Convolution

Introduction:

In this Project, we will explore the relationship between the training sample size and the choice of network architecture in the Cats & Dogs example. We will use techniques to reduce overfitting, improve performance, and evaluate the performance on a validation and test set. We will compare the results obtained using a network trained from scratch and a pre-trained network.

This collection includes 25,000 images of dogs and cats, 12,500 from each class. We will build a new dataset with three subsets: a training set with 1,000, 1500, and 2000 samples of each class; a validation set with 500 samples of each class; and a test set with 1,000 samples of each class.
Then we'll build a convnet with two more Conv2D and MaxPooling 2D stages. This increases the model's capacity while also reducing the feature's size. We begin with 180 x 180-pixel inputs and conclude with 7 x 7 feature maps before flattening the layer. We are using the dropout regularisation technique to reduce the overfitting of the model. Because we are looking at a binary classification problem, end the model with a single unit (a Dense layer of size 1) and a sigmoid activation. For the compilation step, we will use RMSprop optimizer, as usual. we'll use binary cross-entropy as the loss function. Then we will fit the model with training and validation datasets. We will evaluate the model with the test dataset.

Then we will add data augmentation to the model and evaluate the model. We will repeat the process with different training dataset sizes 1500, and 2000 respectively. And evaluate the models with dropout, with and without data augmentation.

Then we will use pre-trained convolution bases with classifiers on the same datasets with different sizes with dropout, with and without data augmentation, with and without fine tunning for all the datasets. And finally, Evaluate the performance.

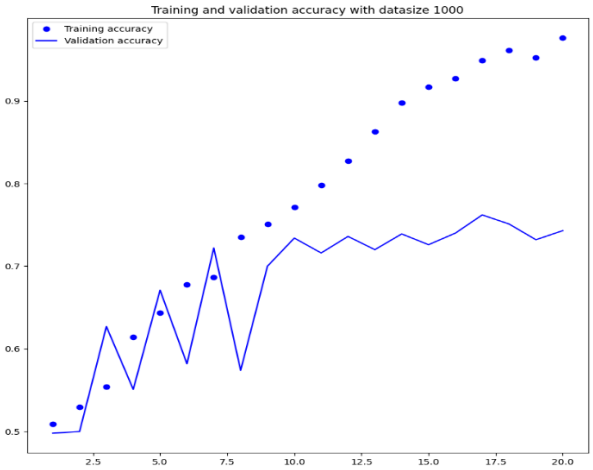
Summary

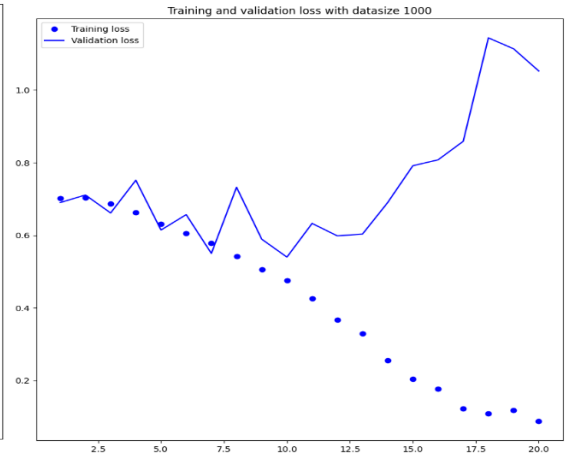
1.Consider the Cats & Dogs example. Start initially with a training sample of 1000, a validation sample of 500, and a test sample of 500 (like in the text). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?

We have trained a model from scratch with a training sample size of 1000, a validation sample size of 500, and a test sample size of 500. And to reduce overfitting we have used "dropout".

With Dropout:

model1 with dropout	**Training**	**Validation**	**Test**
Accuracy	97.6%	74%	73.9%
Loss	0.0873	1.0523	0.5505





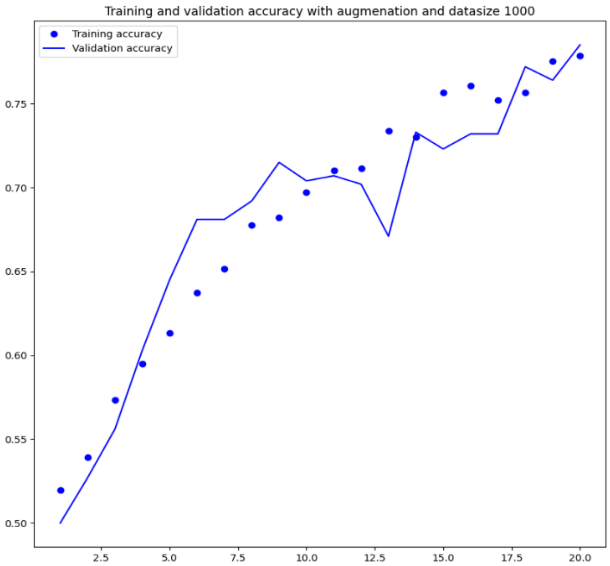
From the plots, we can say that the training accuracy improves linearly with time, until reaching almost 100%, but the validation accuracy peaks at 74%. The validation loss reaches its lowest level after only 10 epochs and then levels, but the training loss decreases linearly as training progresses. From the table, we can say that the test accuracy was 73.9%.

