CONVOLUTION SUMMARY

Bhumika Shah

811286091

This dataset is selected from the larger set of 25,000 pictures, with an equal number of pictures for cats and dogs. For this experiment, we chose a subset of 12,000 images to use for training, 1,000 for validation, and 1,000 for testing. That would provide 500 of each class for the three sets of data so both cats and dogs would be equally represented in each set.

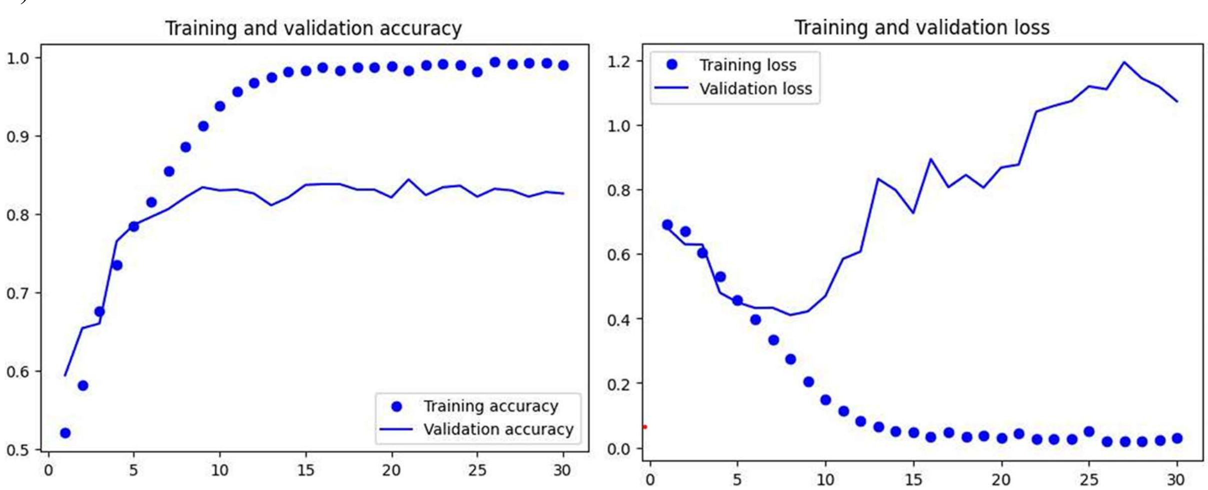
The model has been developed in such a way as to classify these images. Therefore, it takes input images of 180x180 pixels with 3 color channels (RGB). It represents the essential type of model that consists of several Convolutional 2D or Conv2D layers, followed by Max Pooling layers, which help capture crucial patterns within the images. Further, behind these layers, comes a Flatten layer, which transforms the data into a 1D format, followed by Dense layers that combine the features. The output layer is implemented using the sigmoid activation function, which is optimal for binary tasks, such as distinguishing cats from dogs.

Before feeding the images to the model, they are first resized uniformly to 180x180 pixels. A Rescaling layer also normalizes the pixel values; it changes the standard range of 0-255 into a range of 0-1. This improves the model's training and performance.

I've walked through the graph and the findings in the code below.

1)

```
Epoch 30/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 19s 51ms/step - accuracy: 0.9925 - loss: 0.0255 - val_accuracy: 0.8260 - val_loss: 1.0730
```

