CONVOLUTION REPORT
TARUN KUMAR KORIMI
811299922

Building a convolutional neural network that can effectively and precisely identify photographs of cats and dogs by identifying their unique properties is the goal of this project.
This experiment uses a Kaggle dataset with 25,000 training and 12,500 test photos including equal numbers of cats and dogs.

Problem to be defined:

The Cats-vs-Dogs dataset aims to determine if a picture belongs to the dog or cat class.

Methods

Data:

With a compressed size of 543MB, the Cats-vs-Dogs dataset consists of 25,000 photos that are equally divided between dogs and cats. I downloaded the dataset, unzipped it, and then made a new dataset with three subsets:

- Training dataset with 1000 samples from every class
- Validation dataset with 500 samples
- Test dataset with 500 samples

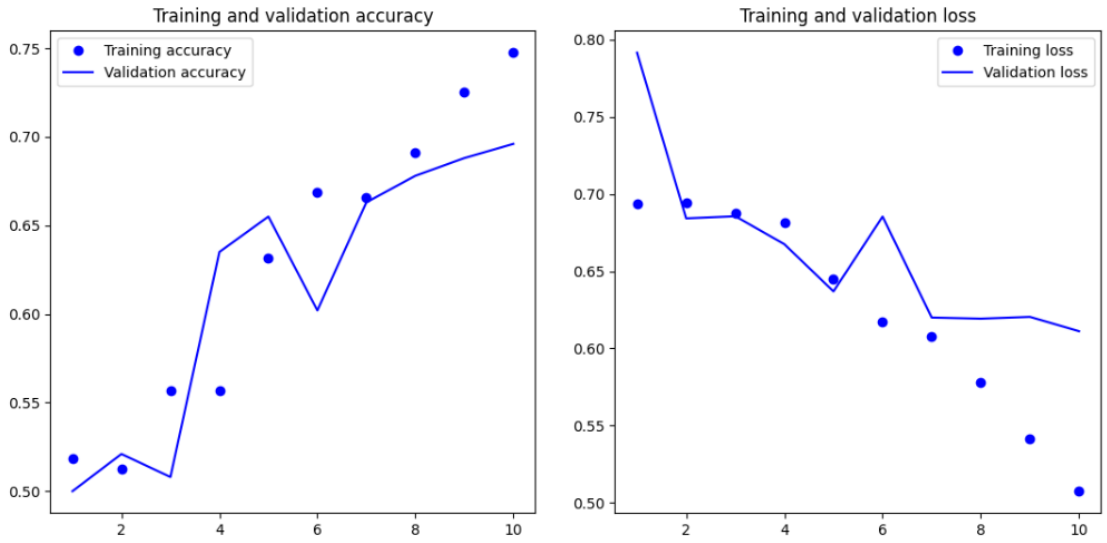
Since the problem we are working on is more complex and calls for a wider image, we must enlarge the size of our neural network. To do this, we will add a step to our present Conv2D + MaxPooling2D architecture**.** In addition to boosting the network's capacity, this makes sure that the feature maps are not excessively huge when we reach the Flatten layer. Initially, the size of our input photos is 150x150. As we ascend the levels of the network, the feature maps grow smaller until they reach 7x7 right before the Flatten layer. Although the chosen input size seems rather random, it works for the current job.

Data Preprocessing:

- Examine the image files.
- Convert the JPEG data into RBG pixel grids.
- Create floating point tensors out of grids.

The pixel values, which range from 0 to 255, should be rescaled to the [0, 1] interval (since neural networks perform best with tiny input values.)

I took batch size as 255 and applied the data flattening technique to convert the data transformation. With the help of 10 epochs, we got to know the validation accuracy as 69% and test accuracy as 69.7%.



From the above result we can conclude that the test accuracy with no data augmentation is about 69.7% when the Training accuracy is about 92%.

Question 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?

Data Augmentation

This approach can be applied to raise the accuracy of the model. One technique that makes it possible to get trustworthy findings even with small datasets is data augmentation. It entails applying random changes to the given training samples to produce new data. This method makes sure the model sees a lot of different types of pictures during training, which improves its capacity for successful generalization.

The training sample of 1500 and the validation test of 500 form the basis of all the results below.
showing the trained augmented pictures

Test Accuracy: 77.6% Better results than the previous (Question 1) are evident from the validation accuracy of 79%, which may be attributed to the following factors:

The model's performance has improved because of the following:
We added 500 (1000–1500) training samples, which helped to increase test and validation accuracy by almost 10%; additionally, we implemented data augmentation in addition to the convolution layer, which helped us improve the featured extractions that led to better performance.

