MIS 64061 Assignment_2: Exploring Convolution Networks (convnets) on Image Data

Eyob Tadele

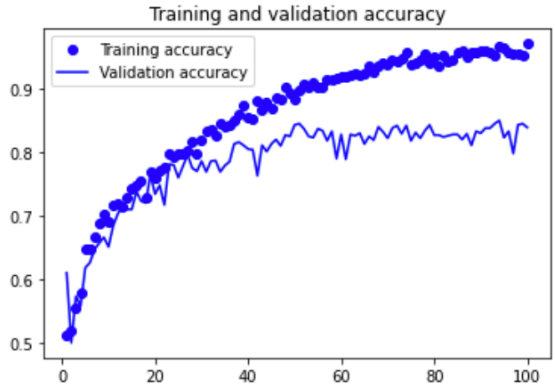
03/27/2022

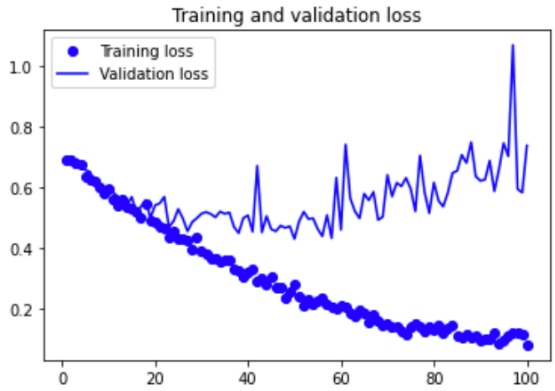
Project Objective

The objective of this assignment is to explore and extend the fundamental understanding of concepts in deep learning for computer vision. This is done by changing and modifying some of the convolution network components, with emphasis on trying out varying size of training sets. The Cats and Dogs dataset is used for this project. This project utilizes two approches, training a network from scratch, or using a pre-trained convnet. In order to reduce the risk of overfitting due to small datasets, techniques such as augmentation, regularization, and drop out will be applied.

The following runs of the convnet models will vary depending on the sizes of the training sets selected and its relationship with the choice of models built from scratch versus pre-trained models.

1. Initially starting with a training set of 1000, a validation set of 500, and a test set of 500 from the Cats-vs-Dogs dataset. In order to reduce overfitting and imporve performance, the first approch applied was data augmentation where we can include random transformations on the training set so as to expose the model to more variations of the image. As such, the augmentation was achieved by applying a horizontal flipping to a random 50% of the images, rotating the image by a random value in the range of -36 to +36 degrees, and zoom in/out the image in the range of -20% to +20%. Additionally, a dropout layer was applied to our model right before the densely connected classifier.



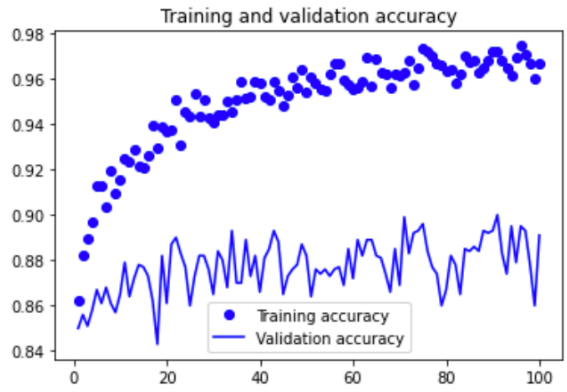


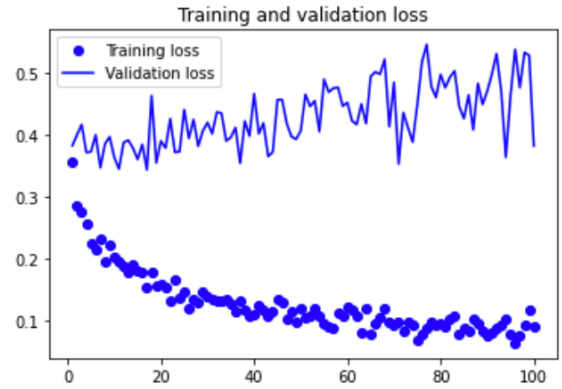
As can be observed from the results above, validation set starts to overfit at about 50 epochs. Also, the accuracy on the test set is at **83.8%** as indicated below.

In [15]:
```
test_model = keras.models.load_model(
"convnet_from_scratch_with_augmentation_dropout_Dataset1.keras")
test_loss, test_acc = test_model.evaluate(test_dataset1)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 2s 51ms/step - loss: 0.4147 - accuracy: 0.8380
Test accuracy: 0.838
```

2. Increasing training sample size to 2000, while maintaining the validation and test sets:



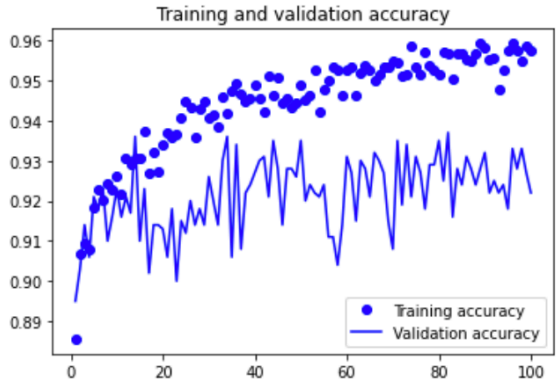


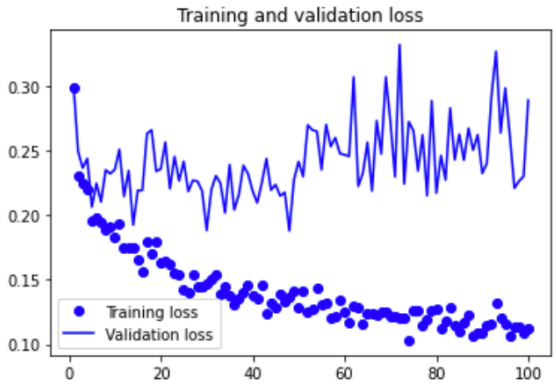
As the above results indicate, validation accuracy/loss has slightly improved and it tends to start overfitting around 90 epochs. The test accuracy has also shown a slight improvement to about **84.7%** as indicated below.