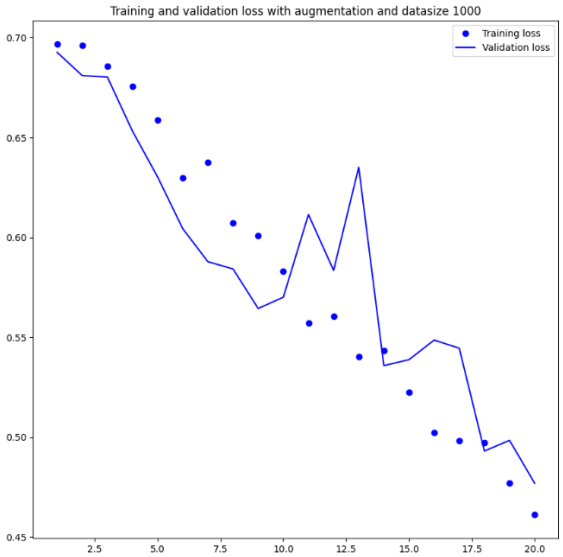
This happened due to overfitting, which was caused by having too few samples to learn from, rendering you unable to train a model that can generalize to new data. So, we have used data augmentation on the same model.

The results are as follows:

With Dropout and Data Augmentation:

	Training	**Validation**	**Test**
Accuracy	77.85%	78%	75.9%
Loss	0.4610	0.4768	0.5047





From the plots, we can say that the training accuracy improves linearly with time, and the validation accuracy peaks at 78.95%. The training and validation losses go on to reduce linearly as training progresses. We might have trained the model with more epochs which might improve the training accuracy and reduce the losses. From the table, we can say that the test accuracy was 75.9% which was far better than the model without data augmentation. Also, we might have trained the model with a greater number of epochs which might have improved both the training and test accuracies of the model.

2. Increase your training sample size. You may pick any amount. Keep the validation and test samples the same as above. Optimize your network (again training from scratch). What performance did you achieve?

We have again re-trained the model from scratch with the increase in training sample size to 1500 and by keeping the remaining two sets as same with dropout.

With Dropout:

	Training	**Validation**	**Test**
Accuracy	97.57%	72%	71.3%
Loss	0.0651	1.3075	0.5763

we can say that the training accuracy improves linearly with time, until reaching almost 100%, but the validation accuracy peaks at 72%. The validation loss reaches its lowest level after only 15 epochs and then levels, but the training loss decreases linearly as training progresses. From the table, we can say that the test accuracy was 71.3%.

the accuracy was not so good. This might happen due to overfitting, which was caused by having too few samples to learn from, rendering you unable to train a model that can generalize to new data. So, we have used data augmentation on the same model.

The results are as follows:

With Dropout and Data Augmentation:

	Training	**Validation**	**Test**
Accuracy	79.4%	77%	78.5%
Loss	0.4466	0.4619	0.4674

we can say that the training accuracy improves linearly with time, and the validation accuracy peaks at 77%. The training and validation losses go on to reduce linearly as training progresses. We might have trained the model with more epochs which might improve the training accuracy and reduce the losses. From the table, we can say that the test accuracy was 78.5% which was far better than the model without data augmentation and with a training sample size of 1000. Also, we might have trained the model with a greater number of epochs which might have improved both the training and test accuracies of the model.

