Assignment 2: Deep Convolutional Neural Network - Cats vs Dogs

Name : Akhila Kalpuri
Date : 03-23-2024

Purpose

The purpose of this assignment is to explore the application of convolutional neural networks (convnets) to image data and understand the relationship between sample sizes and the choice of training convnets from scratch versus using a pretrained network. The assignment aligns with Module Learning Outcomes (MLO) 1 and 2, focusing on applying convnets to image data and explaining the impact of sample sizes on training methodologies.

Getting Started

1. Applying Convnets to Image Data: Utilize convnets to classify images, focusing on the Cats & Dogs example.
2. Exploring Sample Sizes: Investigate the effect of sample sizes on training methodologies, comparing training from scratch versus using a pretrained convnet.

Summary of Findings

Four models were constructed with a training sample size of 1000.
Model 2, with augmented images and a dropout rate of 0.5, achieved 71% accuracy.
Increasing the training sample to 2000 improved Model 2's accuracy to 81%.

Three models were built with a training sample size of 2000.
Model 5, with padding being the same, showed improved performance compared to Model 2.

Models were trained with sample sizes ranging from 3000 to 5000.
The best-performing model achieved 88.4% accuracy with a training sample size of 5000.

Pretrained models were evaluated, with Model 10 achieving 88.4% accuracy with 5000 training samples.
Fine-tuned VGG models outperformed scratch models, with 100% accuracy in some cases.

Conclusion

The relationship between training sample size and model performance is evident.
Increasing the sample size improves accuracy, especially for scratch models.
Pretrained models offer high accuracy but may not always require extensive training data.
Recommendations include leveraging pretrained models for high accuracy and considering sample size based

on the desired performance level.

Recommendations

Utilize pretrained models for tasks requiring high accuracy without extensive training data.
Consider increasing sample sizes for scratch models to improve performance.
Evaluate model performance based on specific task requirements and available data.

Below is the step-by-step process of how I did my code and the graphs and summary and also my opinion:

Step1: Loading all the required libraries and functions
Importing the JSON Activation Code
Saving kaggle.json to kaggle (2).json
Downloading the Data
Unzipping the Data
Unzipping Train Data

Consider the Cats & Dogs example. Start initially with a training sample of 1000, a validation sample of 500, and a test sample of 500 (like in the text). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?

Creating directory named cats vs dogs small to store the images into 3 subsets named train, validation and test and Dividing the training sample of 1000, a validation sample of 500, and a test sample of 5

Data Pre-Processing and functions - Using image_dataset_from_directory to read images and functions

Found 1000 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 500 files belonging to 2 classes.

Viewing the shape of the images

data batch shape: (32, 180, 180, 3) labels batch shape: (32,)

Model - 1 MaxPooling Operation with Increase in filters from 32 to 256 in 5 Input Layers : Instan- tiating a small convnet for dogs vs. cats classification

Summary of Model - 1

Model: "model_1"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 180, 180, 3)]	0
rescaling_1 (Rescaling)	(None, 180, 180, 3)	0
conv2d_5 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_4 (MaxPooling 2D)	(None, 89, 89, 32)	0
conv2d_6 (Conv2D)	(None, 87, 87, 64)	18496
max_pooling2d_5 (MaxPooling 2D)	(None, 43, 43, 64)	0
conv2d_7 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_6 (MaxPooling 2D)	(None, 20, 20, 128)	0
conv2d_8 (Conv2D)	(None, 18, 18, 256)	295168
max_pooling2d_7 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_9 (Conv2D)	(None, 7, 7, 256)	590080
flatten_1 (Flatten)	(None, 12544)	0
dense_1 (Dense)	(None, 1)	12545

==========

Total params: 991,041

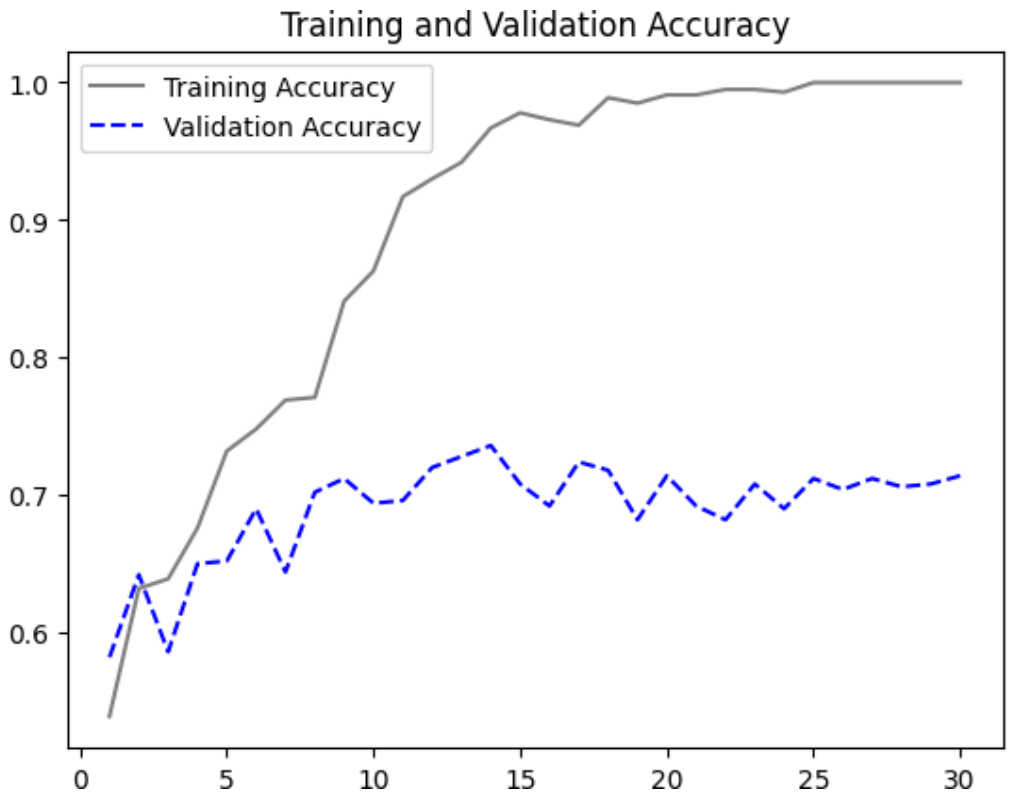
Trainable params: 991,041

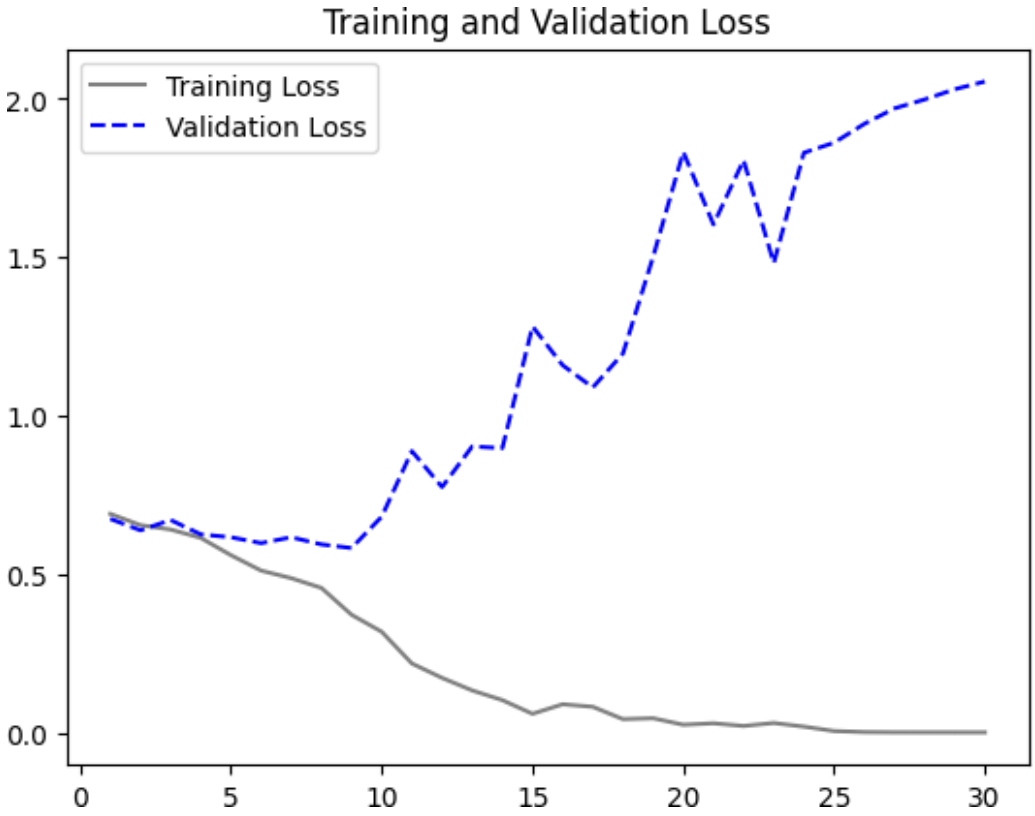
Non-trainable params: 0

_____

Training the model 1

Looking at the visuals of the Training and Validation Accuracy/Loss





Evaluating the performance of Model_1 on test set

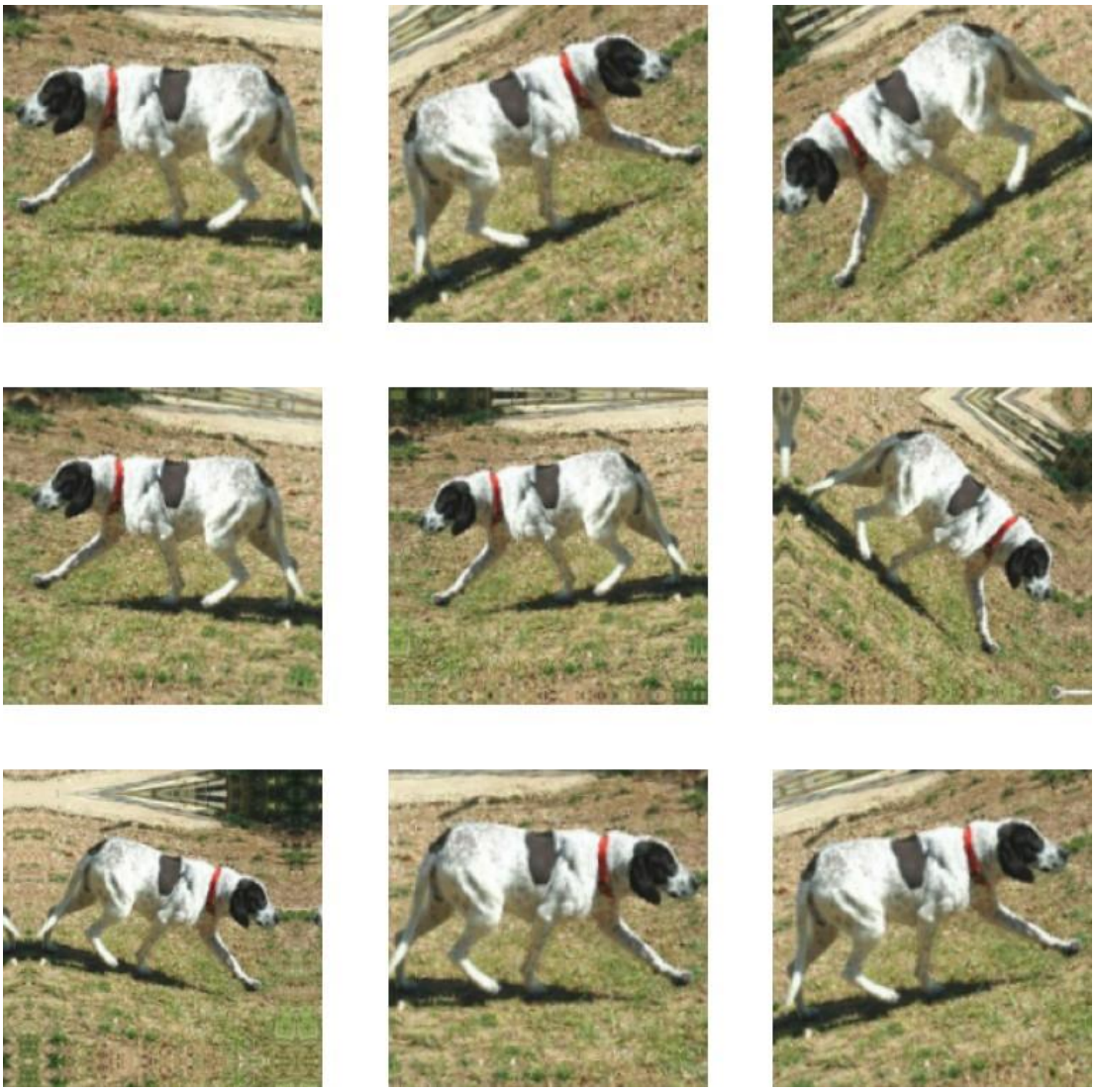
Loss: 0.653

Accuracy: 0.696

Using Measures to Avoid OverfittingData Augmentation

Using few of the techniques such as random flip, random zoom, random rotation so as to create augmented versions of the image.

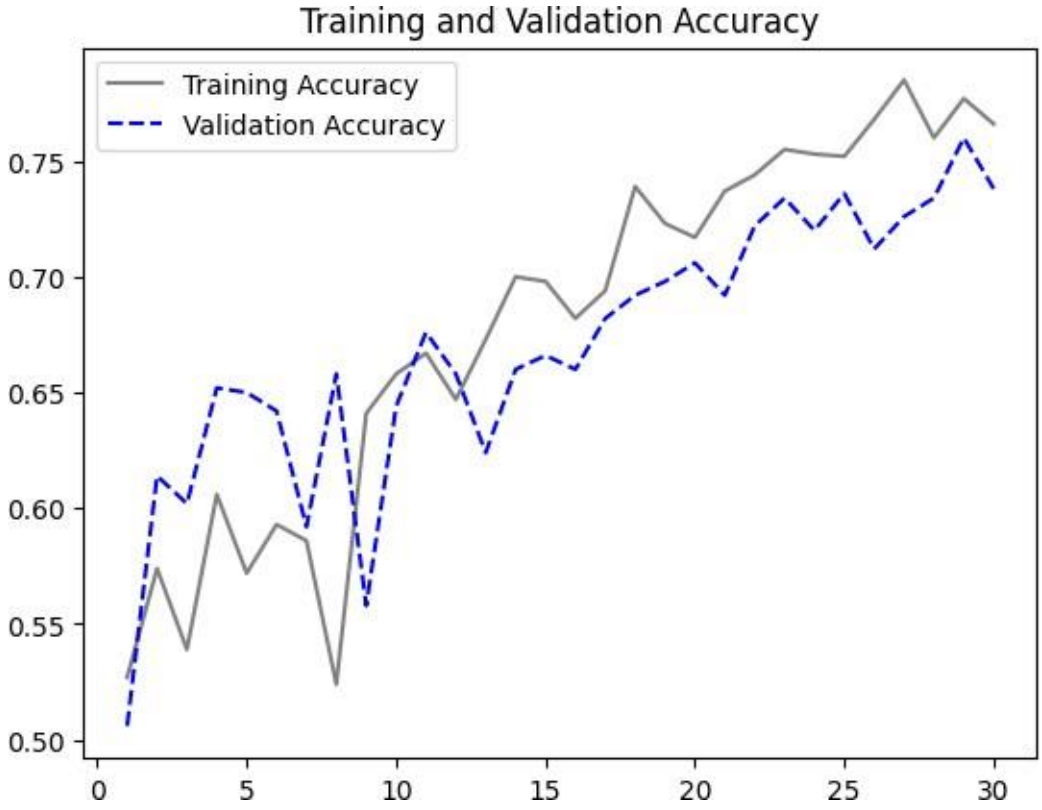
Looking at the augmented images

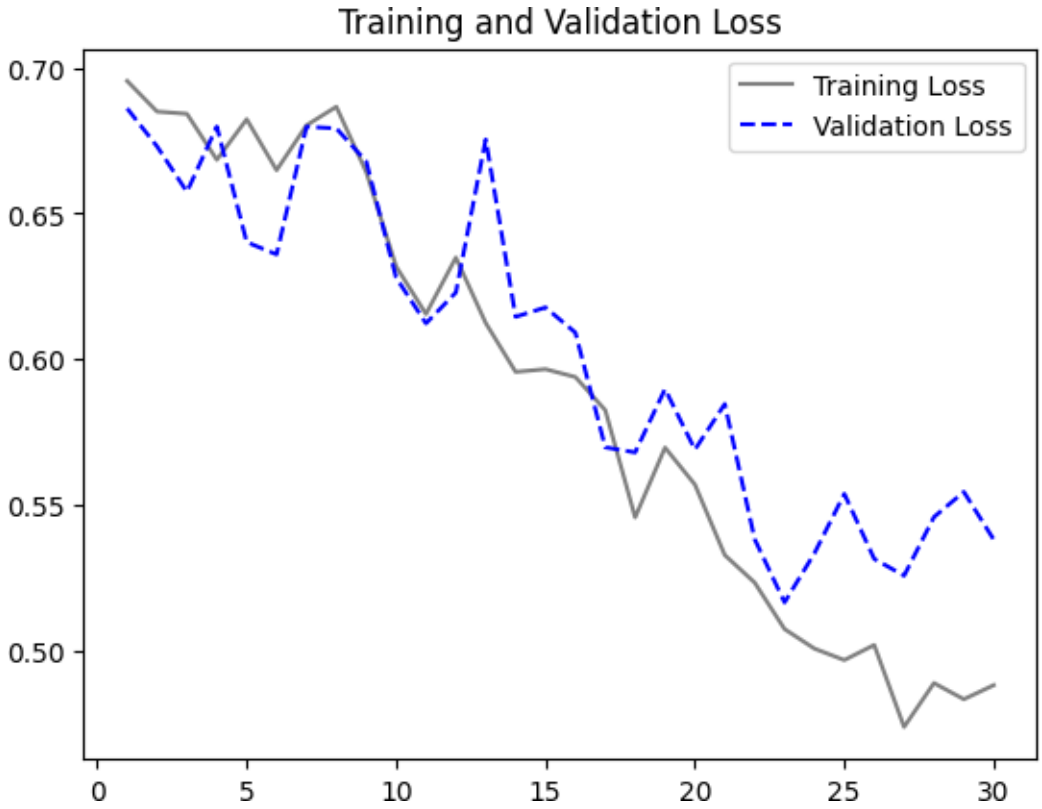


Model - 2 MaxPooling Operation with Increase in filters from 32 to 256 in 5 Input Layers with thedata being used from the Augmented Images and a droput rate of 0.5*