In [18]:
```
test_model = keras.models.load_model(
"convnet_from_scratch_with_augmentation_dropout_Dataset2.keras")
test_loss, test_acc = test_model.evaluate(test_dataset2)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 2s 52ms/step - loss: 0.4505 - accuracy: 0.8470
Test accuracy: 0.847
```

3. Further increasing the training sample to 5000 images and keeping validation and training sets the same. Based on the trend from the above two steps increasing the size of the training set is expected to improve accuracy of the model.





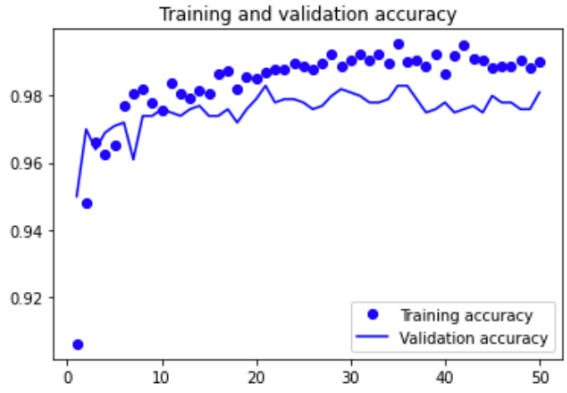
The result above shows that the model above starts to overfit at about 30 epochs, but test accuracy improved to about **92.7%**.

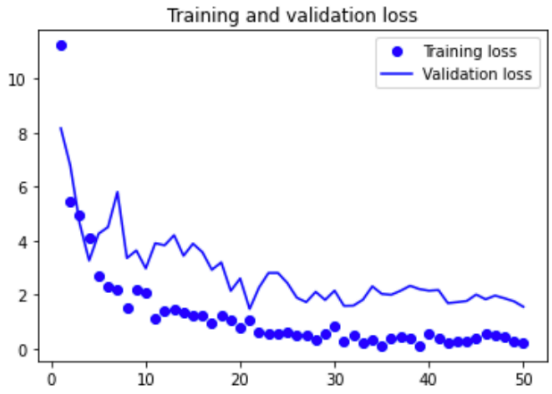
In [21]:
```
test_model = keras.models.load_model(
"convnet_from_scratch_with_augmentation_dropout_Dataset3.keras")
test_loss, test_acc = test_model.evaluate(test_dataset3)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 2s 53ms/step - loss: 0.2109 - accuracy: 0.9270
Test accuracy: 0.927
```

4. Applying a pre-trained network to the datasets from steps 1-3 Utilizing a model that was previously trained on a large dataset, especially on a large-scale image classification task can be an effective approch to deep learning on a small dataset. In this specific case, a large convnet trained on the ImageNet dataset (1.4 million labeled images and 1,000 different classes). ImageNet contains many animal classes, including different species of cats and dogs, and you can thus expect it to perform well on the dogs-versus-cats classification problem. It will use the VGG16 architecture.

4.1. Using the pretrained model on the original 1000 training dataset, we can see an improvement in validation, as well as test accuracy as shown below:



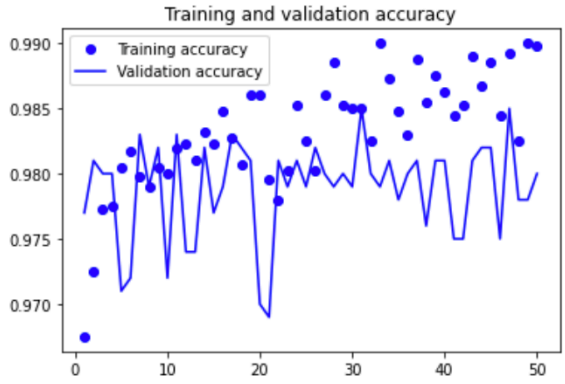


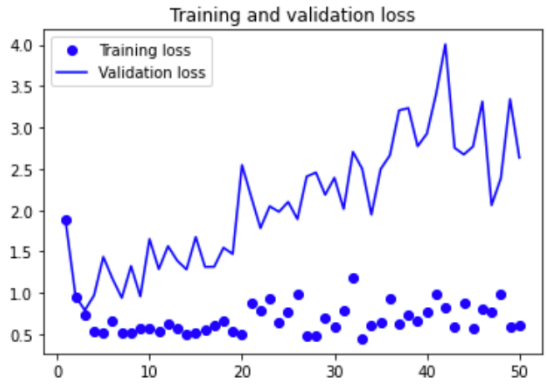
In [28]:
```
test_model = keras.models.load_model(
    "feature_extraction_with_data_augmentation_1.keras")
test_loss, test_acc = test_model.evaluate(test_dataset1)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 9s 268ms/step - loss: 2.2952 - accuracy: 0.9750
Test accuracy: 0.975
```

It can be observed that validation accuracy reaches to about **98%** and test accuracy at **97.5%**. It is a marked improvement over the above three experiments.