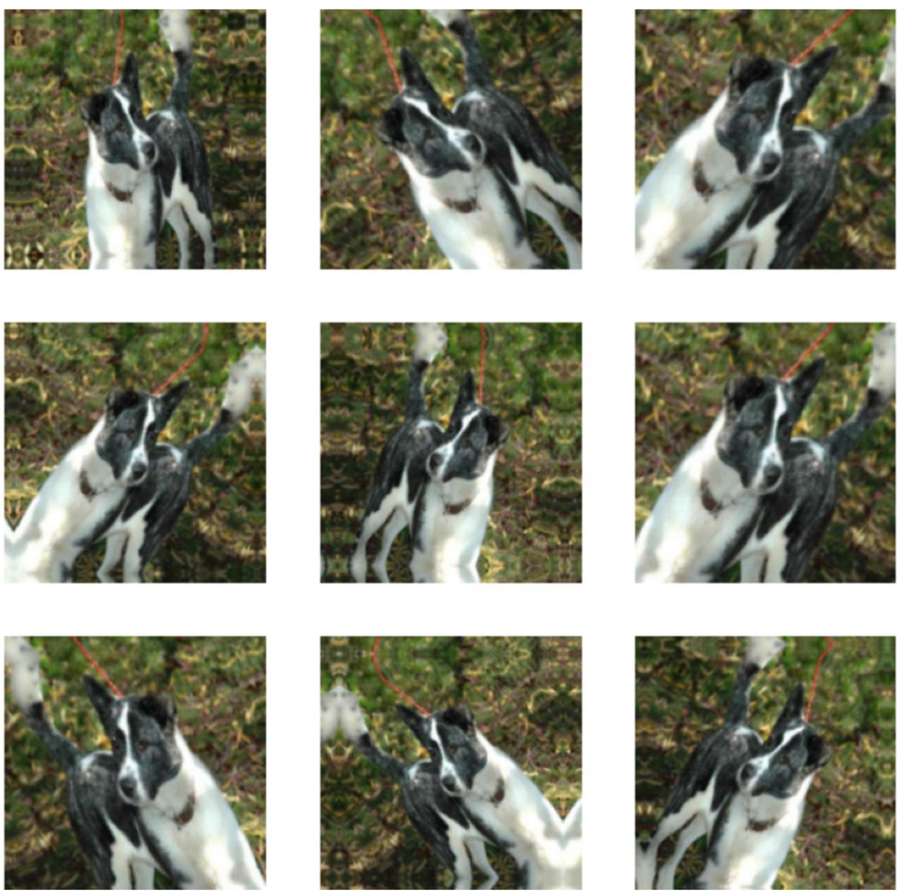



```
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 113ms/step - accuracy: 0.7890 - loss: 0.4568
Test accuracy: 0.792
```

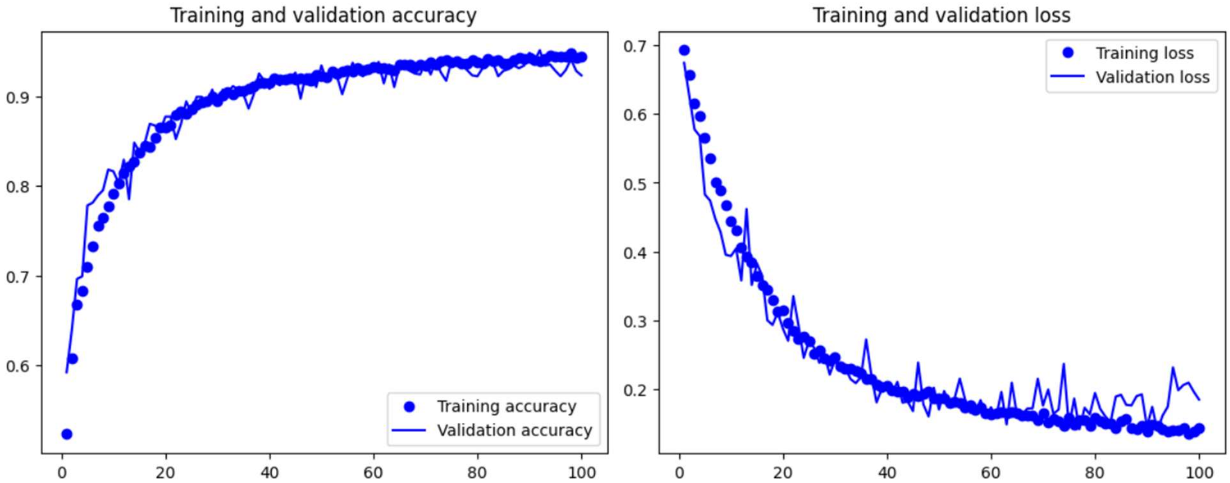
It was trained for 30 epochs with the Adam optimizer and the binary cross-entropy loss. While it predicts with an accuracy of 99.25% on the training data, its performance on test data is 79.2%, which shows that model overfitting has taken place. The validation accuracy reached a maximum of 84.40% and steadily decreased after 10 epochs, even though the training accuracy was still on the rise, further indicating that this network has overfitted to the training data.