Question 3: 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 objective is to find the ideal training sample size to get best prediction results

- We are unable to determine the appropriate sample size since, as we know, utilizing more and more data will assist to enhance the model's performance.
- This involved the usage of test sets including 500 samples and 2000 training samples with validation. Compared to training samples of 1000 and 2000 photos, I have found that test accuracy is higher with 1500 images.
- When there are 1000 training samples, training accuracy improves.
- Raising the training sample to 2000 while maintaining the 500-sample validation and test sets

Results

Training samples	Validation Accuracy	Test Accuracy	Data Augmentation
1000	69%	69.7%	NO
1500	79%	77.6%	YES
2000	84.1%	83.2%	YES

Question 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.

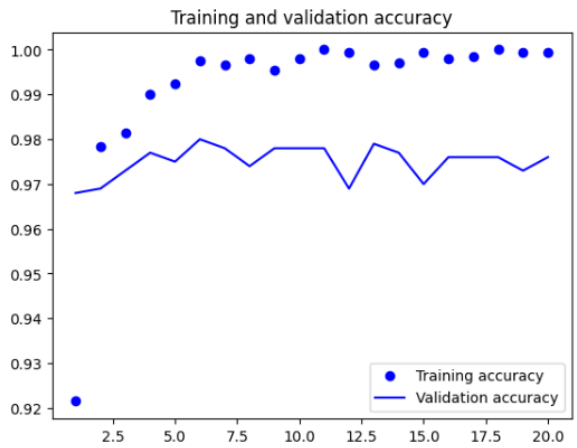
Pre-trained model

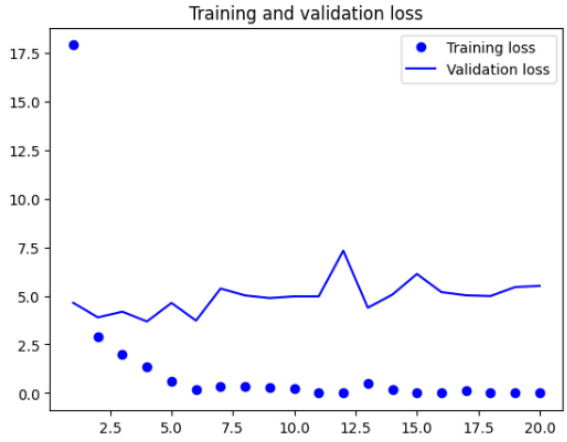
Pretrained networks are mostly used for feature extraction and fine-tuning.
If a pretrained network's initial dataset is big and diverse, it may be utilized as a generic model with its characteristics applied to a variety of computer vision applications. Deep learning's ability to apply learned properties to a variety of tasks is one of its primary benefits over other machine learning techniques. One example of analyzing a large trained convolutional neural network is to use the ImageNet dataset, which has 1.4 million annotated images and 1,000 different classifications. Numerous animal categories are included in the collection, including several dog and cat breeds. The architecture of this network is called VGG16, which is a wellliked and simple convnet design for ImageNet.

In this instance, we will use feature extraction to enhance the results, first without data augmentation and subsequently with data augmentation.

Pre-Trained model without Augmentation

Our validation accuracy is 98% and train accuracy is 99.8%. The charts demonstrate that we are overfitting almost immediately, even if we are using dropout at a rather high rate.





Pre-Trained model with Data Augmentation:

- validation accuracy is 97.8%
- train accuracy is 97.7% • test accuracy: 97.2%.

Fine-tuning a pretrained model

- validation accuracy is 98% • train accuracy is 99.5% • test accuracy: 97.1%.

TABLE FOR PRE-TRAINED MODEL

Data Augmentation	Train Accuracy (%)	Validation Accuracy (%)
NO	99.8%	98%
YES	97.7%	97.8%

CONCLUSION:

We obtained a training accuracy of 92% using a short training set of 1000 samples.
The purpose of data augmentation is for reducing overfitting.

Methods for preventing overfitting:

- Growing the training sample is not always an option. One way to make the most of the limited training data is through data augmentation.
- The number of learnable parameters in the model, or the number of layers and units in layers, is what determines how much overfitting occurs as the model's size is decreased.
- Restricting the weights to only accept very small values helps to regularize the weight values' distribution, avoiding or lessening overfitting and limiting the complexity of a network.
- A good way to lessen overfitting during training is to zero off some of the layer's output attributes. The term "dropout rate" describes the proportion of characteristics that are null.

The tables drawn above provide the model settings and the sample sizes for the train, test, and validation sets. We provide both the results without data augmentation for the initial model, as well as with augmentation for the trained models t with different train and validation sizes or with an increase in train size. For the pre-trained model, we compare the validation accuracy, accuracy, and data augmentation.

A larger training set or a different validation set size both improve the accuracy of the model. Data augmentation did not improve the model's accuracy or validation accuracy when we compared the pre-trained model with and without data augmentation. When limited training data is available, pre-trained models perform better overall than models built from scratch.