4.2. Using the pretrained model on a training dataset of 2000, we can see an improvement in validation, as well as test accuracy as shown below:



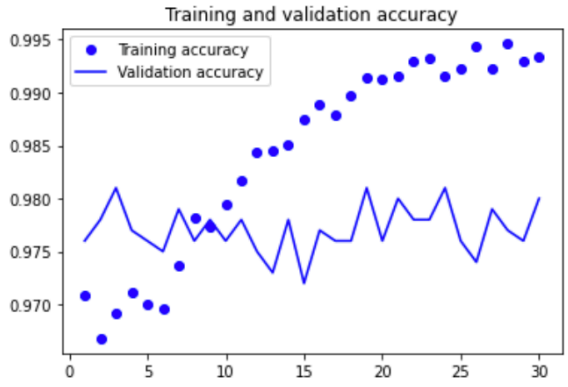


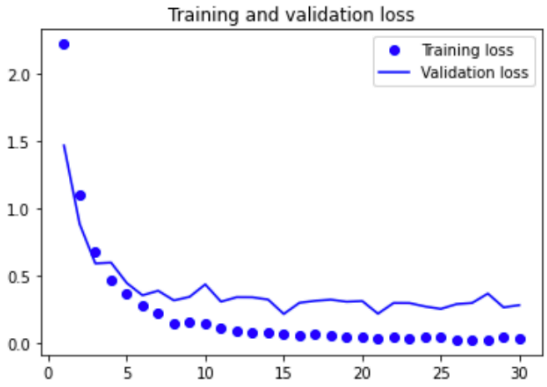
In [32]:
```
test_model = keras.models.load_model(
    "feature_extraction_with_data_augmentation_2.keras")
test_loss, test_acc = test_model.evaluate(test_dataset2)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 9s 268ms/step - loss: 1.0348 - accuracy: 0.9660
Test accuracy: 0.966
```

We can infer from the results above that the pretrained model still has an improvement of on the 'buld from scratch model' at a test accuracy of **96.6%**. But it is a little unclear on the overfitting figure, as it looks unstable.

4.3. A further fine-tuning is applied to the pretrained model to improve the model. This was done by increasing the training dataset to 5000, reducing the epoch to 30, freezing the top layers, and lowering the learning rate to limit the magnitude of the modifications. The results of these modifications are as follows:





In [40]:
```
test_model = keras.models.load_model(
    "fine_tuning.keras")
test_loss, test_acc = test_model.evaluate(test_dataset3)
print(f"Test accuracy: {test_acc:.3f}")
```

```
32/32 [==============================] - 9s 268ms/step - loss: 0.3845 - accuracy: 0.9820
Test accuracy: 0.982
```

The results above show that the pre-trained model with the fine-tuning modifications can increase the validation accuracy to over **98%** and a test accuracy of **98.2%**.

Conclusion:

Based on a series of observations made by changing the size of the training datasets, we can infer that performance of the model improves as the size of the dataset increases in both 'train-from-scratch' and 'pre-trained' models. However, since the size of the dataset is small, overfitting becomes an issue. In order to address this problem and improve performance, data augmentation and dropout are applied to the train from scratch model. These methods certainly helped in reducing overfitting and improve model performance. In the case of a pre-trained model, it shows a marked improvement over the 'build-from-scratch' model. The pre-trained model also incorporated the concept of freezing and lowering the learning rate to further improve the model. To sum up, the larger the training dataset, the more likelihood for the model to have a higher accuracy and parallel to that modifications can be made to the model aiding in prediction/classification accuracy.

Reference:

A detailed run of all the relevant codes are referenced below for further information.

Choose Files no files selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

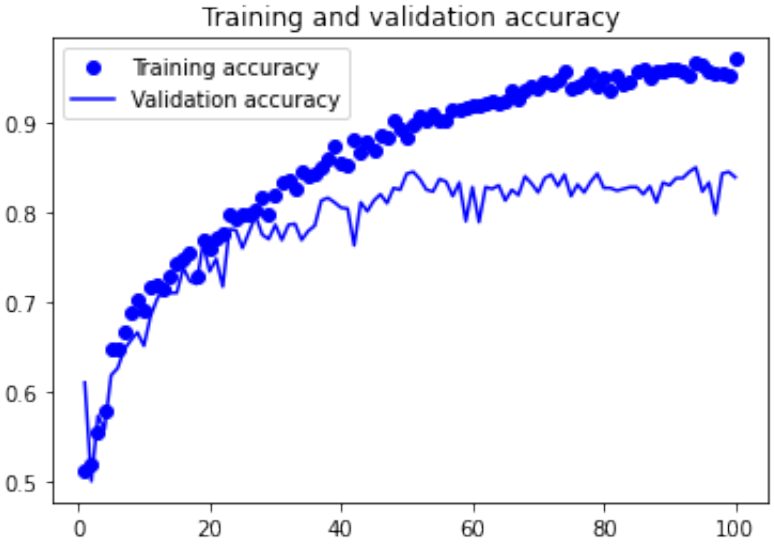
```
Saving kaggle.json to kaggle.json
```