Training the model 2

Visualizing the Training and Validation Accuracy/Loss 2


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30


Training and Validation Loss
Training Loss
Validation Loss
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30

Evaluating the performance of Model_2 on the test set

Loss: 0.588
Accuracy: 0.714

Comparing the Model 1 and Model 2 : we can clearly see that the accuracy rate of model 2 ishigher than model 1

Model - 3 MaxPooling Operation with Increase in filters from 32 to 512 in 6 Input Layers with theuse of Augmented Images and Dropout rate of 0.5

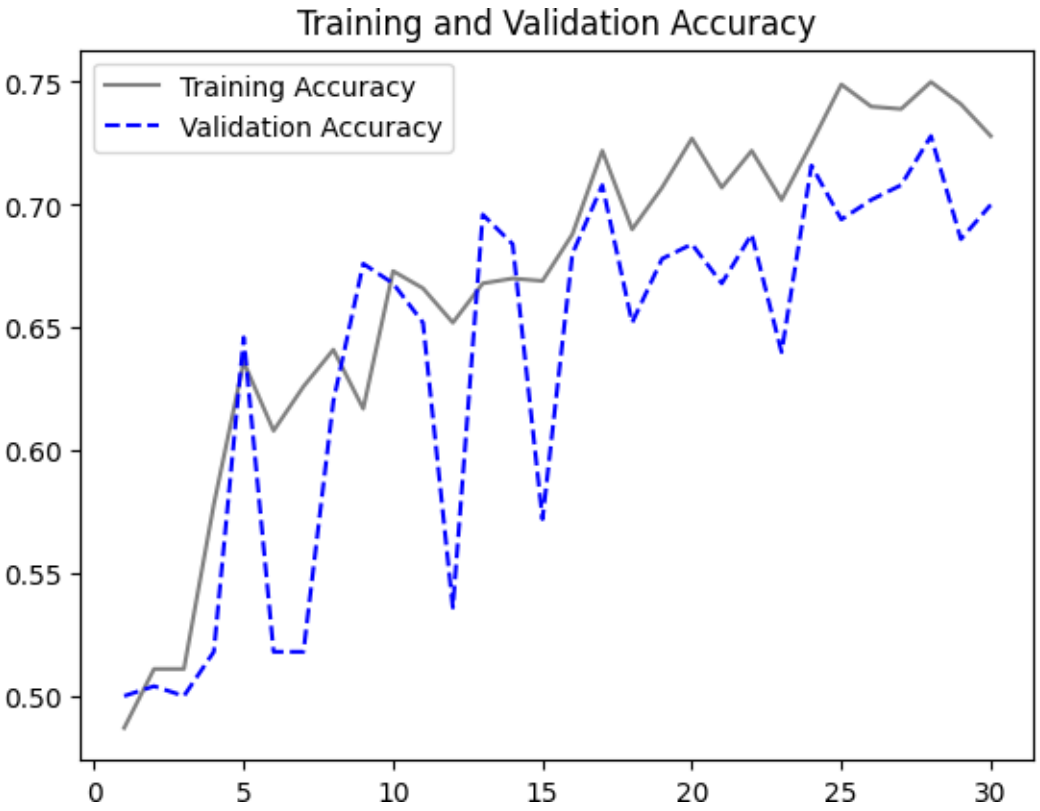
Model: "model_3"

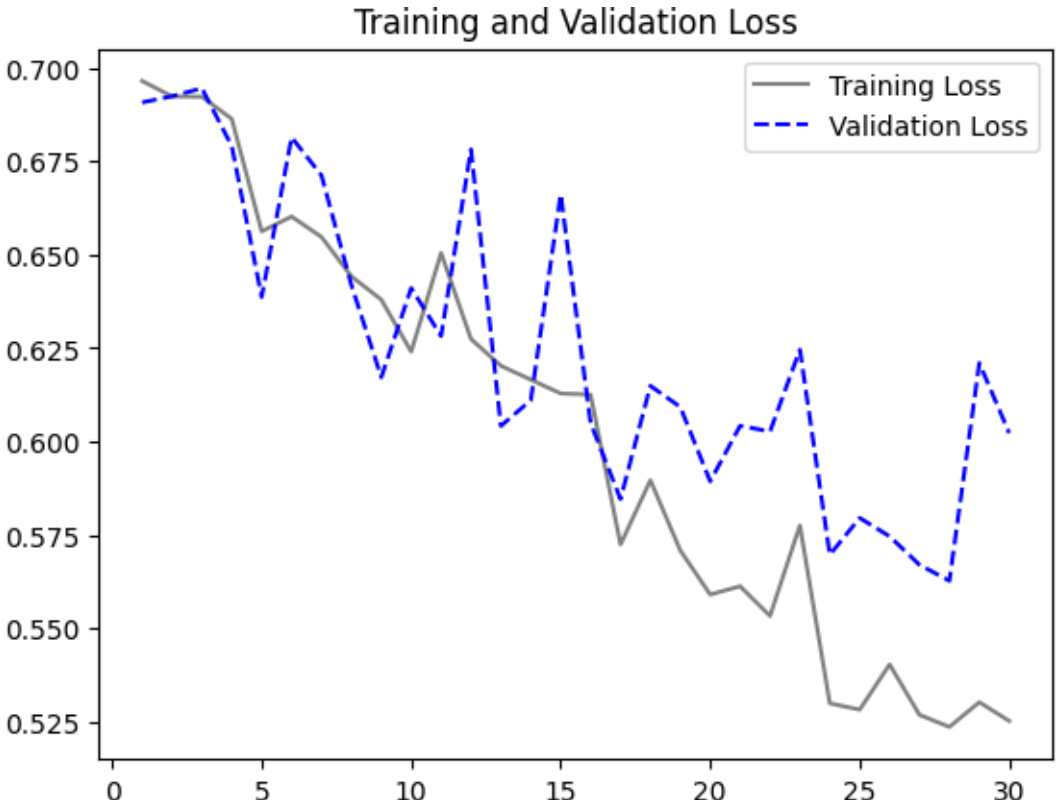
Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[(None, 180, 180, 3)]	0
sequential_1 (Sequential)	(None, 180, 180, 3)	0
rescaling_4 (Rescaling)	(None, 180, 180, 3)	0
conv2d_21 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_17 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_22 (Conv2D)	(None, 87, 87, 64)	18496
max_pooling2d_18 (MaxPooling2D)	(None, 43, 43, 64)	0
conv2d_23 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_19 (MaxPooling2D)	(None, 20, 20, 128)	0
conv2d_24 (Conv2D)	(None, 18, 18, 256)	295168
max_pooling2d_20 (MaxPooling2D)	(None, 9, 9, 256)	0
conv2d_25 (Conv2D)	(None, 7, 7, 256)	590080
max_pooling2d_21 (MaxPooling2D)	(None, 3, 3, 256)	0
conv2d_26 (Conv2D)	(None, 1, 1, 512)	1180160
flatten_4 (Flatten)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 1)	513

Total params: 2,159,169
Trainable params: 2,159,169
Non-trainable params: 0

Training the model 3

Visualizing the Training and Validation Accuracy/Loss 2





Evaluating the performance of Model_3 on the test set
Loss: 0.620
Accuracy: 0.674

Model - 4 MaxPooling Operation with Increase in filters from 64 to 1024 in 5 Input Layers withthe use of Augmented Images and Dropout rate of 0.6

Model's summary

Model: "model_4"

Layer (type)	Output Shape	Param #
input_6 (InputLayer)	[(None, 180, 180, 3)]	0
sequential_1 (Sequential)	(None, 180, 180, 3)	0
rescaling_5 (Rescaling)	(None, 180, 180, 3)	0
conv2d_27 (Conv2D)	(None, 178, 178, 64)	1792
max_pooling2d_22 (MaxPooling2D)	(None, 89, 89, 64)	0

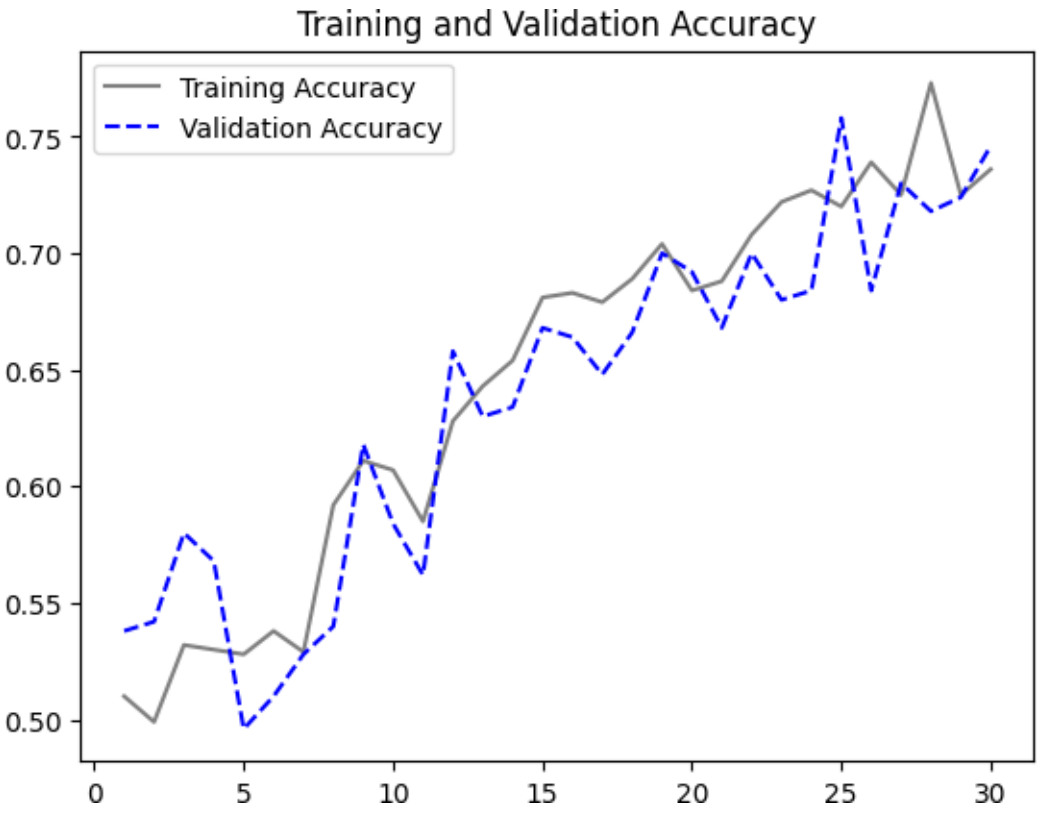
conv2d_28 (Conv2D) (None, 87, 87, 128) 73856
max_pooling2d_23 (MaxPoolin g2D) (None, 43, 43, 128) 0
conv2d_29 (Conv2D) (None, 41, 41, 256) 295168
max_pooling2d_24 (MaxPoolin g2D) (None, 20, 20, 256) 0
conv2d_30 (Conv2D) (None, 18, 18, 512) 1180160
max_pooling2d_25 (MaxPoolin g2D) (None, 9, 9, 512) 0
conv2d_31 (Conv2D) (None, 7, 7, 1024) 4719616
flatten_5 (Flatten) (None, 50176) 0

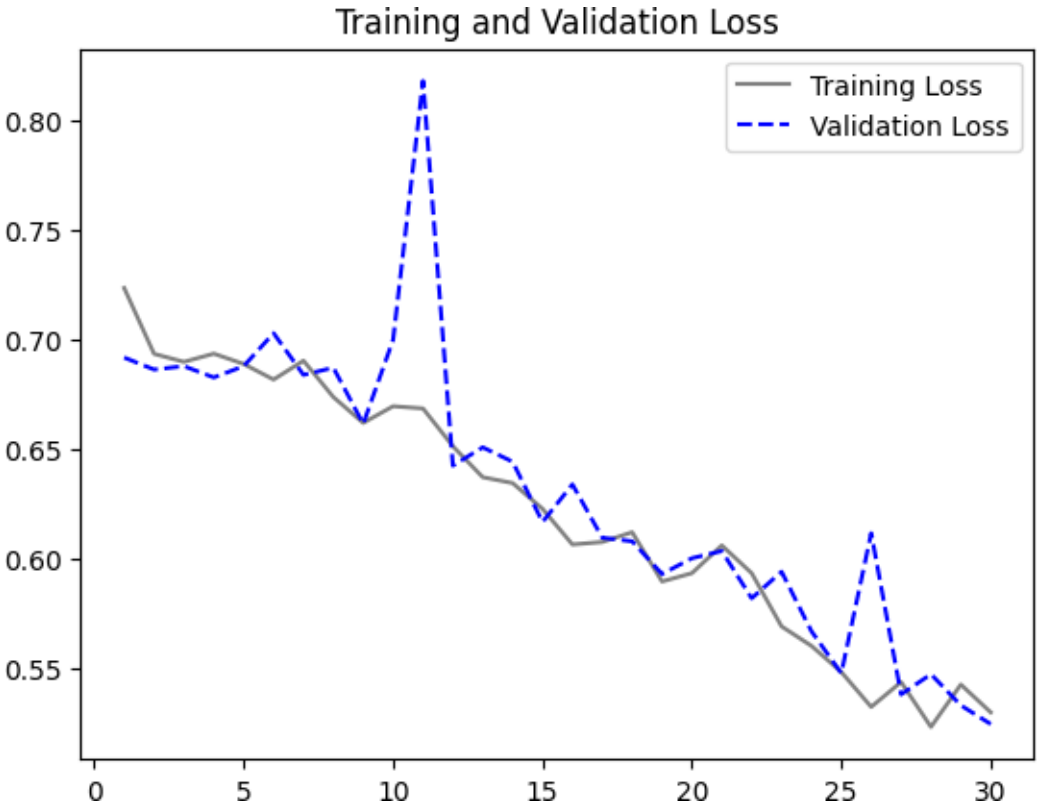
dropout_3 (Dropout) (None, 50176) 0
dense_5 (Dense) (None, 1) 50177

