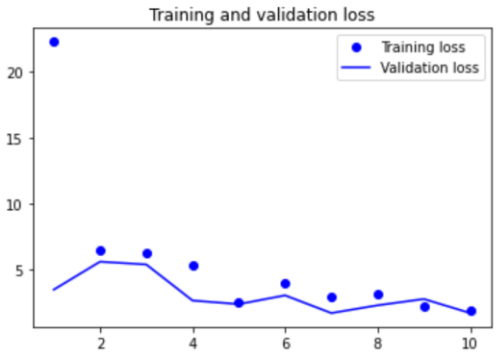
From:

```
63/63 [==============================] - 6s 96ms/step - loss: 0.2114 - accuracy: 0.9335 - val_loss: 0.6480 - val_accuracy: 0.8530
Epoch 99/100
63/63 [==============================] - 6s 93ms/step - loss: 0.1995 - accuracy: 0.9380 - val_loss: 0.6140 - val_accuracy: 0.8500
Epoch 100/100
63/63 [==============================] - 6s 94ms/step - loss: 0.2026 - accuracy: 0.9245 - val_loss: 0.6327 - val_accuracy: 0.8360
```

To:

```
47/47 [==============================] - 21s 435ms/step - loss: 2.9166 - accuracy: 0.9723 - val_loss: 1.6310 - val_accuracy: 0.9875
Epoch 8/10
47/47 [==============================] - 21s 431ms/step - loss: 3.0849 - accuracy: 0.9693 - val_loss: 2.2239 - val_accuracy: 0.9835
Epoch 9/10
47/47 [==============================] - 21s 430ms/step - loss: 2.1209 - accuracy: 0.9757 - val_loss: 2.7023 - val_accuracy: 0.9775
Epoch 10/10
47/47 [==============================] - 21s 428ms/step - loss: 1.7904 - accuracy: 0.9787 - val_loss: 1.6330 - val_accuracy: 0.9875
```



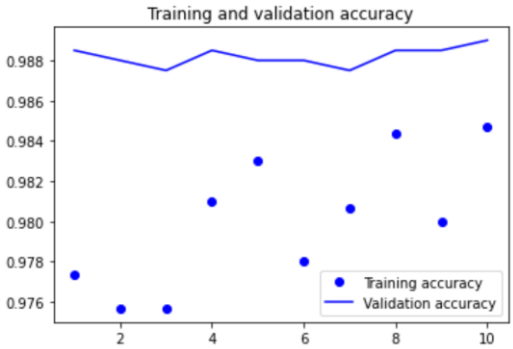


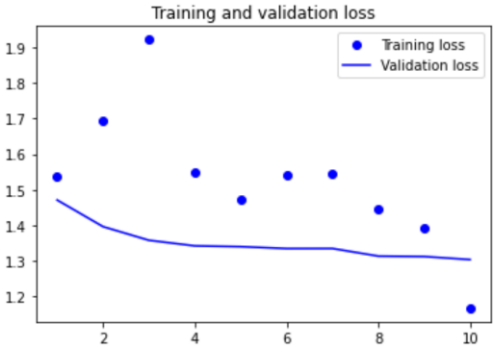
Here we can clearly see the changes in validation accuracy from the previous model. It's almost 0.98, which is close to the testing accuracy of 0.9787. So we can conclude that it's not overfitting.

In order to fine-tune a pre-trained model, we have to adjust the learning rate of the model. The learning rate has an impact on finding local minimums and maximums. Large adjustments to the learning rate will have drastic impacts on the model.

```
Epoch 7/10
47/47 [==============================] - 20s 427ms/step - loss: 1.5447 - accuracy: 0.9807 - val_loss: 1.3344 - val_accuracy: 0.9875
Epoch 8/10
47/47 [==============================] - 21s 434ms/step - loss: 1.4456 - accuracy: 0.9843 - val_loss: 1.3131 - val_accuracy: 0.9885
Epoch 9/10
47/47 [==============================] - 21s 433ms/step - loss: 1.3908 - accuracy: 0.9800 - val_loss: 1.3119 - val_accuracy: 0.9885
Epoch 10/10
47/47 [==============================] - 22s 456ms/step - loss: 1.1668 - accuracy: 0.9847 - val_loss: 1.3033 - val_accuracy: 0.9890
```

We now Display curves of loss and accuracy during training





We can see here that there is a slight change from the last model in accuracy and testing accuracy.