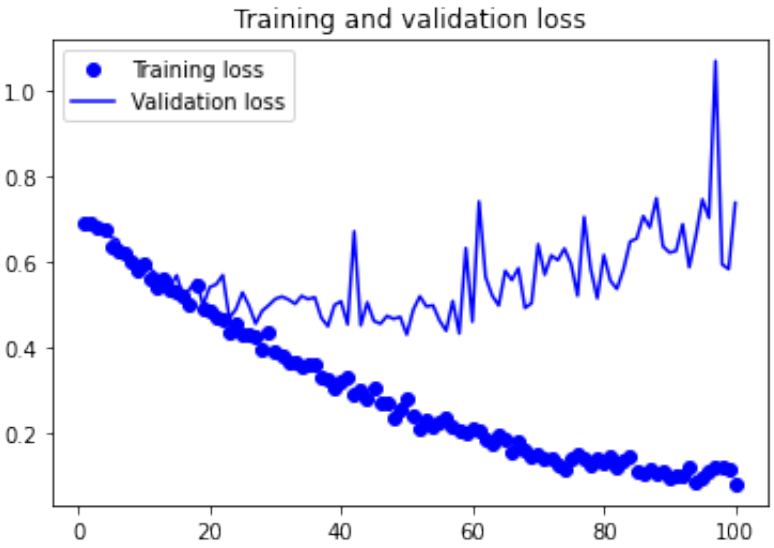
Out[1]:
```
{'kaggle.json': b'{"username":"eytadele","key":"e5ce6664a92ff3d4b4fcad4d2acd51f3"}'}
```

```
Downloading dogs-vs-cats.zip to /content
 98% 797M/812M [00:03<00:00, 258MB/s]
100% 812M/812M [00:03<00:00, 250MB/s]
```

```
Found 2000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 4000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 10000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
Found 1000 files belonging to 2 classes.
```

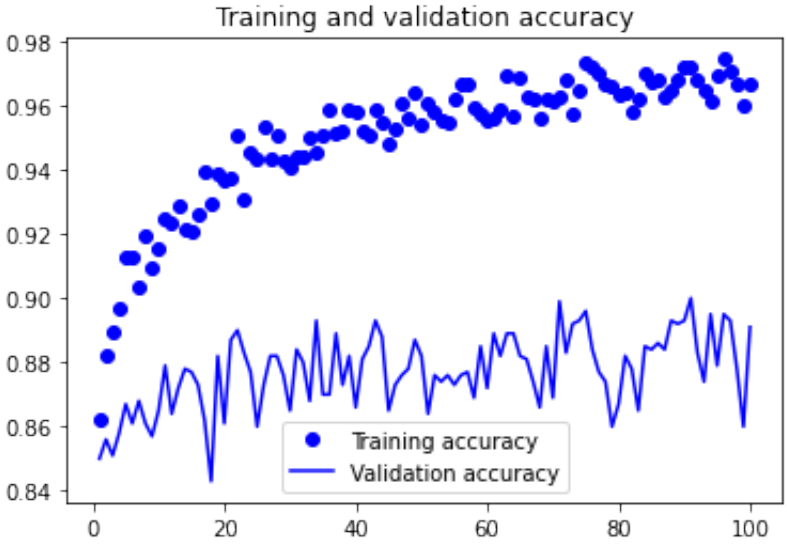
```
Epoch 1/100
63/63 [==============================] - 22s 162ms/step - loss: 0.693
4 - accuracy: 0.5125 - val_loss: 0.6872 - val_accuracy: 0.6100
Epoch 2/100
63/63 [==============================] - 9s 143ms/step - loss: 0.6914
- accuracy: 0.5180 - val_loss: 0.6915 - val_accuracy: 0.5000
Epoch 3/100
63/63 [==============================] - 9s 144ms/step - loss: 0.6806
- accuracy: 0.5555 - val_loss: 0.6858 - val_accuracy: 0.5730
Epoch 4/100
63/63 [==============================] - 9s 145ms/step - loss: 0.6776
- accuracy: 0.5775 - val_loss: 0.6824 - val_accuracy: 0.5580
Epoch 5/100
63/63 [==============================] - 10s 146ms/step - loss: 0.634
2 - accuracy: 0.6470 - val_loss: 0.6574 - val_accuracy: 0.6180
Epoch 6/100
63/63 [==============================] - 9s 146ms/step - loss: 0.6284
- accuracy: 0.6480 - val_loss: 0.6523 - val_accuracy: 0.6260
Epoch 7/100
63/63 [
```

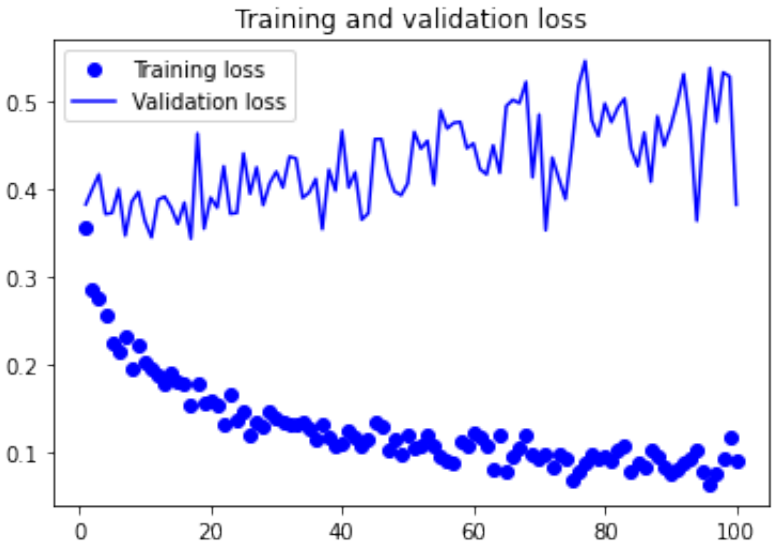





```
32/32 [==============================] - 2s 51ms/step - loss: 0.4147
- accuracy: 0.8380
Test accuracy: 0.838
```