The graphs show that the model's training accuracy keeps improving, but validation accuracy levels off and starts to drop after an initial improvement. While training loss decreases, validation loss begins to rise, indicating overfitting. This suggests the model is learning the training data well but struggling to generalize to new data. To fix this, techniques like data augmentation, regularization, or transfer learning could help.

2) Displaying the enhanced images that were trained



```
Epoch 100/100
375/375 ━━━━━━━━━━━━━━━━━━━━ 20s 52ms/step - accuracy: 0.9375 - loss: 0.1504 - val_accuracy: 0.9230 - val_loss: 0.1849
```

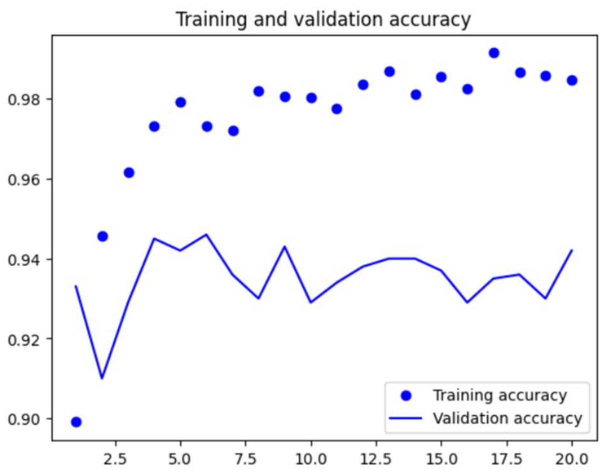


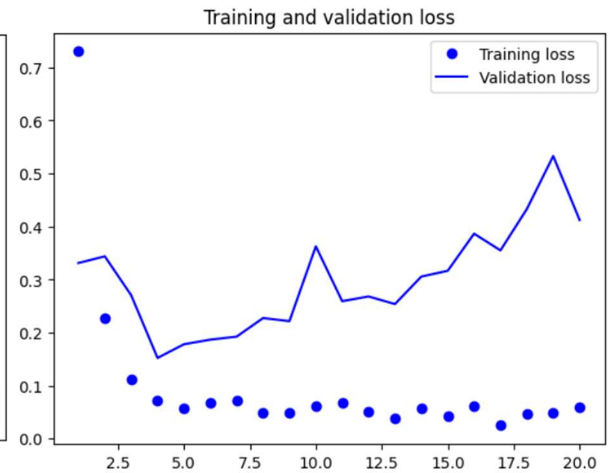

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.9179 - loss: 0.1816
Test accuracy: 0.924
```

I have used data augmentation in Keras, including horizontal flipping, slight rotation, and random zooming. The architecture of the model was similar as before, but with data augmentation, I added a 50 percent dropout layer before the final dense layer. After that, the model was trained for 100 epochs using the Adam optimizer and binary cross-entropy loss. ModelCheckpoint was used to save the best model based on validation loss. It has a training accuracy of 93.75%, with its validation accuracy peaking at 95.10%, and a test accuracy radically improved to 92.4%. Data augmentation helped reduce overfitting and further generalized the model compared to the old version.

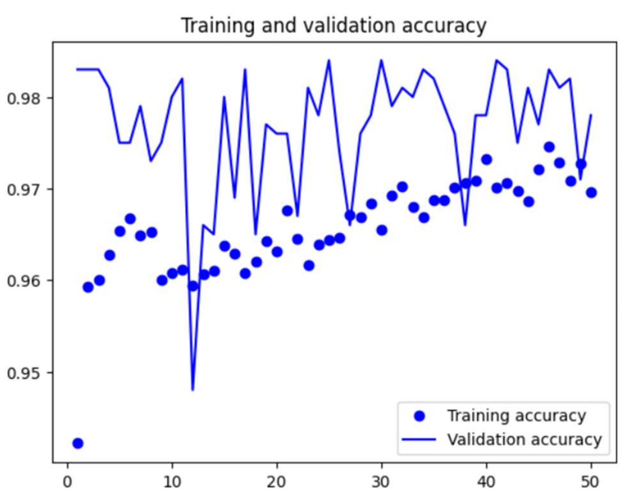
Losses and accuracies of training and validation are much closer to each other and more stable after data augmentation. Test accuracy has risen significantly from 79.2% to 92.4%, showing better generalization and reduced overfitting compared with the non-augmented model. This indicates that by performing data augmentation, the model was able to learn more robust features, which then generalized better on unseen data.

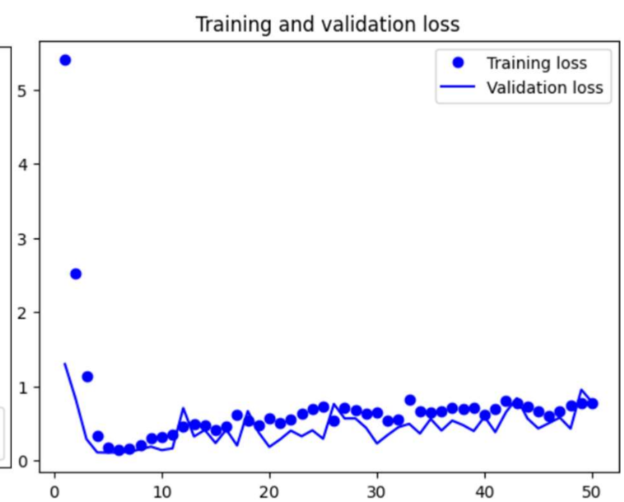
3)






```
Epoch 50/50
375/375 ━━━━━━━━━━━━━━━━━━━━ 41s 76ms/step - accuracy: 0.9676 - loss: 0.8506 - val_accuracy:
0.9780 - val_loss: 0.7834
```






```
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 54ms/step - accuracy: 0.9743 - loss: 0.0806