===
Total params: 6,320,769
Trainable params: 6,320,769
Non-trainable params: 0

Training the model 4

Visualizing the Training and Validation Accuracy/Loss 2

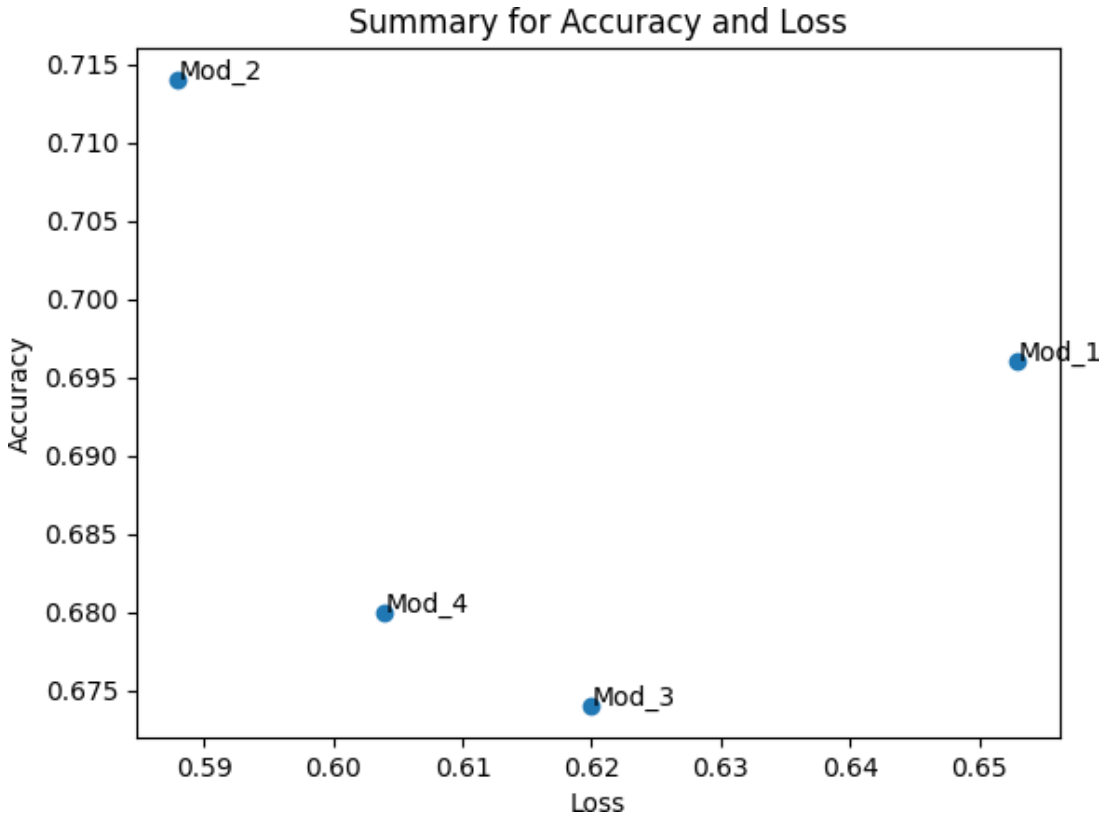




Evaluating the performance of Model_4 on the test set
Loss: 0.604
Accuracy: 0.680

Summary for Question 1:
We did attempt to construct four models using a thousand as the training sample. Let's now evaluate the accuracy and loss of all four models to determine which produces the best results.

Model 1: filters from 32 to 256, 5 Input Layers
Model 2: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5
Model 3: filters from 32 to 512, 6 Input Layers, Augmented Images and Dropout rate of 0.5
Model 4: filters from 64 to 1024, 5 Input Layers, Augmented Images and Dropout rate of 0.6



Conclusions

From the above graph we can conclude that model 2 is the best among all with higher accuracy and minimum loss, however model 4 has the highest loss

Recommendation

As we can see model 2 is performing best among all 4 models hence we should choose model with filters from 32 to 256, 5 Input Layers, Augmented Images and Dropout rate of 0.5

Model - 5 MaxPooling Operation with Increase in filters from 32 to 256 in 5 Input Layers with the data

being used from the Augmented Images and a droput rate of 0.5 (Training Sample - 2000)

Model Summary

Model: "model_5" Layer (type) Output Shape Param #

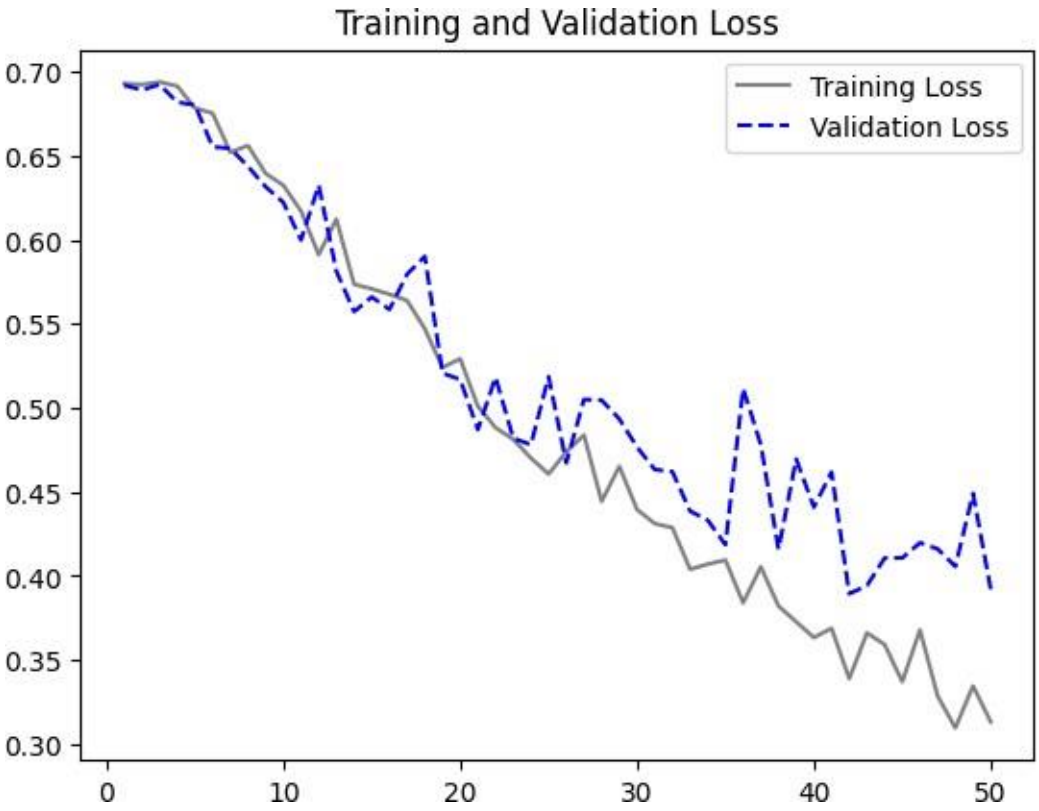
==========

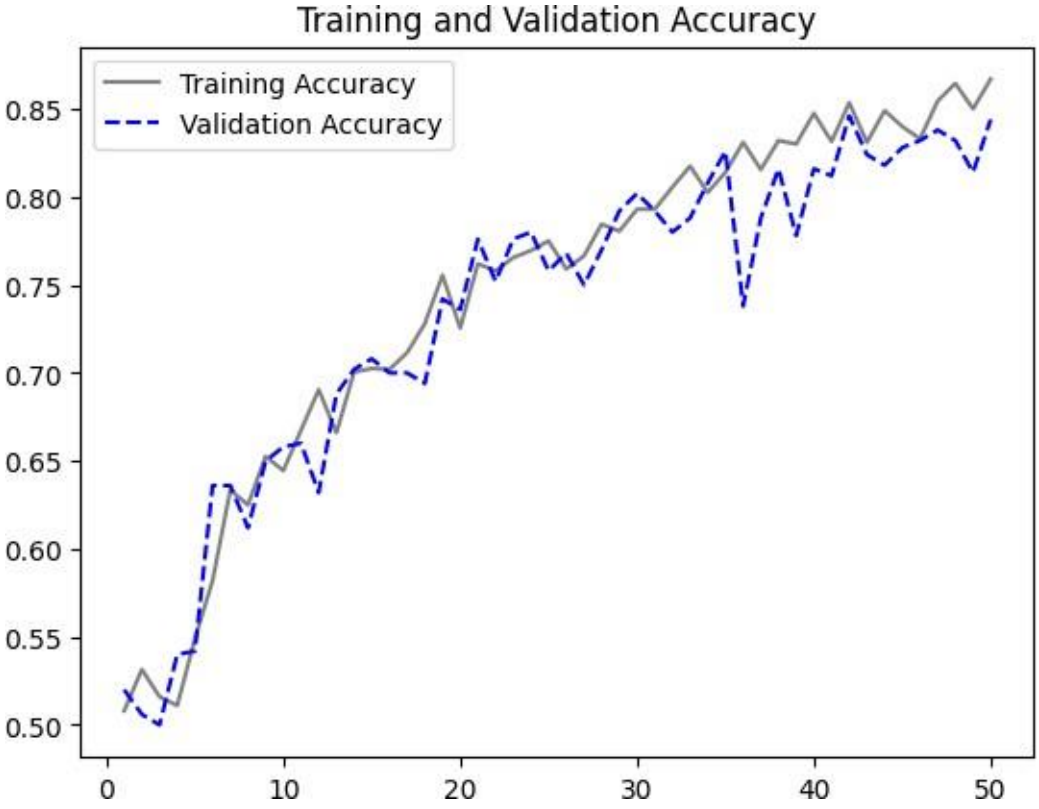
rescaling_6 (Rescaling) (None, 180, 180, 3) 0
conv2d_32 (Conv2D) (None, 178, 178, 32) 896
max_pooling2d_26 (MaxPoolin g2D) (None, 89, 89, 32) 0
conv2d_33 (Conv2D) (None, 87, 87, 64) 18496
max_pooling2d_27 (MaxPoolin g2D) (None, 43, 43, 64) 0
conv2d_34 (Conv2D) (None, 41, 41, 128) 73856
max_pooling2d_28 (MaxPoolin g2D) (None, 20, 20, 128) 0
conv2d_35 (Conv2D) (None, 18, 18, 256) 295168
max_pooling2d_29 (MaxPoolin g2D) (None, 9, 9, 256) 0
conv2d_36 (Conv2D) (None, 7, 7, 256) 590080
flatten_6 (Flatten) (None, 12544) 0
dropout_4 (Dropout) (None, 12544) 0
dense_6 (Dense) (None, 1) 12545

==========
Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0
_____

Training the model

Visualizing the Training and Validation Accuracy/Loss 2


Training and Validation Loss
Training Loss
Validation Loss
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
10
20
30
40
50


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
10
20
30
40
50

Evaluating the performance of Model_5 on test set
Loss: 0.459
Accuracy: 0.814

Summary : The accuracy of the model, which was only generated with 1000 training examples, was 71%, but the accuracy of the same model increased to 81%, or a 10% improvement, when training samples were increased to 2000.

Model - 6 Strides Operation with Padding being "Same" with Increase in filters from 32 to 256in 5 Input Layers with the data being used from the Augmented Images and a droput rate of 0.5 (Training Sample - 2000).

Model summary

Model: "model_6"

Layer (type)	Output Shape	Param #
input_9 (InputLayer)	[(None, 180, 180, 3)]	0
sequential_2 (Sequential)	(None, 180, 180, 3)	0
rescaling_8 (Rescaling)	(None, 180, 180, 3)	0
conv2d_42 (Conv2D)	(None, 90, 90, 32)	896
conv2d_43 (Conv2D)	(None, 45, 45, 64)	18496
conv2d_44 (Conv2D)	(None, 23, 23, 128)	73856
conv2d_45 (Conv2D)	(None, 12, 12, 256)	295168
conv2d_46 (Conv2D)	(None, 6, 6, 256)	590080
flatten_8 (Flatten)	(None, 9216)	0
dropout_6 (Dropout)	(None, 9216)	0