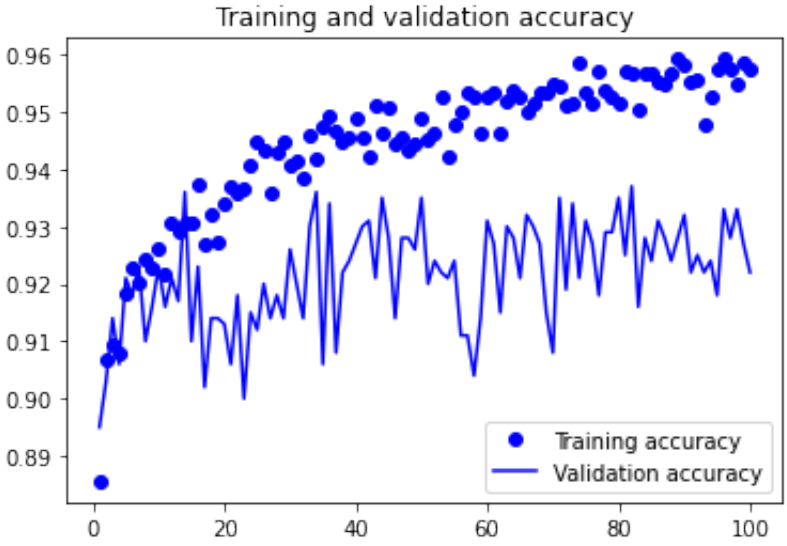
```
Epoch 14/100
125/125 [==============================] - 16s 128ms/step - loss: 0.1
920 - accuracy: 0.9218 - val_loss: 0.3791 - val_accuracy: 0.8780
Epoch 15/100
125/125 [==============================] - 16s 128ms/step - loss: 0.1
825 - accuracy: 0.9205 - val_loss: 0.3608 - val_accuracy: 0.8770
Epoch 16/100
125/125 [==============================] - 16s 128ms/step - loss: 0.1
783 - accuracy: 0.9260 - val_loss: 0.3849 - val_accuracy: 0.8730
Epoch 17/100
125/125 [==============================] - 17s 130ms/step - loss: 0.1
539 - accuracy: 0.9392 - val_loss: 0.3439 - val_accuracy: 0.8620
Epoch 18/100
125/125 [==============================] - 17s 131ms/step - loss: 0.1
788 - accuracy: 0.9298 - val_loss: 0.4636 - val_accuracy: 0.8430
Epoch 19/100
125/125 [==============================] - 17s 130ms/step - loss: 0.1
567 - accuracy: 0.9390 - val_loss: 0.3553 - val_accuracy: 0.8820
Epoch 20/100
125/125 [==============================] - 17s 131ms/step - loss: 0.1
```

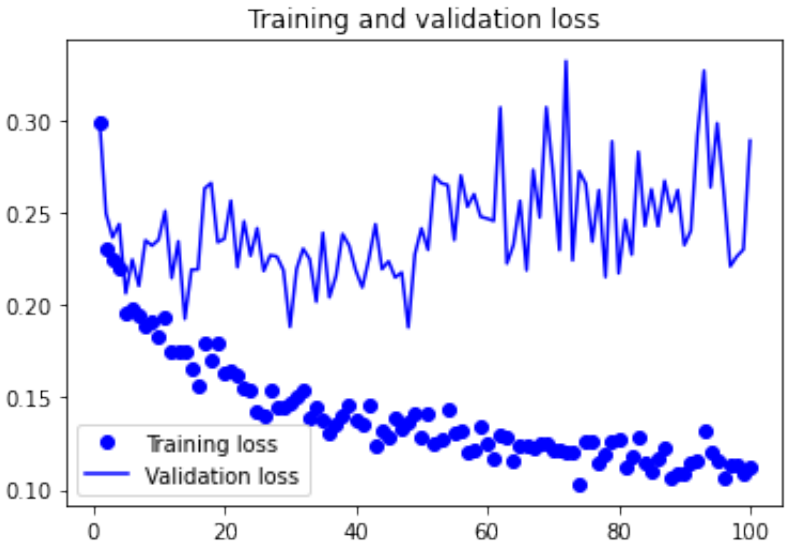





```
32/32 [==============================] - 2s 52ms/step - loss: 0.4505
- accuracy: 0.8470
Test accuracy: 0.847
```

```
Epoch 1/100
313/313 [==============================] - 38s 120ms/step - loss: 0.2
982 - accuracy: 0.8856 - val_loss: 0.3011 - val_accuracy: 0.8950
Epoch 2/100
313/313 [==============================] - 39s 121ms/step - loss: 0.2
300 - accuracy: 0.9068 - val_loss: 0.2494 - val_accuracy: 0.9030
Epoch 3/100
313/313 [==============================] - 38s 121ms/step - loss: 0.2
246 - accuracy: 0.9095 - val_loss: 0.2365 - val_accuracy: 0.9140
Epoch 4/100
313/313 [==============================] - 38s 120ms/step - loss: 0.2
195 - accuracy: 0.9080 - val_loss: 0.2437 - val_accuracy: 0.9060
Epoch 5/100
313/313 [==============================] - 38s 119ms/step - loss: 0.1
956 - accuracy: 0.9185 - val_loss: 0.2063 - val_accuracy: 0.9210
Epoch 6/100
313/313 [==============================] - 38s 119ms/step - loss: 0.1
983 - accuracy: 0.9228 - val_loss: 0.2245 - val_accuracy: 0.9180
Epoch 7/100
313/313 [==============================] - 38s 119ms/step - loss: 0.1
```

