Convolution Assignment

- Teja Tarapatla

Overview:

The goal of this project is to create a convolutional neural network that can quickly and effectively classify photos of cats and dogs by recognizing their distinctive characteristics. The dataset used for the experiment was sourced from Kaggle and included 25,000 test and 12,500 training images of the same number of cats and dogs. However, just a portion of the 2000 photos will be used to build the model.

The classification of cats from dogs is one challenge that uses convolutional neural networks (CNNs), which are useful for recognizing and learning visual patterns in images. Convolutions extract crucial information like edges, textures, and shapes to distinguish between identical things, such as the various shapes of fur or the faces of dogs and cats. This approach gives CNNs greater flexibility, allowing them to recognize these features anywhere in the image (spatial invariance). Convolutions also reduce the amount of parameters compared to fully linked layers, which speeds up and enhances the model's memory efficiency during training. By stacking numerous convolutional layers, the model creates a hierarchical comprehension of the data, progressively picking up knowledge from fundamental (like edges) to complex (like fur patterns or face structures). For tasks like cats vs. dogs, where minute distinctions between classes must be recorded for precise classification, this tiered method works incredibly well.

Q1: Consider the Cats & Dogs example. Start initially with a training sample of 1000, a validation sample of 500, and a test sample of 500 (half the sample size as the sample Jupiter notebook on Canvas). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?

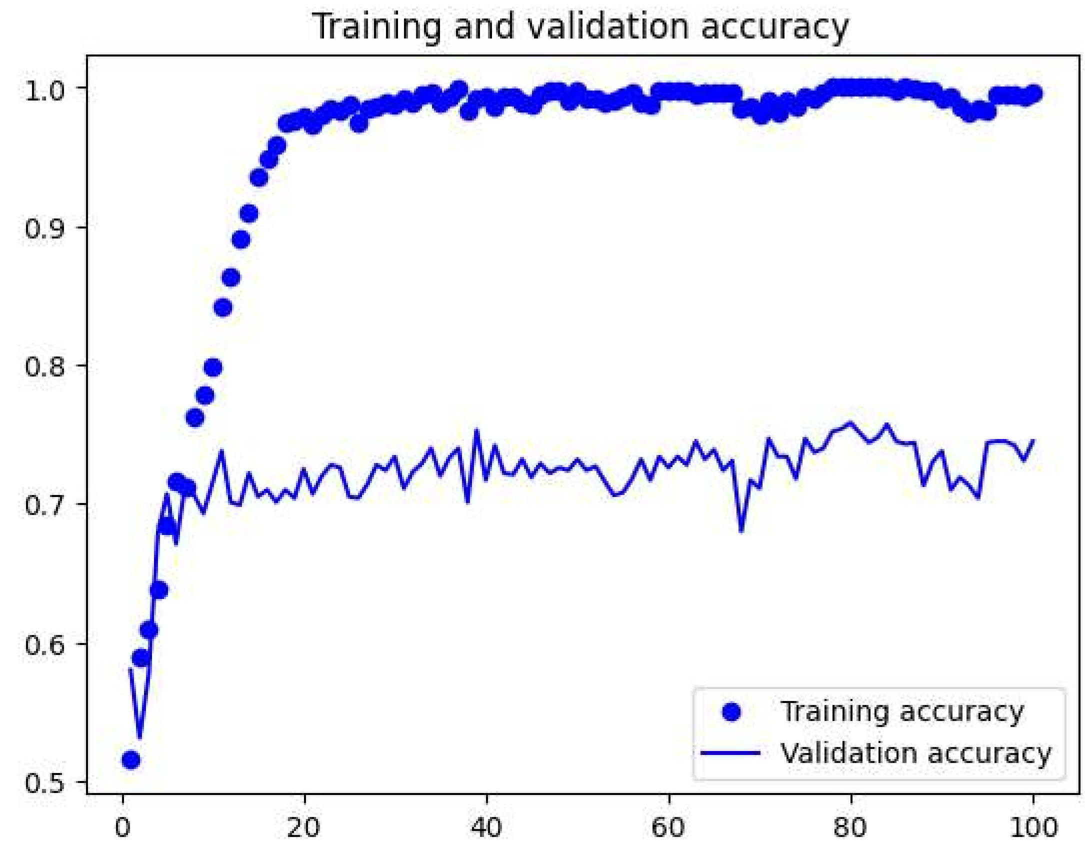
Answer: We used a training sample of 1000 (validation = 500 and test = 500) to analyze the Cats & Dogs data set. To avoid overfitting, which can happen with the provided training sample of 1000, I have employed the 50% dropout technique.

Review the image files.

Use the JPEG data to create RBG pixel grids.