dense_8 (Dense) (None, 1) 9217

===

Total params: 987,713

Trainable params: 987,713

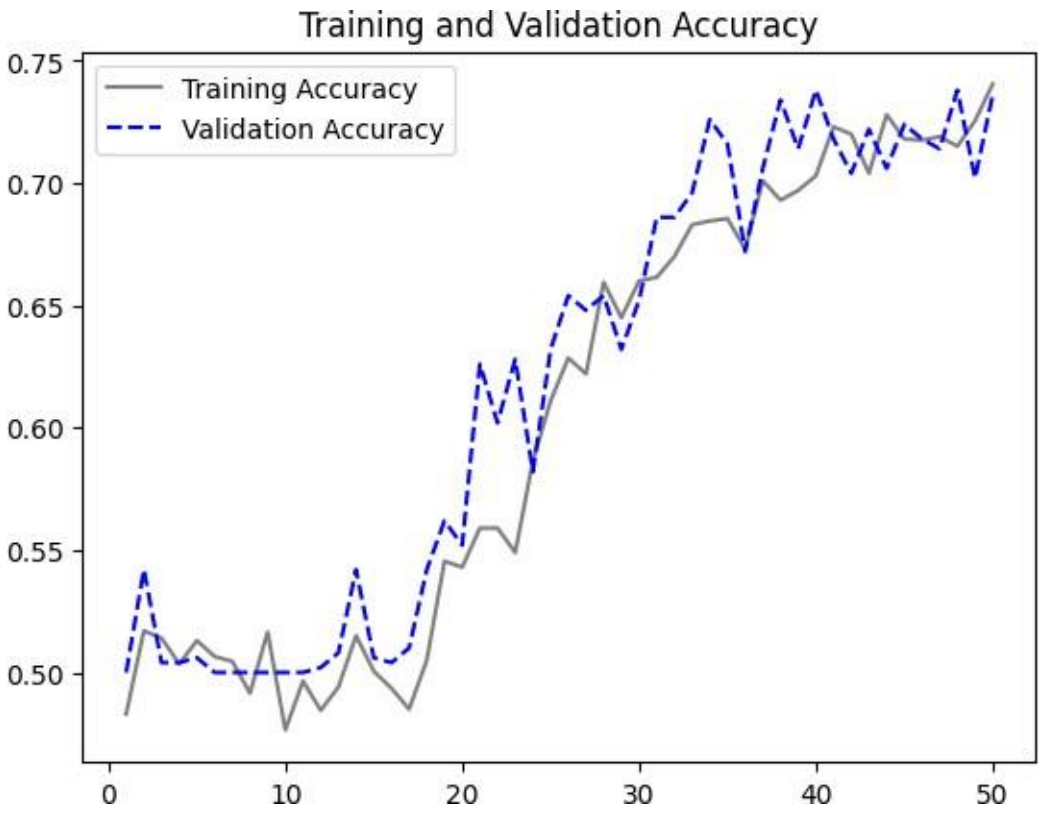
Non-trainab

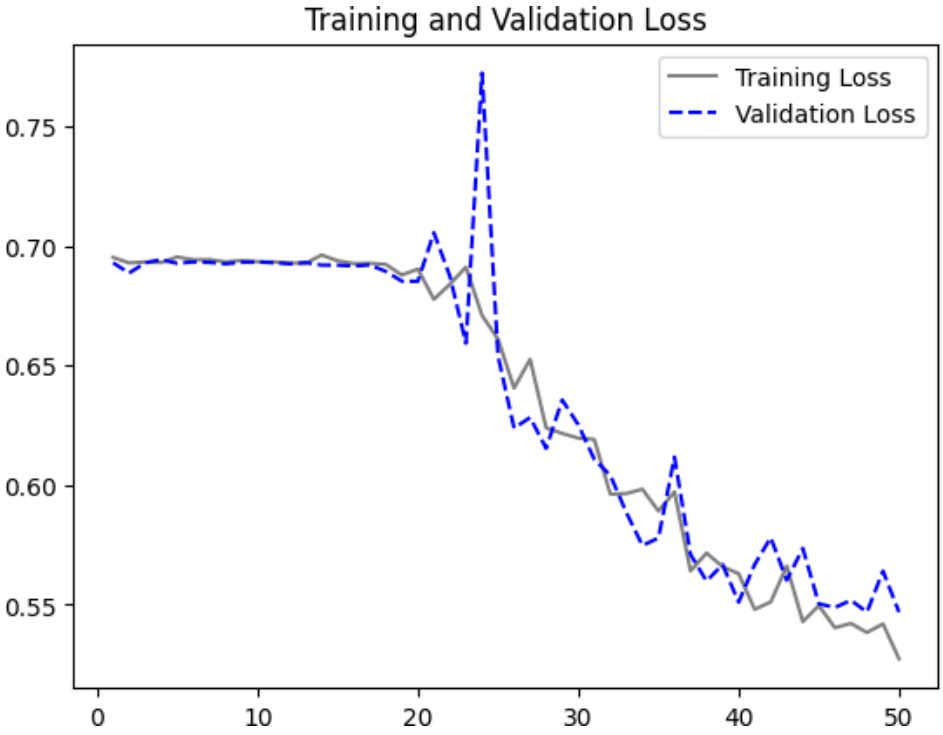
le params: 0

_____

Training the model 6

Visualizing the Training and Validation Accuracy/Loss





Loss: 0.602
Accuracy: 0.674

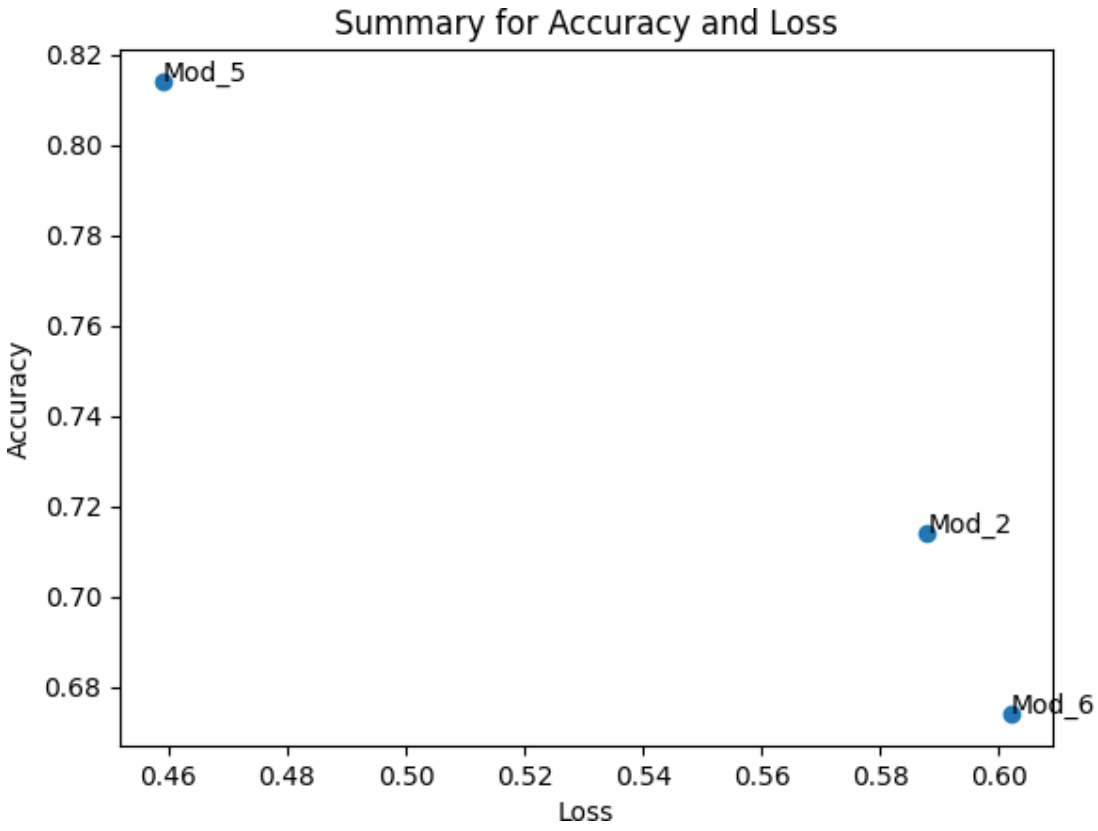
Summary for Question 2

We did try to build 2 more models with training sample being 2000.Now lets compare the loss and Accuracy of 3 models to see which model gives better result

Model 2: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5, trainingsize 1000

Model 5: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5, trainingsize 2000

Model 6: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5, trainingsize 2000, Padding being same



When the models' performances were compared, it was found that the model did not gain much from using strides with padding. With the addition of a Max Pooling Layer, Model 5 outperformed the Strides model in accuracy by 14%. Additionally, an improved accuracy of 81% was attained by fine-tuning the network and expanding the training dataset from 1000 to 2000 samples.

We plotted Models 5 and 6 to answer the second question and provide a visual comparison of their performance. The graphs clearly show that Model 5 had the lowest loss of 45.9% and the highest accuracy of all the models, reaching 81%. The model performed significantly better once the training samples were increased to 2000 and various augmented photos were added.

1. Now change your training sample so that you achieve better performance than those from Steps1 and 2. This sample size may be larger, or smaller than those in the previous steps. The objectiveis to find the ideal training sample size to get best prediction results

As we saw in above graph that with the increase in training sample size the Accuracy is also increasing hence will increase the sample size to 3000 and 5000 for better performance

Training Sample 3000

Evaluating the Performance of Model_6 on test set

Found 3000 files belonging to 2 classes.

Found 500 files belonging to 2 classes.

data batch shape: (32, 180, 180, 3)labels batch shape: (32,)

Using few of the techniques such as random flip, random zoom, random rotation so as to create augmented versions of the image

Model - 7 MaxPooling Operation with Increase in filters from 32 to 256 in 5 Input Layers with thedata being used from the Augmented Images and a droput rate of 0.5 (Training Sample - 3000)

Model: "model_7"

LayLayer (type)	Output Shape	Param #
input_10 (InputLayer)	[(None, 180, 180, 3)]	0
sequential_3 (Sequential)	(None, 180, 180, 3)	0
rescaling_9 (Rescaling)	(None, 180, 180, 3)	0
conv2d_47 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_30 (MaxPoolin	(None, 89, 89, 32)	0g2D)
conv2d_48 (Conv2D)	(None, 87, 87, 64)	18496
max_pooling2d_31 (MaxPoolin	(None, 43, 43, 64)	0g2D)
conv2d_49 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_32 (MaxPoolin	(None, 20, 20, 128)	0g2D)
conv2d_50 (Conv2D)	(None, 18, 18, 256)	295168
max_pooling2d_33 (MaxPoolin	(None, 9, 9, 256)	0g2D)
conv2d_51 (Conv2D)	(None, 7, 7, 256)	590080
flatten_9 (Flatten)	(None, 12544)	0
dropout_7 (Dropout)	(None, 12544)	0
dense_9	(Dense) (None, 1)	12545

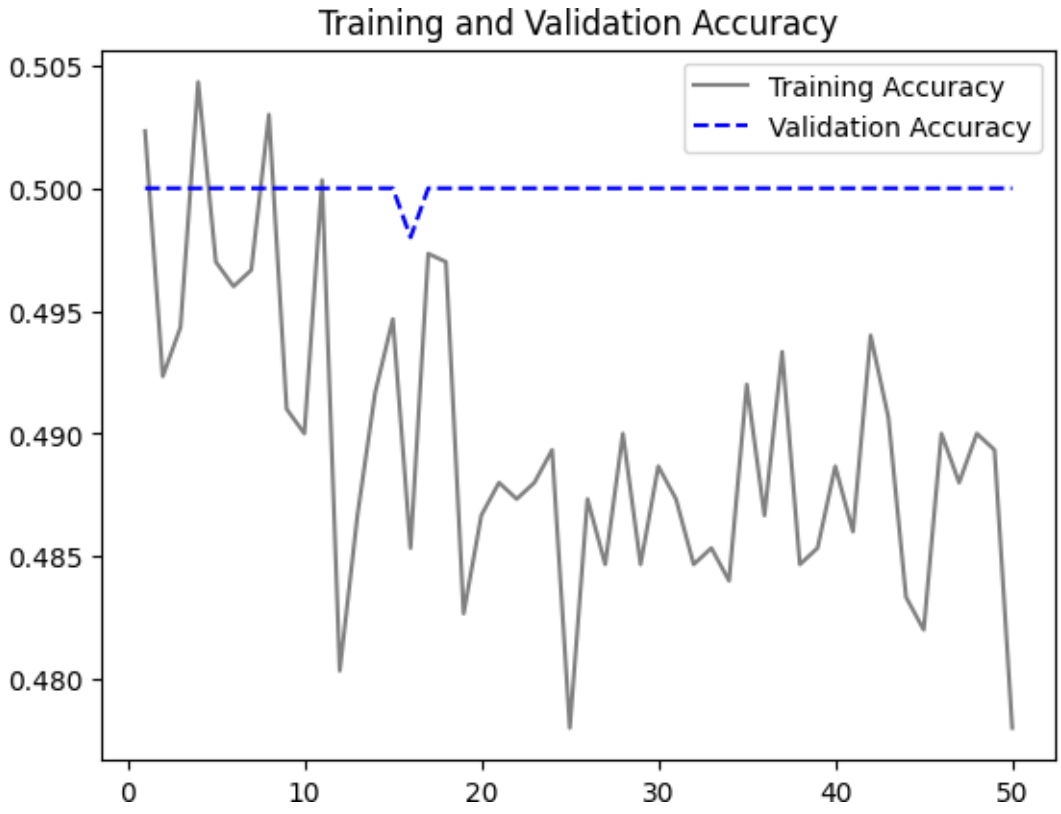
==========

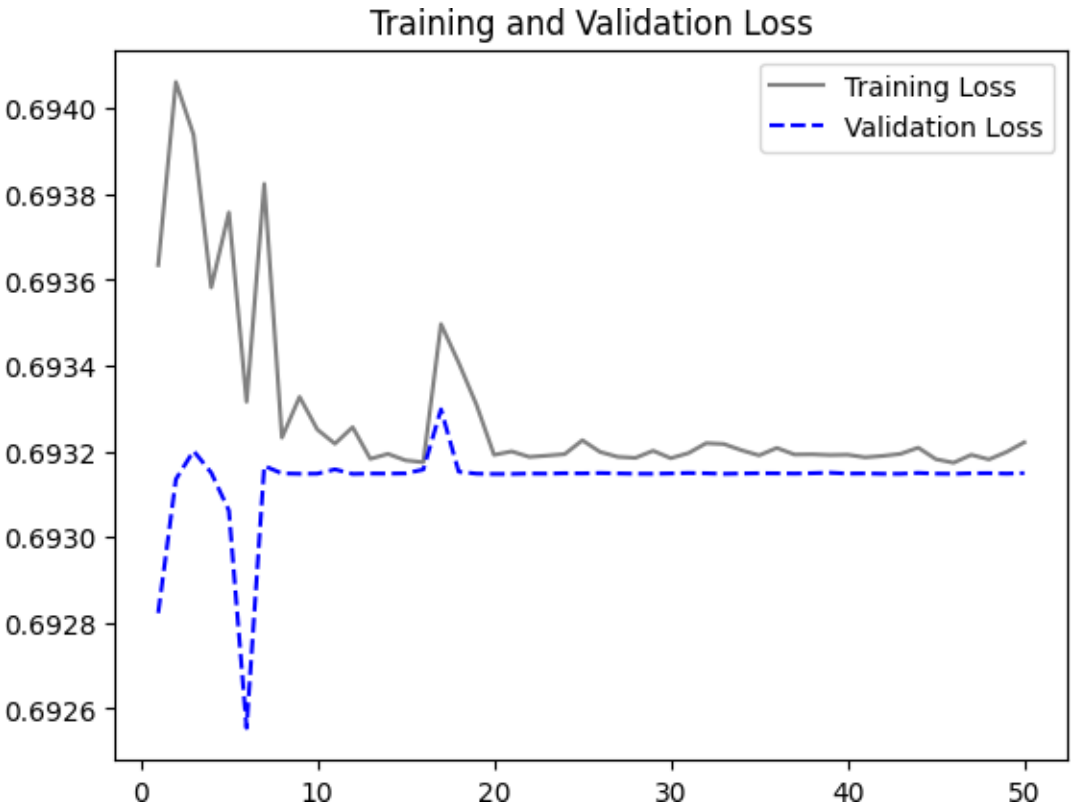
Total params: 991,041

Trainable params: 991,041

Non-trainable params: 0

_____




Training and Validation Loss
Training Loss
Validation Loss
0.6940
0.6938
0.6936
0.6934
0.6932
0.6930
0.6928
0.6926
0
10
20
30
40
50

Loss: 0.693
Accuracy: 0.500

In the previous Model 6, we attempted to replace the conventional max pooling operation with strides, but the results were not as promising as expected. and in model 7 we used Maxpooling only. Therefore, we are exploring a hybrid approach that combines both max pooling and stridesto evaluate the performance of this new model.

Max pooling is a downsampling operation that reduces the spatial dimensions of the feature map, aiming to capture the most prominent features while discarding less relevant information. On the other hand, strides determine the step rate of the sliding window used to extract and learn the features from the data. This hybrid approach aims to leverage the advantages of both tech- niques, potentially enhancing the model's ability to capture intricate patterns and features while maintaining computational efficiency.

Model - 8 MaxPooling + Strides of Step-Size 2 Operation with Increase in filters from 32 to 256

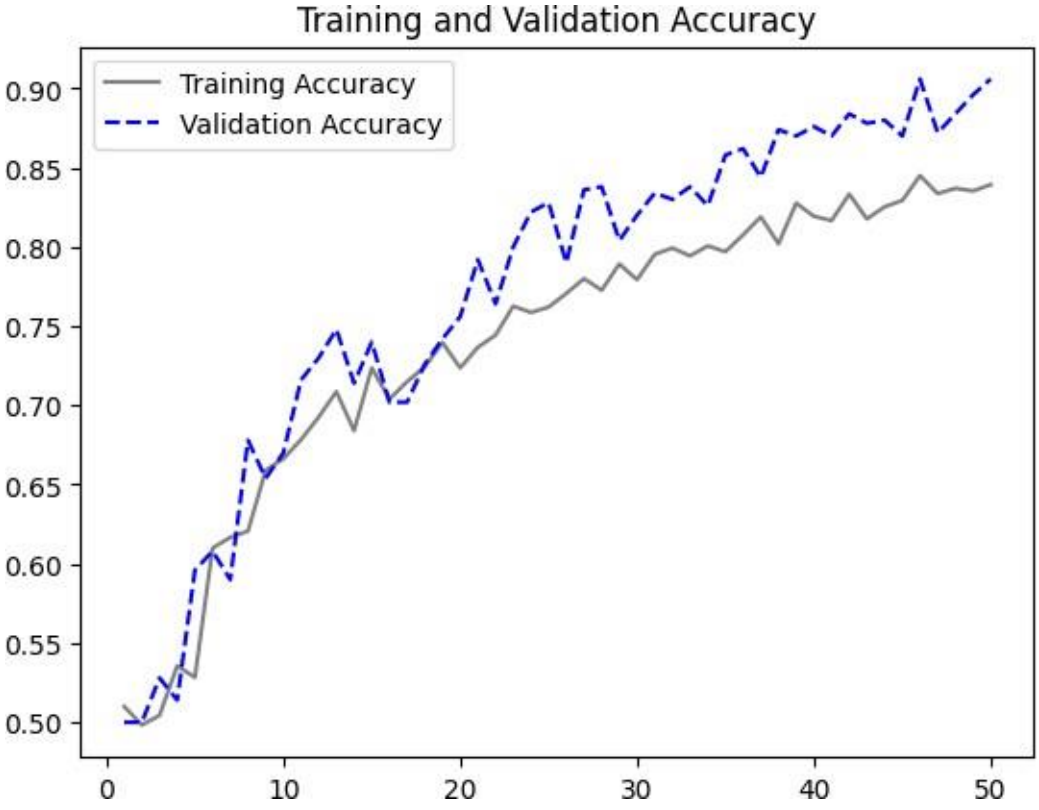
in 5 Input Layers with the data being used from the Augmented Images and a droput rate of 0.5(Training Sample - 3000)