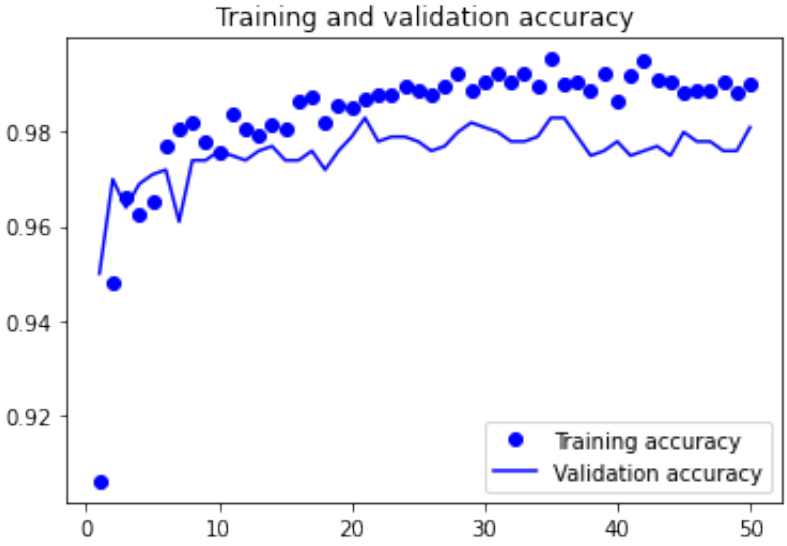


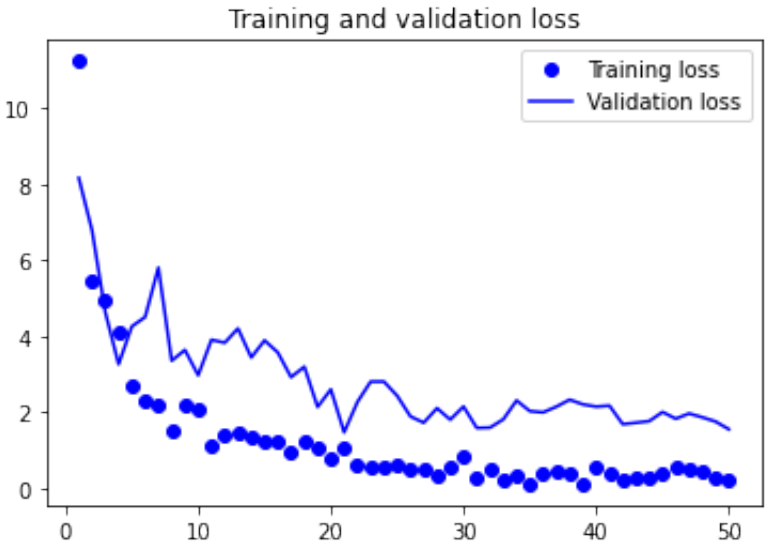



```
32/32 [==============================] - 2s 53ms/step - loss: 0.2109
- accuracy: 0.9270
Test accuracy: 0.927
```

4. Repeat Steps 1-3, but now using a pretrained network. The sample sizes you use in Steps 2 and 3 for the pretrained network may be the same or different from those using the network where you trained from scratch. Again, use any and all optimization techniques to get best performance.

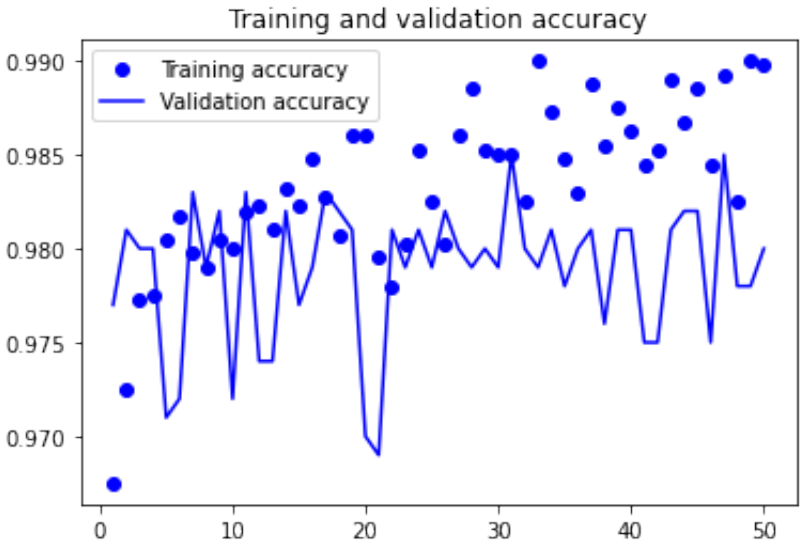
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
(https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5)
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step