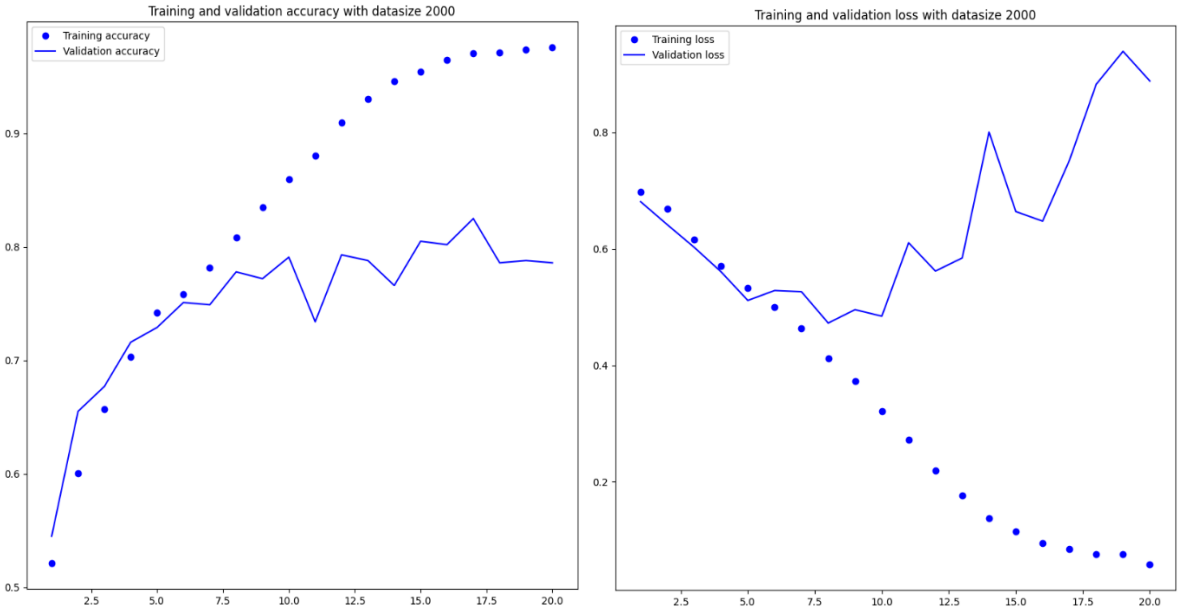
3. Now change your training sample so that you achieve better performance than those from Steps 1 and 2. This sample size may be larger, or smaller than those in the previous steps. The objective is to find the ideal training sample size to get best prediction results?

After comparing the test accuracies of both models as that shows with increase in training sample size the test accuracy increases, we have tried increasing the training sample size to 2000.

The results are as follows:

With Dropout:

	Training	**Validation**	**Test**
Accuracy	97.58%	78.6%	77.1%
Loss	0.0558	0.8881	0.5106



we can say that the training accuracy improves linearly with time, until reaching almost 100%, but the validation accuracy peaks at 78.6%. The validation loss reaches its lowest level after only 15 epochs and then levels, but the training loss decreases linearly as training progresses. From the table, we can say that the test accuracy was 77.1%.

although the accuracy was much better than the remaining models, it's not good enough. This might happen due to overfitting, which was caused by having too few samples to learn from, rendering you unable to train a model that can generalize to new data. So, we have used data augmentation on the same model.

The results are as follows:

With Dropout and Data Augmentation:

	Training	**Validation**	**Test**
Accuracy	82.7%	78.8%	83.2%
Loss	0.3963	0.4856	0.4238

we can say that the training accuracy improves linearly with time, and the validation accuracy peaks at 78.8%. The training and validation losses go on to reduce linearly as training progresses. We might have trained the model with more epochs which might improve the

training accuracy and reduce the losses. From the table, we can say that the test accuracy was 83.2% which was far better than the model without data augmentation and with a training sample size of 1000,1500. Also, we might have trained the model with a greater number of epochs which might have improved both the training and test accuracies of the model.

There is a correlation between test loss and training sample size, which shows that test loss decreases over time and the test accuracy increases, which shows a better improvement over time. Therefore, we can say that the performance of the model increases as the number of training samples increases.

From all these, we can conclude that with an increase in the training sample size, the accuracy of the model increases until a certain sample size. If we train the model beyond that sample limit that may cause overfitting in the model. So, we can say 2000 was an ideal sample size, but we might get better accuracy if we still go beyond that, but until a limit.

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.

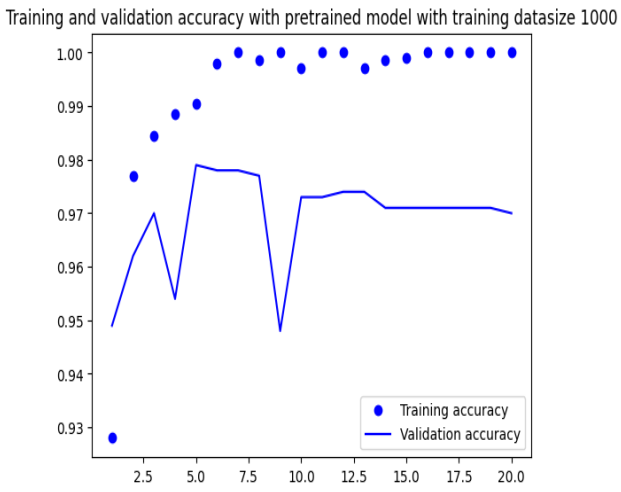
We have used VGG16 architecture as a pretrained network wit three different training data samples sizes.