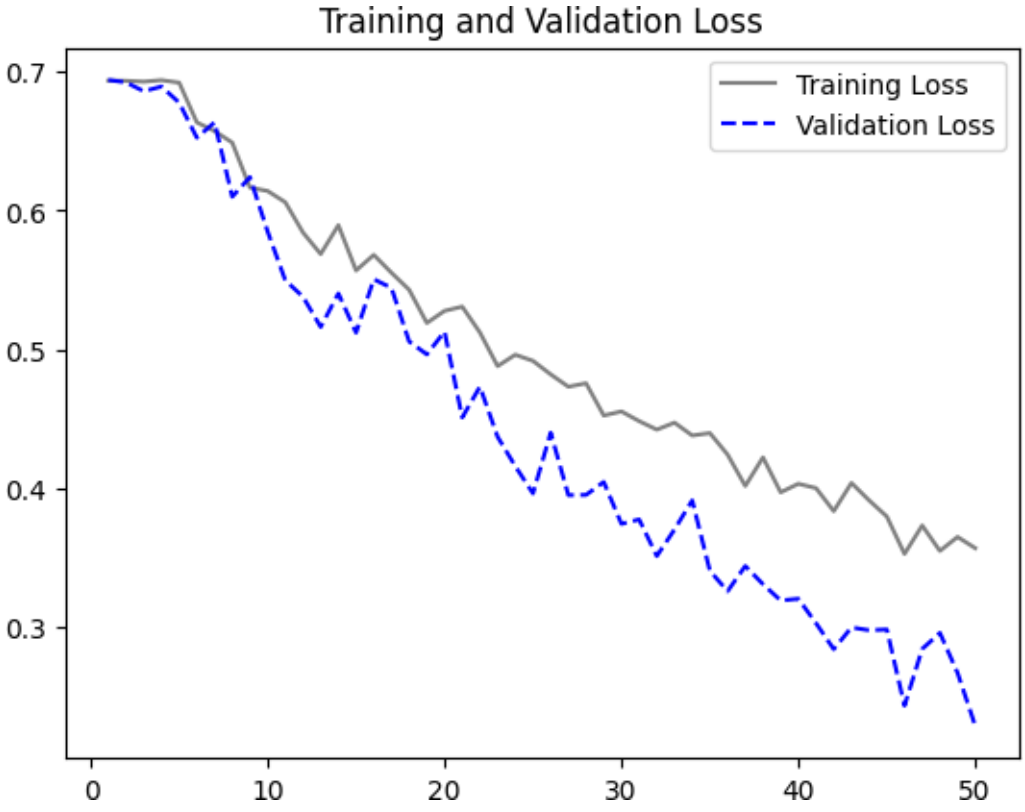
Model Summary:

Model: "model_8"

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 180, 180, 3)]	0
sequential_3 (Sequential)	(None, 180, 180, 3)	0
rescaling_10 (Rescaling)	(None, 180, 180, 3)	0
conv2d_52 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_34 (MaxPoolin	(None, 89, 89, 32)	0g2D)
conv2d_53 (Conv2D)	(None, 87, 87, 64)	18496
max_pooling2d_35 (MaxPoolin g2D)	(None, 43, 43, 64)	0
conv2d_54 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_36 (MaxPoolin g2D)	(None, 20, 20, 128)	0
conv2d_55 (Conv2D)	(None, 18, 18, 256)	295168
max_pooling2d_37 (MaxPoolin g2D)	(None, 9, 9, 256)	0
conv2d_56 (Conv2D)	(None, 7, 7, 256)	590080
flatten_10 (Flatten)	(None, 12544)	0
dropout_8 (Dropout)	(None, 12544)	0
dense_10 (Dense)	(None, 1)	12545

Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
10
20
30
40
50


Training and Validation Loss
Training Loss
Validation Loss
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50

Loss: 0.457
Accuracy: 0.812

Model - 9 MaxPooling + Strides of Step-Size 2 with Padding turned on Operation with Increase in filters from 32 to 512 in 5 Input Layers with the data being used from the Augmented Images and a dropout rate of 0.5 (Training Sample - 3000)

Model summary:

Model: "model_9"

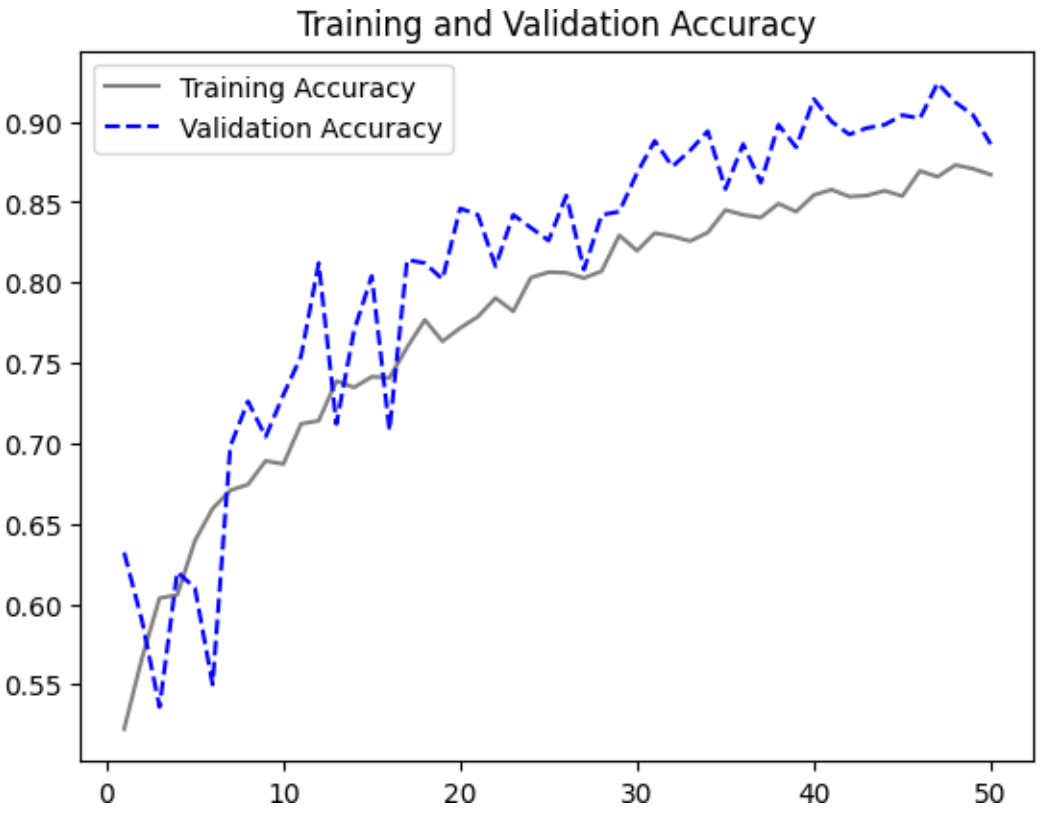
Layer (type)	Output Shape	Param #
input_12 (InputLayer)	[(None, 180, 180, 3)]	0
sequential_3 (Sequential)	(None, 180, 180, 3)	0
rescaling_11 (Rescaling)	(None, 180, 180, 3)	0
conv2d_57 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_38 (MaxPoolin	(None, 89, 89, 32)	0g2D)
conv2d_58 (Conv2D)	(None, 87, 87, 64)	18496
max_pooling2d_39 (MaxPoolin	(None, 44, 44, 64)	0g2D)
conv2d_59 (Conv2D)	(None, 42, 42, 128)	73856
max_pooling2d_40 (MaxPoolin	(None, 21, 21, 128)	0g2D)
conv2d_60 (Conv2D)	(None, 19, 19, 256)	295168
max_pooling2d_41 (MaxPoolin	(None, 10, 10, 256)	0g2D)
conv2d_61 (Conv2D)	(None, 8, 8, 512)	1180160
flatten_11 (Flatten)	(None, 32768)	0
dropout_9 (Dropout)	(None, 32768)	0
dense_11 (Dense)	(None, 1)	32769

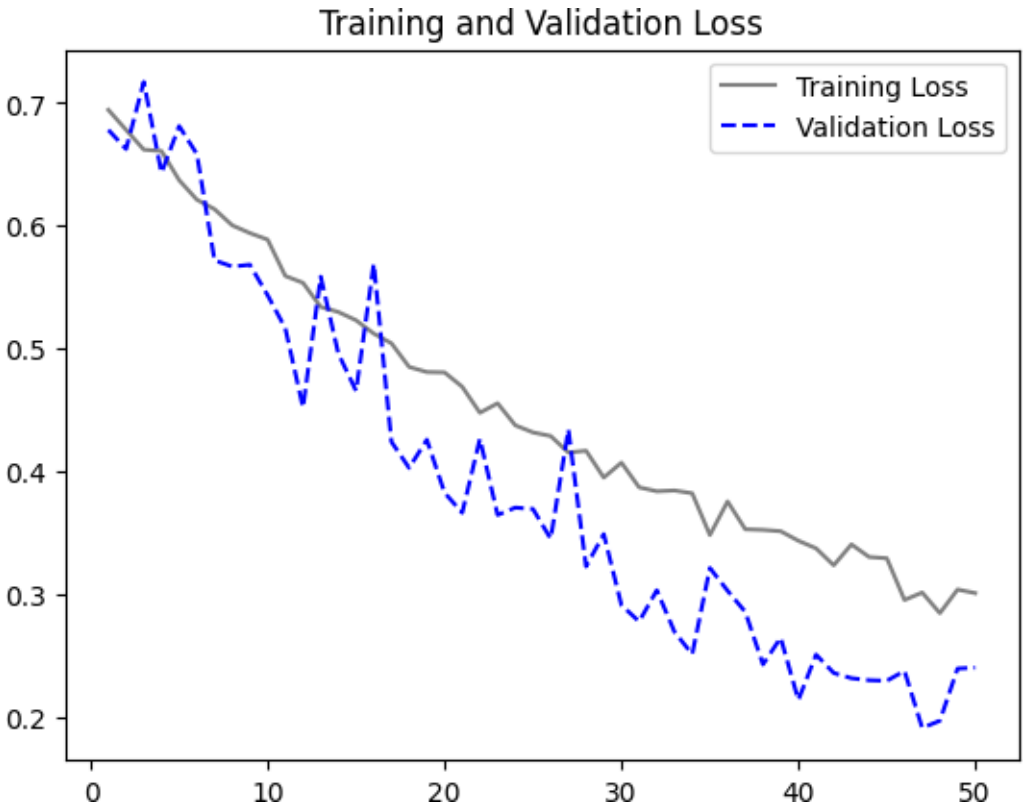
==========

Total params: 1,601,345

Trainable params: 1,601,345

Non-trainable params: 0





Loss: 0.409

Accuracy: 0.832

Let's see which of the models have best performance when the training sample was set to 3000. Note: Here models 8 and 9 were trained differently with strides being used with maxpooling andstrides being used with maxpooling and padding turned on

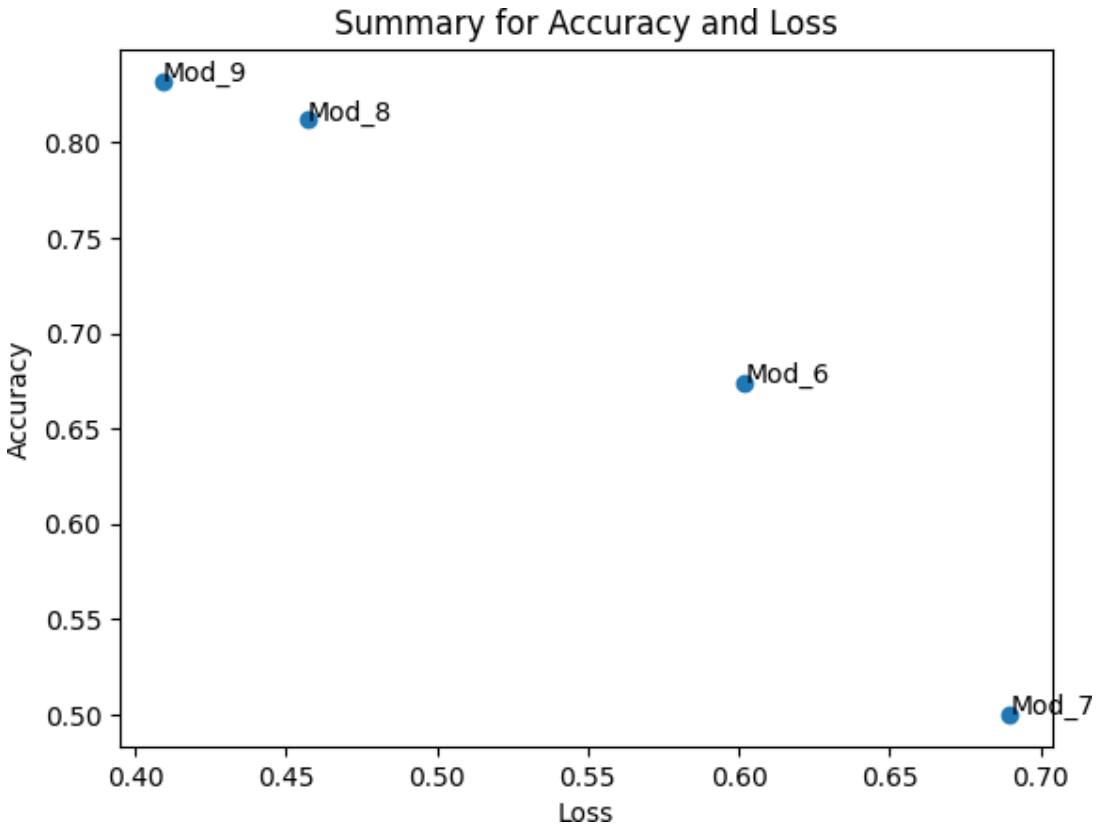
Model 6: trides Operation with Padding being "Same" ,filters from 32 to 512, 5 Input Layers, droput rate of 0.5, Training Sample - 3000

Model 7: MaxPooling Operation,filters from 32 to 512, 5 Input Layers, droput rate of 0.5, TrainingSample - 3000

Model 8: MaxPooling + Strides of Step-Size 2,filters from 32 to 512, 5 Input Layers, droput rate of 0.5, Training Sample - 3000

Model 9: MaxPooling + Strides of Step-Size 2 with Padding turned on,filters from 32 to 512, 5

Input Layers, droput rate of 0.5, Training Sample - 3000



Here we can clearly see that the model which was built with 5 layers using maxpooling along with strides and padding on was giving the highest accuracy i.e. 83.2 % with least loss amoung the other 2 models i.e. 40.9%.

Now, we are increasing the training sample to 5000 and building a model from scratch to check it's performance on the unseen data.

Training Sample - 5000

Found 5000 files belonging to 2 classes.
Found 500 files belonging to 2 classes.
Found 500 files belonging to 2 classes

data batch shape: (32, 180, 180, 3)labels batch shape: (32,)

Model - 10 MaxPooling Operation with Increase in filters from 32 to 256 in 5 Input Layers with the data being used from the Augmented Images and a droput rate of 0.5 (Training Sample - 5000)

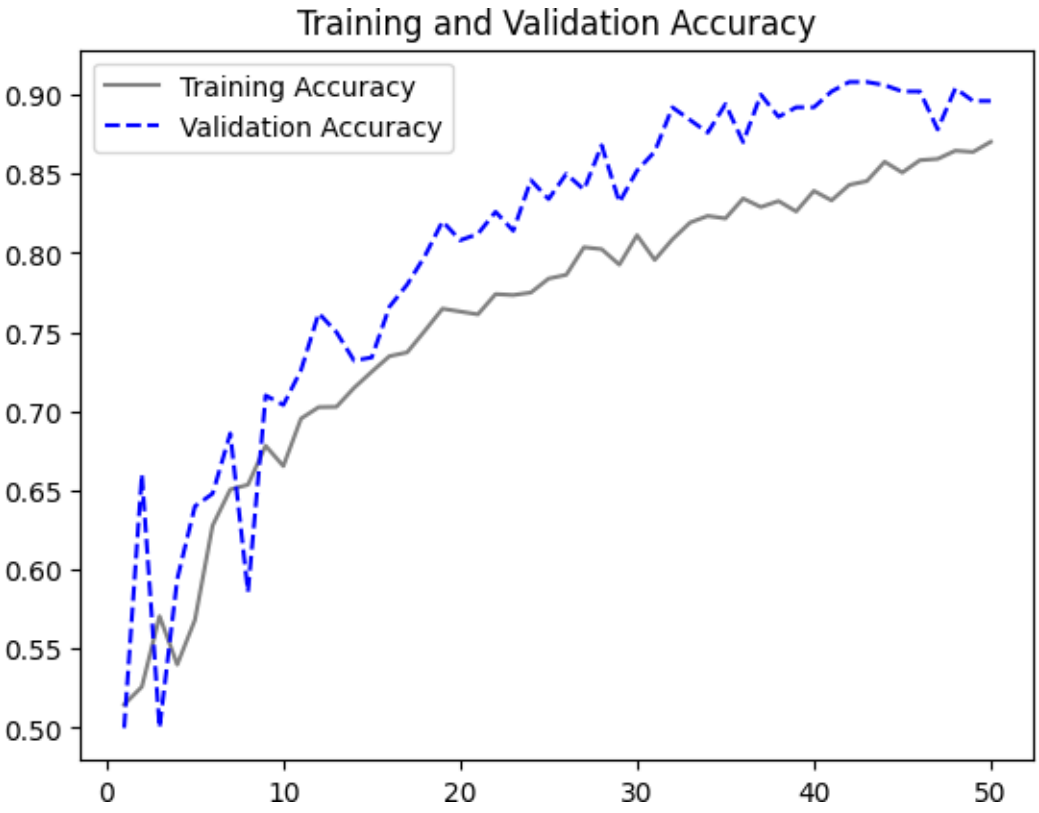
Model Summary:

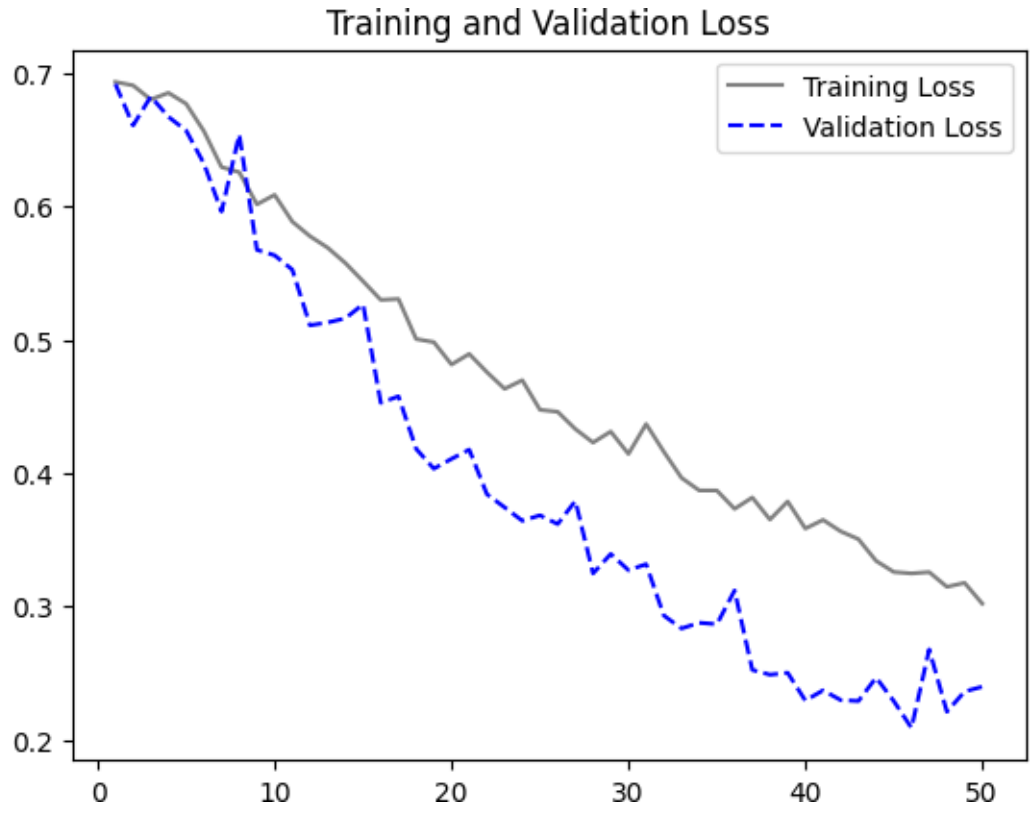
Model: "model_10"

conv2d_63 (Conv2D)	(None, 87, 87, 64)	18496
max_pooling2d_43 (MaxPoolin g2D)	(None, 43, 43, 64)	0
conv2d_64 (Conv2D)	(None, 41, 41, 128)	73856
max_pooling2d_44 (MaxPoolin g2D)	(None, 20, 20, 128)	0
conv2d_65 (Conv2D)	(None, 18, 18, 256)	295168
max_pooling2d_45 (MaxPoolin g2D)	(None, 9, 9, 256)	0
conv2d_66 (Conv2D)	(None, 7, 7, 256)	590080
flatten_12 (Flatten)	(None, 12544)	0
dropout_10 (Dropout)	(None, 12544)	0
dense_12 (Dense)	(None, 1)	12545

==========

Total params: 991,041
Trainable params: 991,041
Non-trainable params: 0


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
10
20
30
40
50


Training and Validation Loss
Training Loss
Validation Loss
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50

Loss: 0.283
Accuracy: 0.884

Summary for Question 3

We built four models, three of which were trained using a 3000 sample size. The best-performing model has an accuracy of 83.2%. Interestingly, the accuracy increased to 88.4% when we increased the training sample to 5000. Thus, we conclude that a training sample size of 5000 significantly improves the model's performance. As for the likely cause of the validation loss being less than the training loss, the split approach that was used is probably a factor.

Here, the validation and test sets are fixed at 500 apiece, but the training sample is almost five thousand times larger. Furthermore, it is crucial to recognize that regularizations like dropout or L1 and L2 regularizers are important during training and contribute to the training loss calculation. On the other hand, these regularizers are turned off during the validation or test phase, which could result in a smaller loss than the training loss.

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
Leveraging a Pre-Trained Model - VGG16 VGG - Model 1 (1000 Training Samples)

Leveraging a Pre-Trained Model - VGG16 VGG - Model 1 (1000 Training Samples)

Summary of the model

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_14 (InputLayer)	[(None, 180, 180, 3)]	0
block1_conv1 (Conv2D)	(None, 180, 180, 64)	1792
block1_conv2 (Conv2D)	(None, 180, 180, 64)	36928
block1_pool (MaxPooling2D)	(None, 90, 90, 64)	0
block2_conv1 (Conv2D)	(None, 90, 90, 128)	73856
block2_conv2 (Conv2D)	(None, 90, 90, 128)	147584
block2_pool (MaxPooling2D)	(None, 45, 45, 128)	0
block3_conv1 (Conv2D)	(None, 45, 45, 256)	295168
block3_conv2 (Conv2D)	(None, 45, 45, 256)	590080
block3_conv3 (Conv2D)	(None, 45, 45, 256)	590080
block3_pool (MaxPooling2D)	(None, 22, 22, 256)	0
block4_conv1 (Conv2D)	(None, 22, 22, 512)	1180160
block4_conv2 (Conv2D)	(None, 22, 22, 512)	2359808

block4_conv3 (Conv2D)	(None,	22,	22,	512)	2359808
block4_pool (MaxPooling2D)	(None,	11,	11,	512)	0
block5_conv1 (Conv2D)	(None,	11,	11,	512)	2359808
block5_conv2 (Conv2D)	(None,	11,	11,	512)	2359808
block5_conv3 (Conv2D)	(None,	11,	11,	512)	2359808
block5_pool (MaxPooling2D)	(None, 5, 5, 512)				0

===

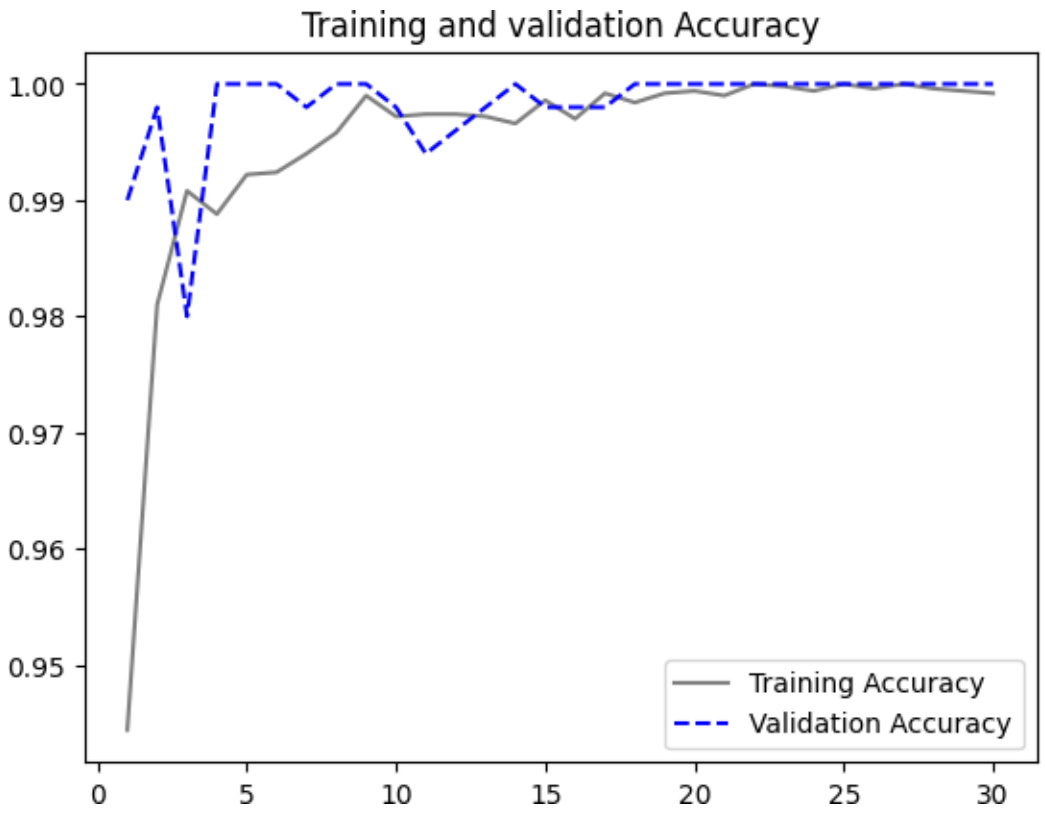
Total params: 14,714,688

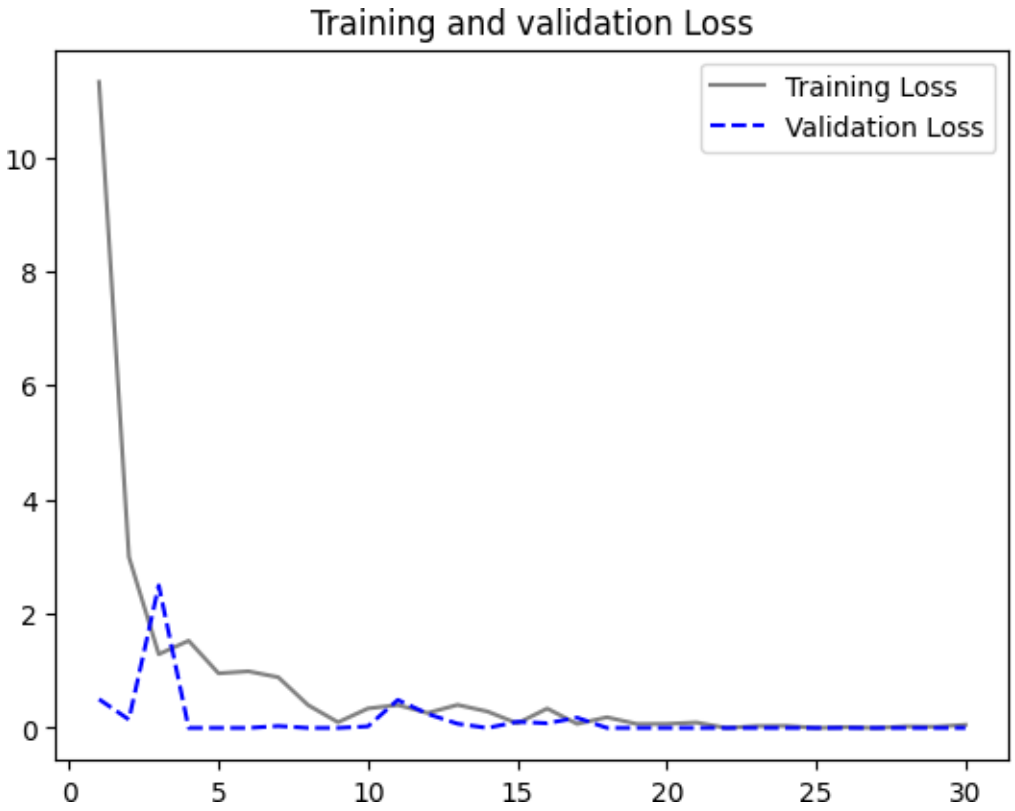
Trainable params: 14,714,688

Non-trainable params: 0

_____

VGG - Model 1 Dense Layer with 256 Nodes and Dropout Rate of 0.5 and optimizer being with the Original Images





Loss: 0.000
Accuracy: 1.000

VGG - Model 2 (1000 Training Samples)

Only the densely linked networks and the classifier are permitted to modify their weights while the pre-trained model is set to maintain its current weights throughout training.

This method helps avoid overfitting because the pre-trained model doesn't change, giving the model a solid base. Furthermore, freezing the pre-trained model training might be very helpful when working with limited training data and processing resources.

We can output the list of trainable weights both before and after freezing the pre-trained model to show the effect of this setting.

This is the number of trainable weights before freezing the conv base: 26

This is the number of trainable weights after freezing the conv base: 0
Convolution base summary

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_14 (InputLayer)	[(None, 180, 180, 3)]	0
block1_conv1 (Conv2D)	(None, 180, 180, 64)	1792
block1_conv2 (Conv2D)	(None, 180, 180, 64)	36928
block1_pool (MaxPooling2D)	(None, 90, 90, 64)	0
block2_conv1 (Conv2D)	(None, 90, 90, 128)	73856
block2_conv2 (Conv2D)	(None, 90, 90, 128)	147584
block2_pool (MaxPooling2D)	(None, 45, 45, 128)	0
block3_conv1 (Conv2D)	(None, 45, 45, 256)	295168
block3_conv2 (Conv2D)	(None, 45, 45, 256)	590080
block3_conv3 (Conv2D)	(None, 45, 45, 256)	590080
block3_pool (MaxPooling2D)	(None, 22, 22, 256)	0
block4_conv1 (Conv2D)	(None, 22, 22, 512)	1180160
block4_conv2 (Conv2D)	(None, 22, 22, 512)	2359808
block4_conv3 (Conv2D)	(None, 22, 22, 512)	2359808

block4_pool (MaxPooling2D)	(None, 11, 11, 512)	0
block5_conv1 (Conv2D)	(None, 11, 11, 512)	2359808
block5_conv2 (Conv2D)	(None, 11, 11, 512)	2359808
block5_conv3 (Conv2D)	(None, 11, 11, 512)	2359808
block5_pool (MaxPooling2D)	(None, 5, 5, 512)	0