Epoch 1/50
63/63 [==============================] - 35s 475ms/step - loss: 11.23
14 - accuracy: 0.9060 - val_loss: 8.1595 - val_accuracy: 0.9500
Epoch 2/50
63/63 [==============================] - 27s 424ms/step - loss: 5.435
5 - accuracy: 0.9480 - val_loss: 6.7817 - val_accuracy: 0.9700
Epoch 3/50
63/63 [==============================] - 27s 424ms/step - loss: 4.949
7 - accuracy: 0.9660 - val_loss: 4.6331 - val_accuracy: 0.9640
Epoch 4/50
63/63 [==============================] - 27s 423ms/step - loss: 4.088
8 - accuracy: 0.9625 - val_loss: 3.2680 - val_accuracy: 0.9690
Epoch 5/50
63/63 [==============================] - 27s 420ms/step - loss: 2.699
1 - accuracy: 0.9655 - val_loss: 4.2602 - val_accuracy: 0.9710
Epoch 6/50
63/63 [==============================] - 27s 419ms/step - loss: 2.322
2 - accuracy: 0.9770 - val_loss: 4.5067 - val_accuracy: 0.9720
Epoch 7/50
```

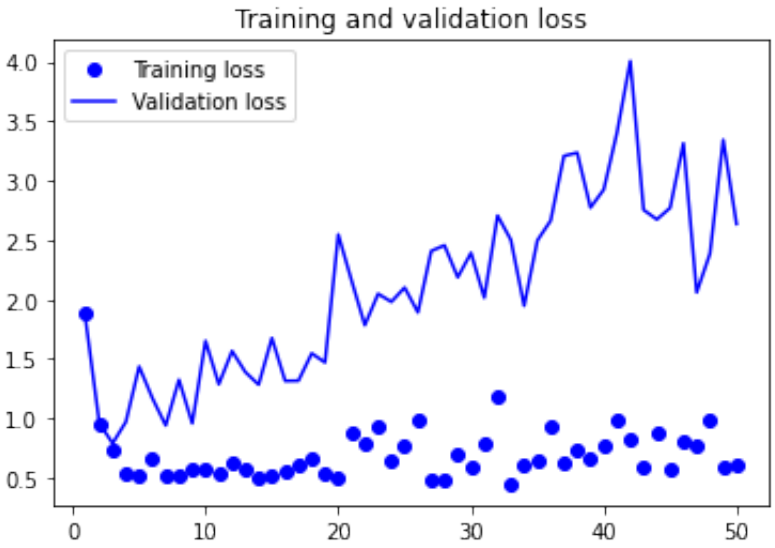





```
32/32 [==============================] - 9s 268ms/step - loss: 2.2952
- accuracy: 0.9750
Test accuracy: 0.975
```

```
Epoch 1/50
125/125 [==============================] - 44s 353ms/step - loss: 1.8
922 - accuracy: 0.9675 - val_loss: 1.8032 - val_accuracy: 0.9770
Epoch 2/50
125/125 [==============================] - 45s 354ms/step - loss: 0.9
455 - accuracy: 0.9725 - val_loss: 0.9609 - val_accuracy: 0.9810
Epoch 3/50
125/125 [==============================] - 45s 354ms/step - loss: 0.7
349 - accuracy: 0.9772 - val_loss: 0.7904 - val_accuracy: 0.9800
Epoch 4/50
125/125 [==============================] - 44s 352ms/step - loss: 0.5
434 - accuracy: 0.9775 - val_loss: 0.9716 - val_accuracy: 0.9800
Epoch 5/50
125/125 [==============================] - 44s 351ms/step - loss: 0.5
102 - accuracy: 0.9805 - val_loss: 1.4368 - val_accuracy: 0.9710
Epoch 6/50
125/125 [==============================] - 44s 351ms/step - loss: 0.6
626 - accuracy: 0.9818 - val_loss: 1.1701 - val_accuracy: 0.9720
Epoch 7/50
125/125 [==============================] - 44s 351ms/step - loss: 0.5
```





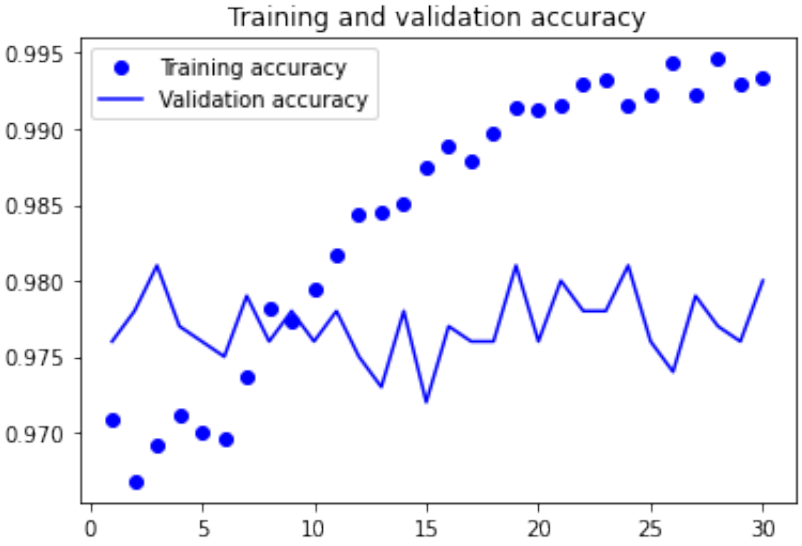