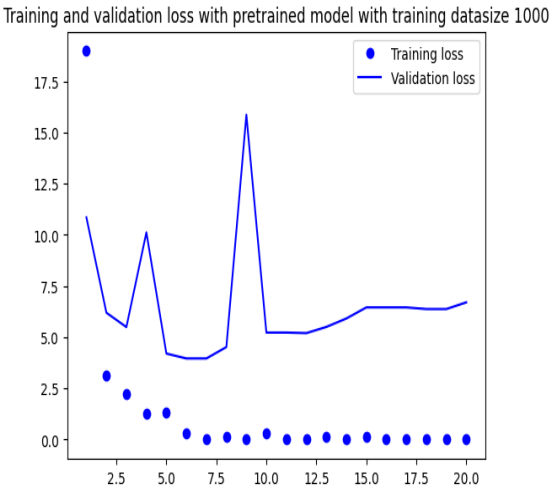
We have used the VGG16 convolution base and classifier with a training sample size of 1000, a validation sample size of 500, and a test sample size of 500. And to reduce overfitting we have used “dropout”.

The results are as follows:

Pretrained model with feature extraction and dropout

	Training	**Validation**	**Test**
Accuracy	100%	97%	97.8%
Loss	2.2333e-07	6.7032	5.0704





We have reached a validation accuracy of about 97% and test accuracy was 97.8% which was much better than what we achieved in training a model from scratch with same sample size. However, our pre-trained model already has the necessary information for the task due to ImageNet's abundance of dog and cat cases, making the comparison unfair. When you employ pre-trained features, this will not always be true.

Despite utilizing a rather high rate of dropout, the plots show that we are rapidly overfitting. This is because this approach does not employ data augmentation, which is necessary for reducing overfitting with tiny picture datasets.

So, we have tried feature extraction together with data augmentation, the results are as follows;
Pretrained model with feature extraction with dropout together with data augmentation

	Training	**Validation**	**Test**
Accuracy	98.35%	97%	97.9%
Loss	0.3404	1.3593	0.6151

we have reached a validation accuracy of over 97%. This is a strong improvement over the previous model without data augmentation. And we got a test accuracy of 97.9%. if we have trained the model we might have achieved much better accuracies. This is only a modest improvement compared to the previous test accuracy. A model's accuracy always depends on the set of samples you evaluate it on. Some sample sets may be more difficult than others, and strong results on one set won't necessarily fully translate to all other sets.

We have frozen some layers in the convolution base and fine-tuned the model, the results are as follows.
Pretrained model with fine-tuning and data augmentation

	Training	**Validation**	**Test**
Accuracy	99.7%	98.2%	98%
Loss	0.0244	0.8839	0.4652

we have reached a validation accuracy of over 98%. This is a strong improvement over the previous model. And we got a test accuracy of 98.2%.

we have tried increasing the training sample size to **1500** and the results are as follows,

Pretrained model with feature extraction and dropout

	Training	**Validation**	**Test**
Accuracy	100%	98.2%	97.2%
Loss	2.1819e-08	4.1543	6.5748

Pretrained model with feature extraction with dropout together with data augmentation

	Training	Validation	Test
Accuracy	98.77%	98.5%	97.2%
Loss	0.1365	0.3203	0.6193

Pretrained model with fine-tuning and data augmentation

	Training	Validation	Test
Accuracy	99.3%	98.5%	97.9%
Loss	0.0587	0.2359	0.6640

we have reached a validation accuracy of over 98.5%. This is a strong improvement over the previous model. And we got a test accuracy of 97.9%.

we have tried increasing the training sample size to **2000** and the results are as follows,

Pretrained model with feature extraction and dropout

	Training	Validation	Test
Accuracy	99.5%	97.6%	97.1%
Loss	0.0195	7.4849	6.1953

Pretrained model with feature extraction with dropout together with data augmentation

	Training	Validation	Test
Accuracy	98.55%	97%	97.3%
Loss	0.0587	0.5889	0.3742

Pretrained model with fine-tuning and data augmentation

	Training	Validation	Test
Accuracy	99.2%	98%	98.2%
Loss	0.0534	0.5076	0. 3892

we have reached a validation accuracy of over 98%. This is a strong improvement over the previous model. And we got a test accuracy of 98.2%.

We found that increasing the training sample size and using a pre-trained network resulted in much better performance. We also found that fine-tuning a pre-trained network on our dataset produced better results than creating a network from the start. The relationship between training sample size and choice of network is as follows: increasing sample size generally improves performance, but network architecture selection is also important. For lower sample sizes,

simpler network architectures could be sufficient, but for bigger sample sizes, complex designs may be required to capture the underlying patterns in the data. Pretrained networks are particularly useful for improving performance when the dataset is small, as they are already trained on large datasets and can capture a wide range of features.