==========

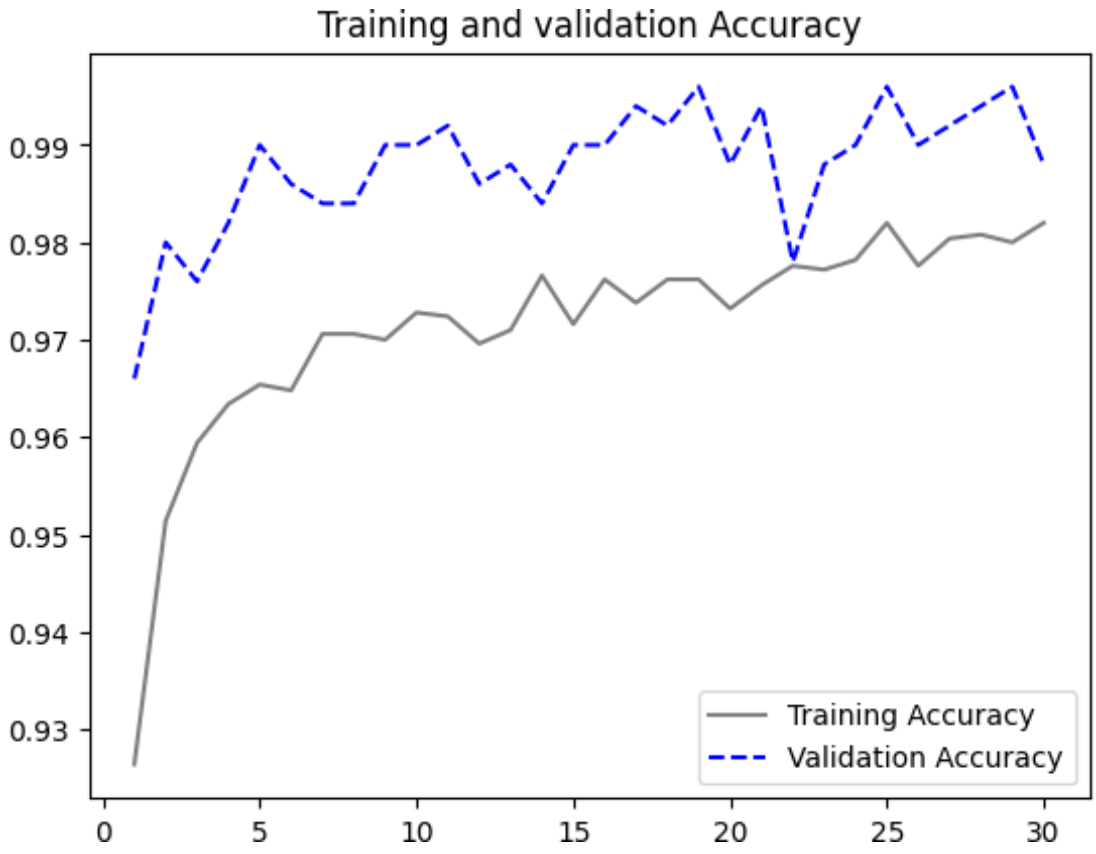
Total params: 14,714,688

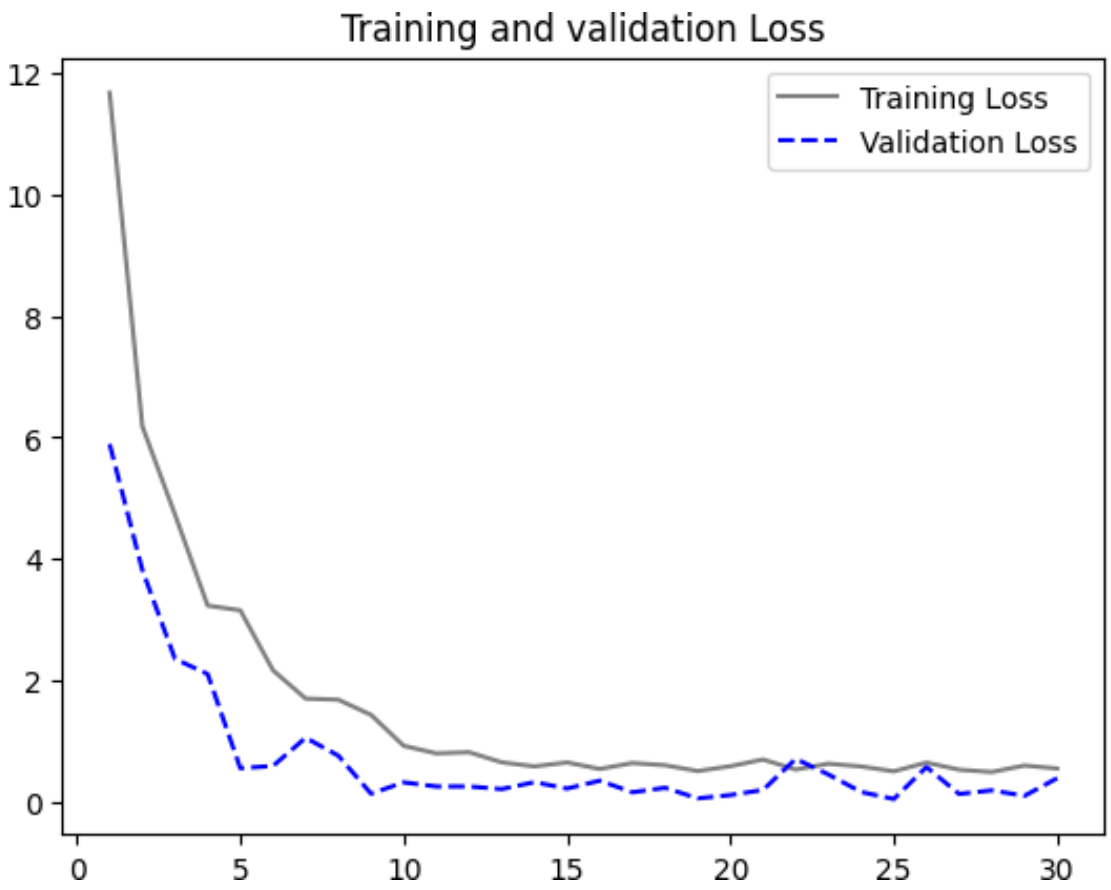
Trainable params: 0

Non-trainable params: 14,714,688

_____

VGG - Model 2 Dense Layer with 256 Nodes and Dropout Rate of 0.5 and optimizer being with the Augmented Images



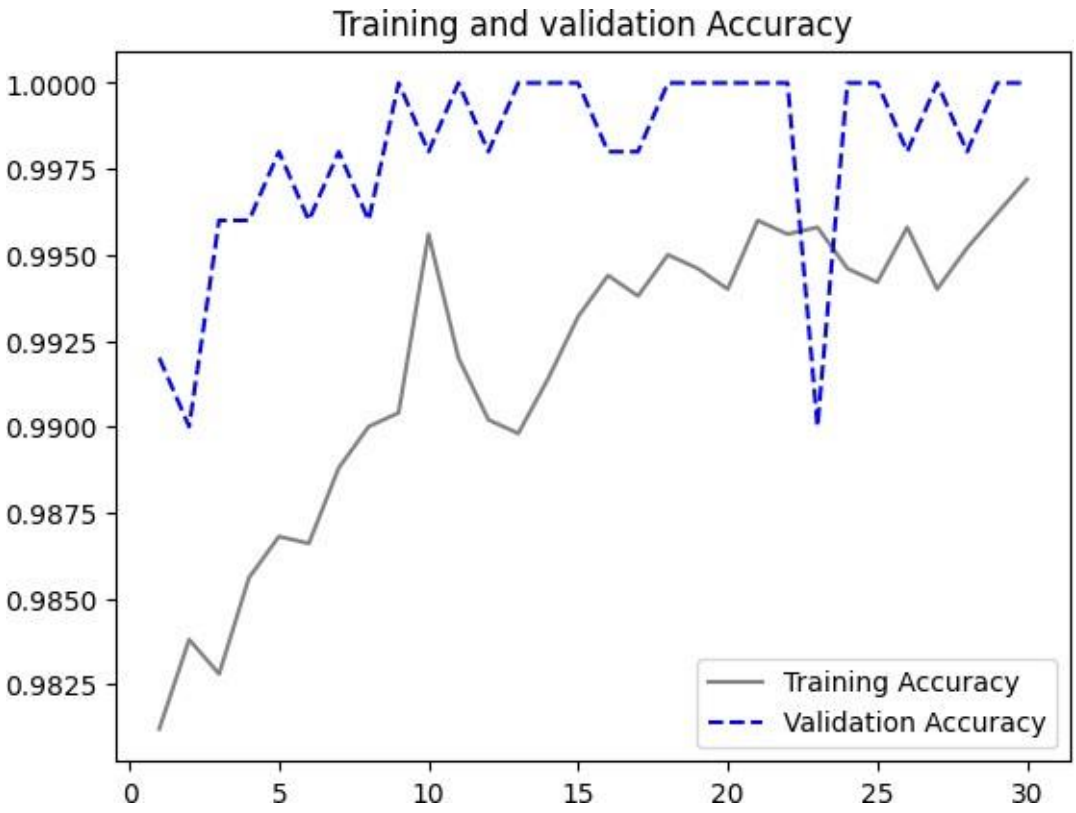


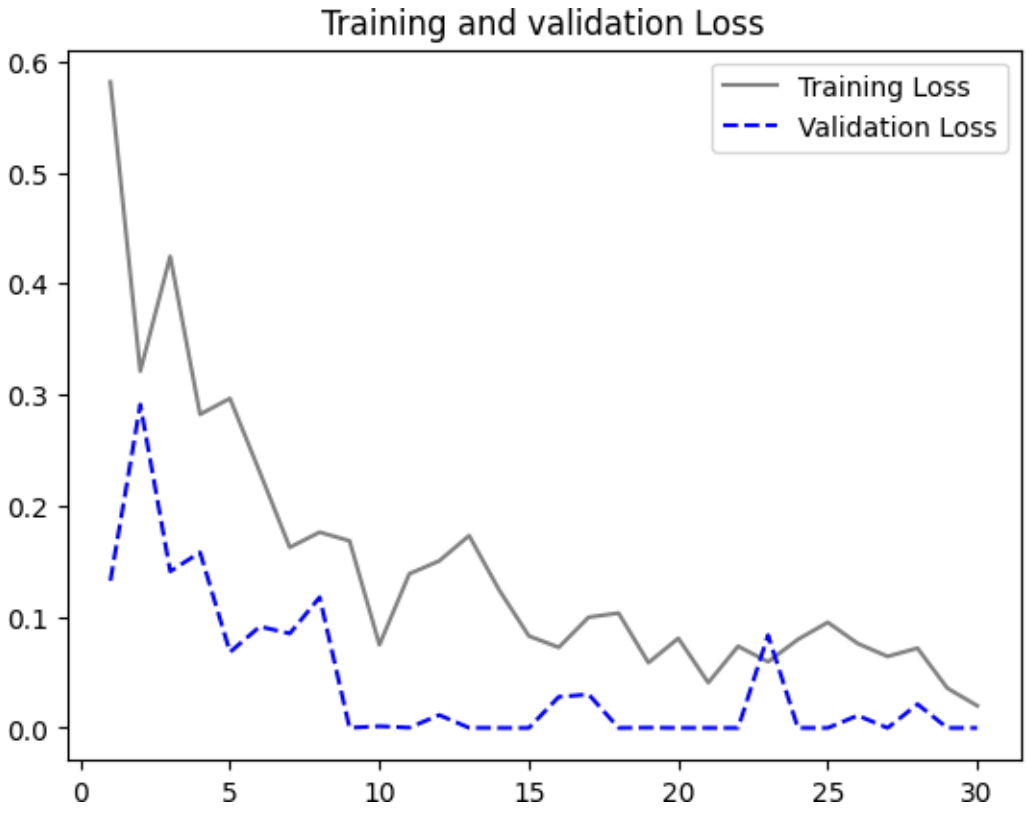
Loss: 0.026

Accuracy: 0.998

Fine Tuning the VGG_Model_2

It is important to realize that pre-trained networks are trained to handle a variety of use cases and classifications, and are not only employed for solitary picture classification tasks. The network's first layers are good at gathering broad data, while its later levels usually focus on extracting features that are unique to the issue at hand. By choosing to freeze the first few layers, we may effectively avoid overfitting and allow the model to incorporate more complex information related to our particular categorization problem. The model is encouraged to concentrate on understanding the subtle nuances of the target classification problem by using this strategic approach.





Loss: 0.000
Accuracy: 1.000

Using the pre-trained network VGG16, we created three models for the examination of the above two VGG16 models. Interestingly, we found that accuracy increased when the pre-trained network was frozen in its first layers and prevented from altering its weights during training. As such, we want to use the same methods with a training sample size of 5000 to construct two models.

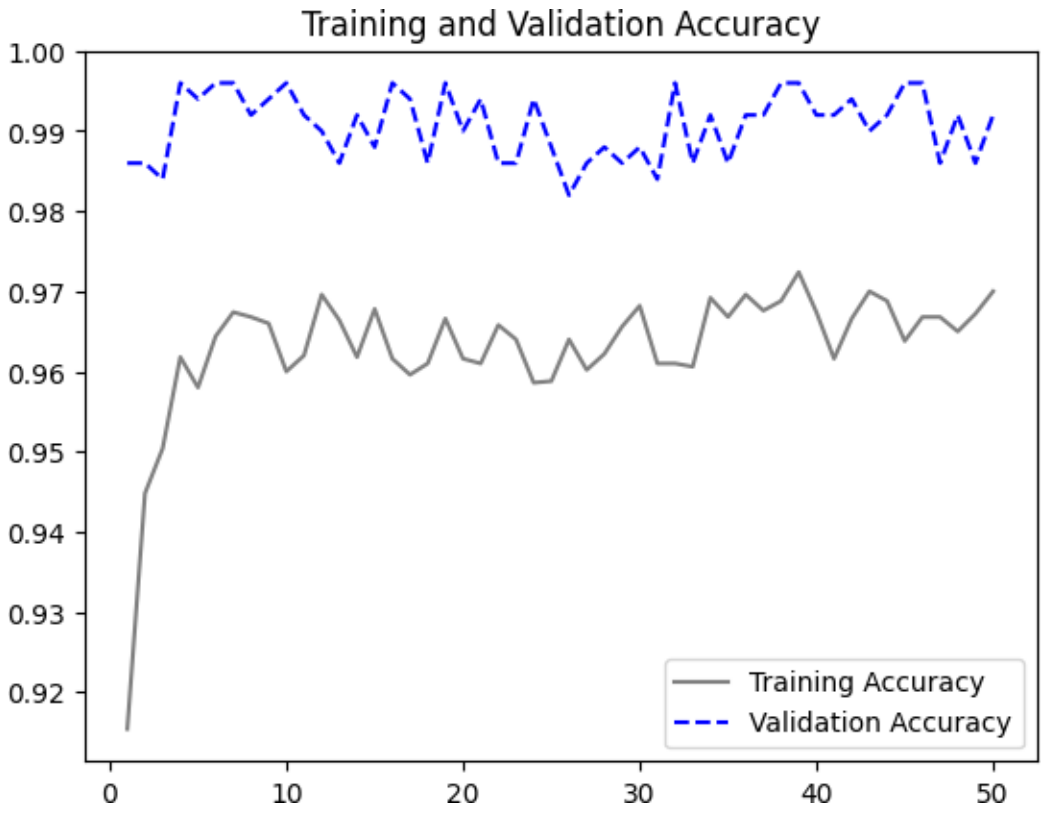
VGG - Model 3 (5000 Training Samples)

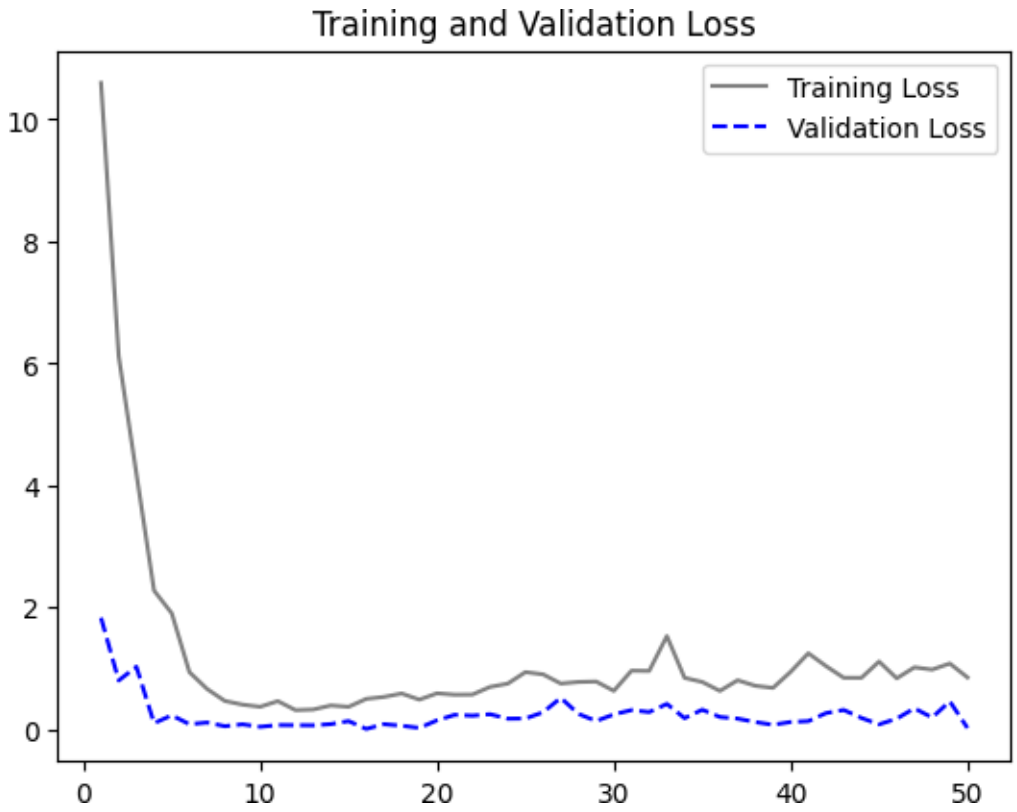
Model: "vgg16"

Layer (type)	Output Shape	Param #
input_17 (InputLayer)	[(None, None, None, 3)]	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688

Found 5000 files belonging to 2 classes.Found 500 files belonging to 2 classes. Found 500 files belonging to 2 classes.