```
32/32 [==============================] - 9s 268ms/step - loss: 1.0348
- accuracy: 0.9660
Test accuracy: 0.966
```

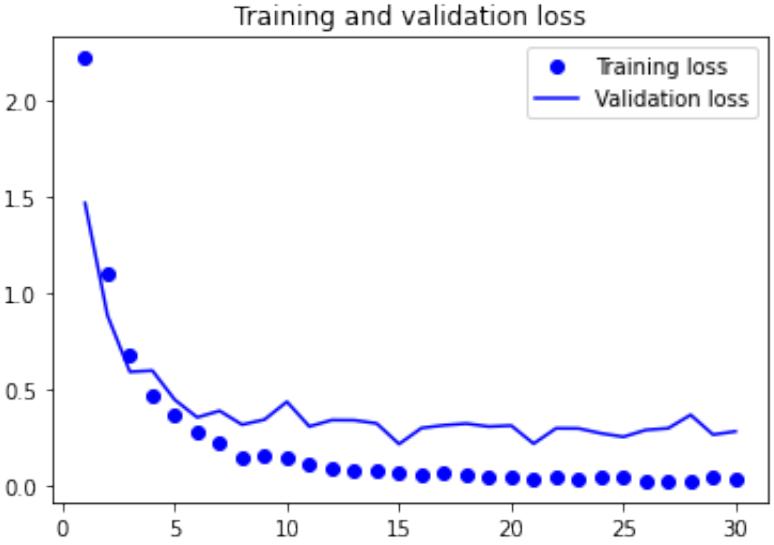
Finetuning the last three convolutional layers by freezing all layers upto the fourth layer, and layers 5 and above should be trainable

```
Epoch 1/30
313/313 [==============================] - 113s 351ms/step - loss: 2.
2209 - accuracy: 0.9708 - val_loss: 1.4687 - val_accuracy: 0.9760
Epoch 2/30
313/313 [==============================] - 110s 350ms/step - loss: 1.
1065 - accuracy: 0.9668 - val_loss: 0.8859 - val_accuracy: 0.9780
```

```
Epoch 3/30
313/313 [==============================] - 110s 350ms/step - loss: 0.
6803 - accuracy: 0.9692 - val_loss: 0.5938 - val_accuracy: 0.9810
Epoch 4/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
4702 - accuracy: 0.9711 - val_loss: 0.6003 - val_accuracy: 0.9770
Epoch 5/30
313/313 [==============================] - 110s 350ms/step - loss: 0.
3741 - accuracy: 0.9700 - val_loss: 0.4491 - val_accuracy: 0.9760
Epoch 6/30
313/313 [==============================] - 110s 350ms/step - loss: 0.
2768 - accuracy: 0.9696 - val_loss: 0.3577 - val_accuracy: 0.9750
Epoch 7/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
2261 - accuracy: 0.9737 - val_loss: 0.3916 - val_accuracy: 0.9790
Epoch 8/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
1476 - accuracy: 0.9782 - val_loss: 0.3200 - val_accuracy: 0.9760
Epoch 9/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
1632 - accuracy: 0.9774 - val_loss: 0.3462 - val_accuracy: 0.9780
Epoch 10/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
1500 - accuracy: 0.9794 - val_loss: 0.4389 - val_accuracy: 0.9760
Epoch 11/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
1153 - accuracy: 0.9817 - val_loss: 0.3105 - val_accuracy: 0.9780
Epoch 12/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
0905 - accuracy: 0.9843 - val_loss: 0.3437 - val_accuracy: 0.9750
Epoch 13/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0853 - accuracy: 0.9845 - val_loss: 0.3429 - val_accuracy: 0.9730
Epoch 14/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
0837 - accuracy: 0.9851 - val_loss: 0.3265 - val_accuracy: 0.9780
Epoch 15/30
313/313 [==============================] - 110s 350ms/step - loss: 0.
0716 - accuracy: 0.9875 - val_loss: 0.2195 - val_accuracy: 0.9720
Epoch 16/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
0557 - accuracy: 0.9889 - val_loss: 0.3024 - val_accuracy: 0.9770
Epoch 17/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0669 - accuracy: 0.9879 - val_loss: 0.3168 - val_accuracy: 0.9760
Epoch 18/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0560 - accuracy: 0.9897 - val_loss: 0.3254 - val_accuracy: 0.9760
Epoch 19/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
```

```
0499 - accuracy: 0.9914 - val_loss: 0.3101 - val_accuracy: 0.9810
Epoch 20/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
0447 - accuracy: 0.9912 - val_loss: 0.3152 - val_accuracy: 0.9760
Epoch 21/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0408 - accuracy: 0.9916 - val_loss: 0.2214 - val_accuracy: 0.9800
Epoch 22/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
0489 - accuracy: 0.9929 - val_loss: 0.3010 - val_accuracy: 0.9780
Epoch 23/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0339 - accuracy: 0.9932 - val_loss: 0.3003 - val_accuracy: 0.9780
Epoch 24/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0505 - accuracy: 0.9915 - val_loss: 0.2750 - val_accuracy: 0.9810
Epoch 25/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0513 - accuracy: 0.9922 - val_loss: 0.2569 - val_accuracy: 0.9760
Epoch 26/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
0276 - accuracy: 0.9944 - val_loss: 0.2923 - val_accuracy: 0.9740
Epoch 27/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0320 - accuracy: 0.9923 - val_loss: 0.3011 - val_accuracy: 0.9790
Epoch 28/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0287 - accuracy: 0.9946 - val_loss: 0.3707 - val_accuracy: 0.9770
Epoch 29/30
313/313 [==============================] - 109s 349ms/step - loss: 0.
0456 - accuracy: 0.9930 - val_loss: 0.2682 - val_accuracy: 0.9760
Epoch 30/30
313/313 [==============================] - 110s 349ms/step - loss: 0.
0419 - accuracy: 0.9934 - val_loss: 0.2849 - val_accuracy: 0.9800
```






```
32/32 [==============================] - 9s 268ms/step - loss: 0.3845
- accuracy: 0.9820
Test accuracy: 0.982
```