Test accuracy: 0.969
```

This first model, with a sample size of 12,000 for training and having run for 20 epochs, has a final validation accuracy of 94.2% and a validation loss of 0.4125, whereas the improved model, with data augmentation and training for 50 epochs, reached higher values for validation accuracy of 97.8% and test accuracy of 96.9%, with slight overfitting for both models. Hence, the data augmentation model generalized better with longer training and higher validation accuracy. This warrants further experiments to try out different sample sizes, such as 6,000, 9,000, 15,000, and 18,000 using the same architecture of data augmentation, and compare their respective validation and test accuracies to find out the optimal performance. These will help reach conclusions on whether a smaller sample size may work similarly to or even better.

4)

```
Epoch 30/30
375/375 ━━━━━━━━━━━━━━━━━━━━ 29s 76ms/step - accuracy: 0.9746 - loss: 0.8869 - val_accuracy: 0.9830 - val_loss: 0.6722
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 64ms/step - accuracy: 0.9831 - loss: 0.4981
Test accuracy: 0.978
```

The model was fine-tuned by unfreezing the last four layers of the InceptionV3 base model and trained for 30 epochs using the Adam optimizer with a learning rate of 0.001. The highest validation accuracy reached 98.5% at epoch 11, with a final accuracy of 98.3%. Throughout training, the model maintained a high training accuracy above 97%, and the test accuracy was 97.8%. While there were minor fluctuations in validation loss, the model showed good stability in validation accuracy, with strong generalization as the test accuracy closely matched the validation accuracy.

The fine-tuned model achieved a higher test accuracy of 97.8%, compared to the previous best result of 96.9% using data augmentation. This improvement indicates that fine-tuning the last few layers of the base model was beneficial. The training process used 375 steps per epoch, likely with the full dataset of 12,000 samples, consistent with previous experiments. Overall, fine-tuning with a sample size of 12,000 has produced the best results, showing that the performance gains came from the fine-tuning strategy rather than changes in the sample size.