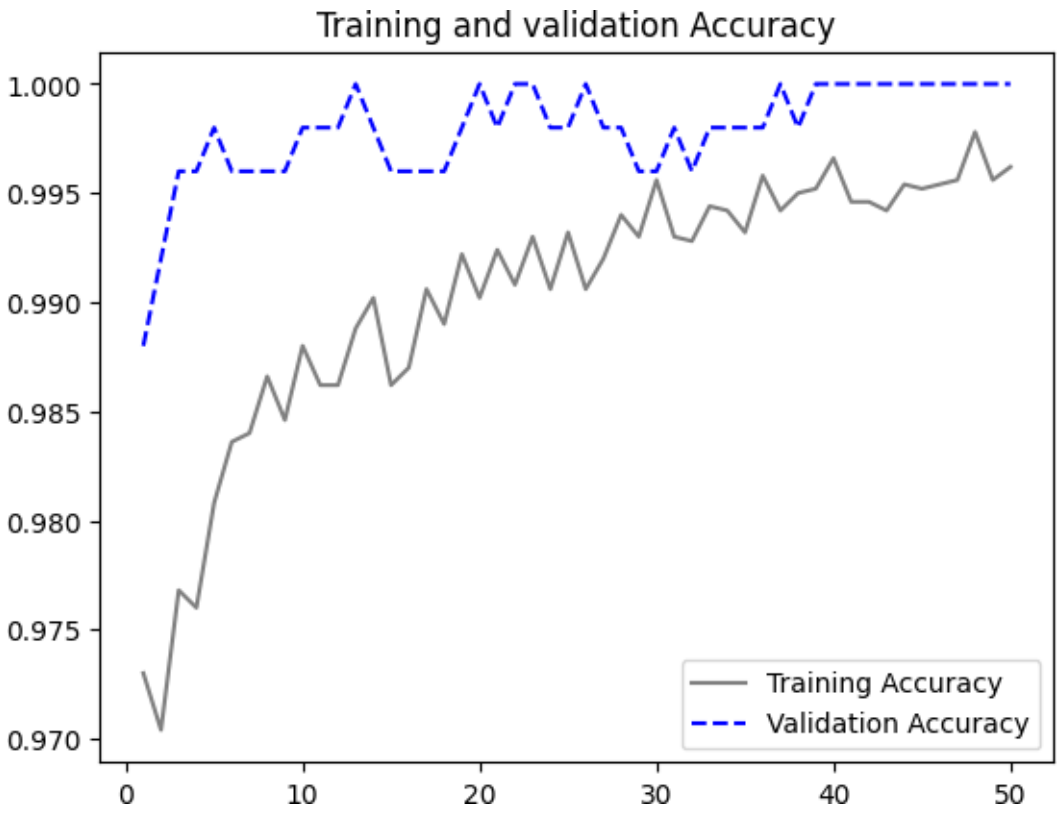


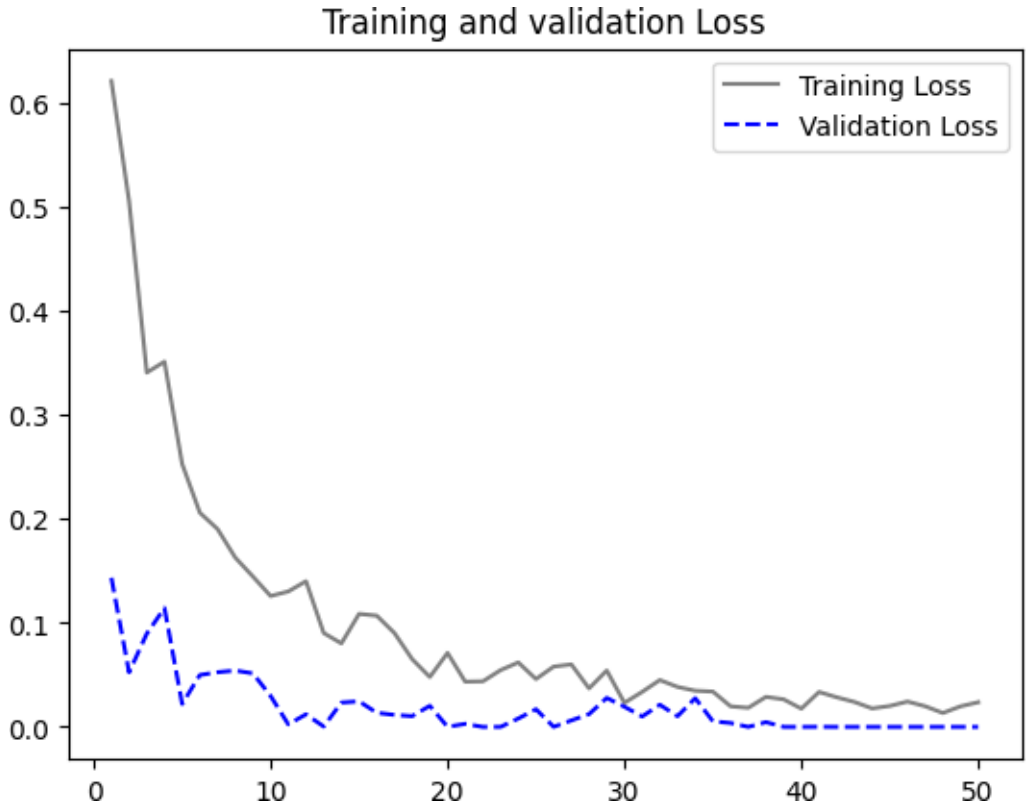


Loss: 0.134
Accuracy: 0.984

Fine Tunning VGG_Model_3 (Training Samples - 5000)

We have decided to freeze the first four layers in order to optimize VGG_Model3. By using this tactic, we hope to stop the model from overfitting and free it up to focus just on identifying the unique characteristics that are pertinent to our specific categorization task.As such, we have simultaneously made sure that the first four layers stay frozen and configured the pre-trained layers to remain unchanged during training. The model performs better with these improvements applied, especially when working with a training sample size of 5000.


Training and validation Accuracy
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0
10
20
30
40
50
Training Accuracy
Validation Accuracy

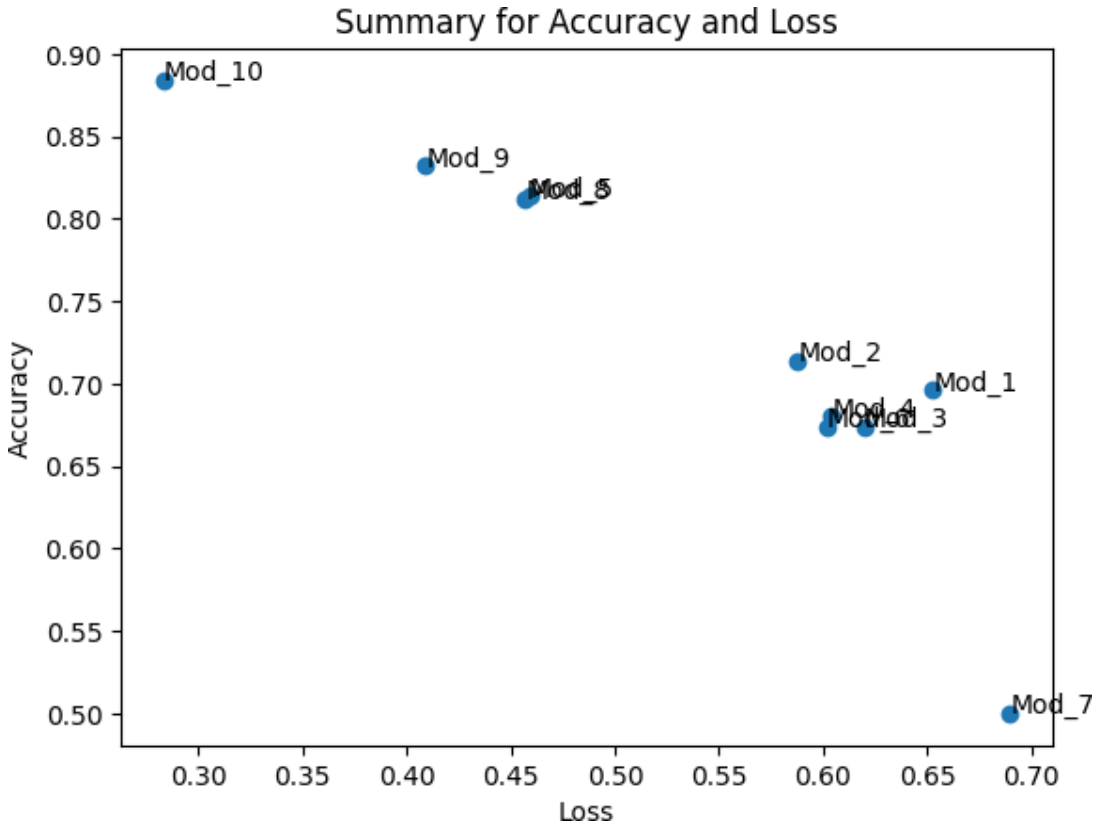


Loss: 0.000
Accuracy: 1.000

After building a total of 15 models—two of which are optimized copies of the original models—we are now prepared to do a comparative study in order to identify the top models in two different categories: Pre-Trained Models and Scratch Models. Our immediate goal is to assess which scratch-built model performs the best. The evaluation comprises a comparison of the accuracy and loss metrics of the ten models that were constructed using four distinct training data. Finding the ideal training sample size for the classification of cats and dogs is the main goal.

Model 1: filters from 32 to 256, 5 Input Layers
Model 2: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5 Model 3: filters from 32 to 512, 6 Input Layers, Augmented Images and Dropout rate of 0.5 Model 4: filters from 64 to 1024, 5 Input Layers, Augmented Images and Dropout rate of 0.6
Model 5: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5, training size 2000
Model 6: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5, training size 2000, Padding being same

Model 7: MaxPooling Operation,filters from 32 to 512, 5 Input Layers, Augmented Images, droput rate of 0.5, Training Sample - 3000
Model 8: MaxPooling + Strides of Step-Size 2,filters from 32 to 512, 5 Input Layers, Augmented Images, droput rate of 0.5, Training Sample - 3000
Model 9: MaxPooling + Strides of Step-Size 2 with Padding turned on,filters from 32 to 512, 5 Input Layers,Augmented Images, droput rate of 0.5, Training Sample - 3000
Model 10: filters from 32 to 512, 5 Input Layers,Augmented Images, droput rate of 0.5, Training Sample - 500

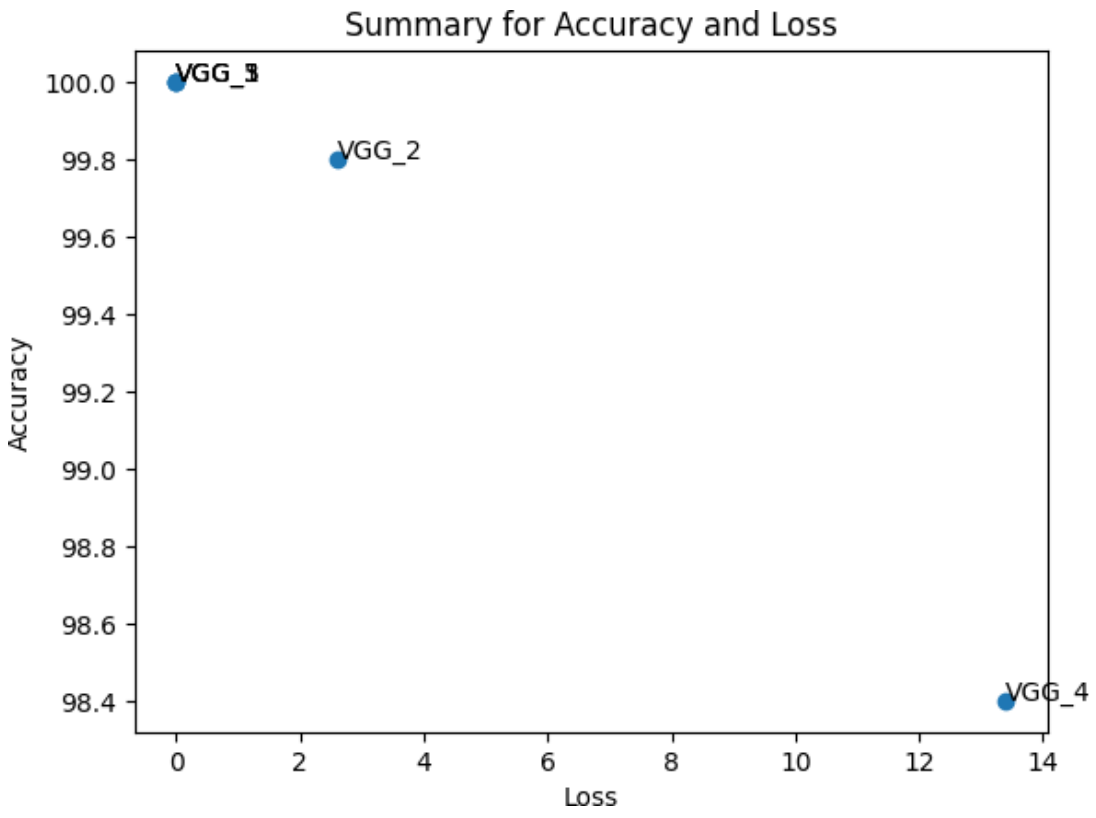


Out of all the scratch models, Model_10, which was trained on 5000 samples, turned out to be the best model with an impressive 88.4% accuracy and a 28.3% loss on the test set. A five-layer architecture with filters ranging from 32 to 256 was used to build Model 10. Augmented photos were integrated into the model during training, along with a max-pooling layer and a 0.5 dropout rate.

following, we created five models with the pre-trained vgg16 network. The first three were created with a sample size of 1000 and an optimizer named rmsprop, while the following two were created with a sample size of 5000 and an optimizer named Adam.

VGG 1: filters from 32 to 256, 5 Input Layers

VGG 2: filters from 32 to 256, 5 Input Layers, Augmented Images and Droput rate of 0.5 VGG 3: filters from 32 to 512, 6 Input Layers, Augmented Images and Dropout rate of 0.5VGG 4: VGG - Model 3 (5000 Training Samples)
VGG 5: Fine Tunning VGG_Model_3 (Training Samples - 5000)



When it came to pre-trained models, the two best performers were Fine-Tuned_VGG_Model_2 and FineTuned_VGG_Model_3, or Model_5and model_3.

with a meagre 0.00% loss and an astounding 100% accuracy. This model was built with 2000 and 5000 training samples, and it was tuned at a learning rate of 0.000001 using the Adam optimizer.

Conclusion

According to the aforementioned analysis, a model's accuracy is closely related to the volume of training data and the underlying architecture, especially when the model is trained directly from its own data. On the other hand, if a pretrained model is used, the accuracy depends on the particular test set that is being assessed. It's important to keep in mind that some sample sets could be more difficult to work with than others, and strong results on one set might not translate to all other sets.

Scratch Model

The profound impact on test accuracy stems from both the size of the training dataset and the chosen model architecture. Notably, integrating contemporary architectural features like residual connections, batch normalization, and depthwise separable convolutions into a basic model, alongside employing data augmentation and dropout techniques, significantly boosted test accuracy. Furthermore, expanding the training dataset from 1,000 to 3,000 samples led to substantial improvements in accuracy. Further increasing the dataset to 6,000 samples resulted in test accuracy levels similar to pretrained models. This phenomenon highlights overfitting, where a lack of samples limits the model's ability to generalize. Increasing the dataset size broadens the model's exposure to the underlying data distribution, enhancing test accuracy.

Pretrained network

However, even while test accuracy is highest when using a pretrained network, the size of the training sample has little bearing on the results. This is a result of the accuracy-measuring model not being trained using the data it handles. Furthermore, methods like data augmentation and fine-tuning do not significantly improve accuracy because the original pretrained model already has high-performance accuracy, owing to the large

size of the dataset used in the pretrained VGG16 model (more than 500 MB in size and more than 138 million parameters).

Recommendations

With a rise in training data from 1,000 to 3,000 samples, the simple scratch model's accuracy increased significantly, from 0.696 to 0.88.4. A test accuracy of about 83.2% was obtained using 3,000 training samples, three contemporary architectures, data augmentation, and dropout approaches. Moreover, the test accuracy was 88% when 5,000 samples were used to train the model. Because Model 10 is the most accurate of the scratch models, it is advised to use it first.

Since we achieved 100% accuracy with the pre-tuned model, I also think that using pretrained models isn't always the best option. Context-independent reasoning states that the background or context of an image does not influence the ability to discriminate between a dog and a cat in the images used for this exercise. Therefore, as long as the sample photos belong to the same category as the target used to train the model, a pretrained model that can identify images can be applied to any image differentiation job.