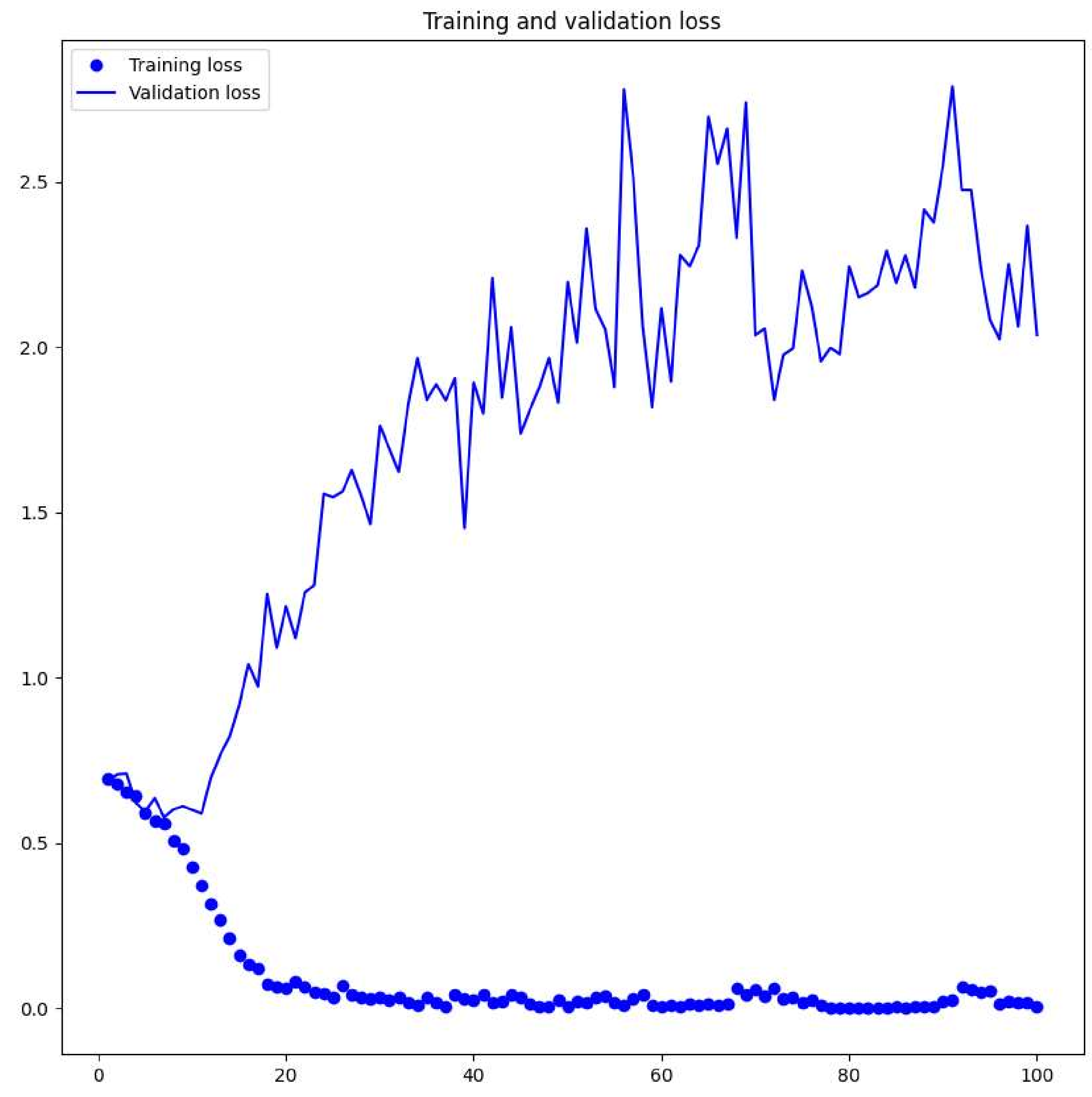
Build floating point tensors from grids.

Because neural networks work best with small input values, the 0–255 pixel values must be rescaled to the [0, 1] interval.



Training and validation accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
20
40
60
80
100
Training accuracy
Validation accuracy

```
<Figure size 640x480 with 0 Axes>
```



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?

Performance of the model was enhanced by increasing the training sample by 500 (1000–1500).

All of the following results are based on the 500-validation test and the 1500 training sample.

DATA AUGMENTATION:

CNNs use a technique known as data augmentation, which creates modified images to artificially expand the training dataset, to enhance model performance and generalization. By enabling the model to learn from a greater range of samples, data augmentation in CNNs increases the model's resistance to changes in real-world images. Some well-liked techniques for data augmentation include the following:

Flipping

Rotation

Scaling and cropping

Translation and brightness

Data augmentation can help with a variety of machine learning tasks, including classifying images, detecting objects, recognizing audio, and processing natural language. Machine learning models, particularly deep learning models, can perform better with the help of this efficient technique. The results from Question 1 were higher for the reasons stated below, as this figure merely illustrates:

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.

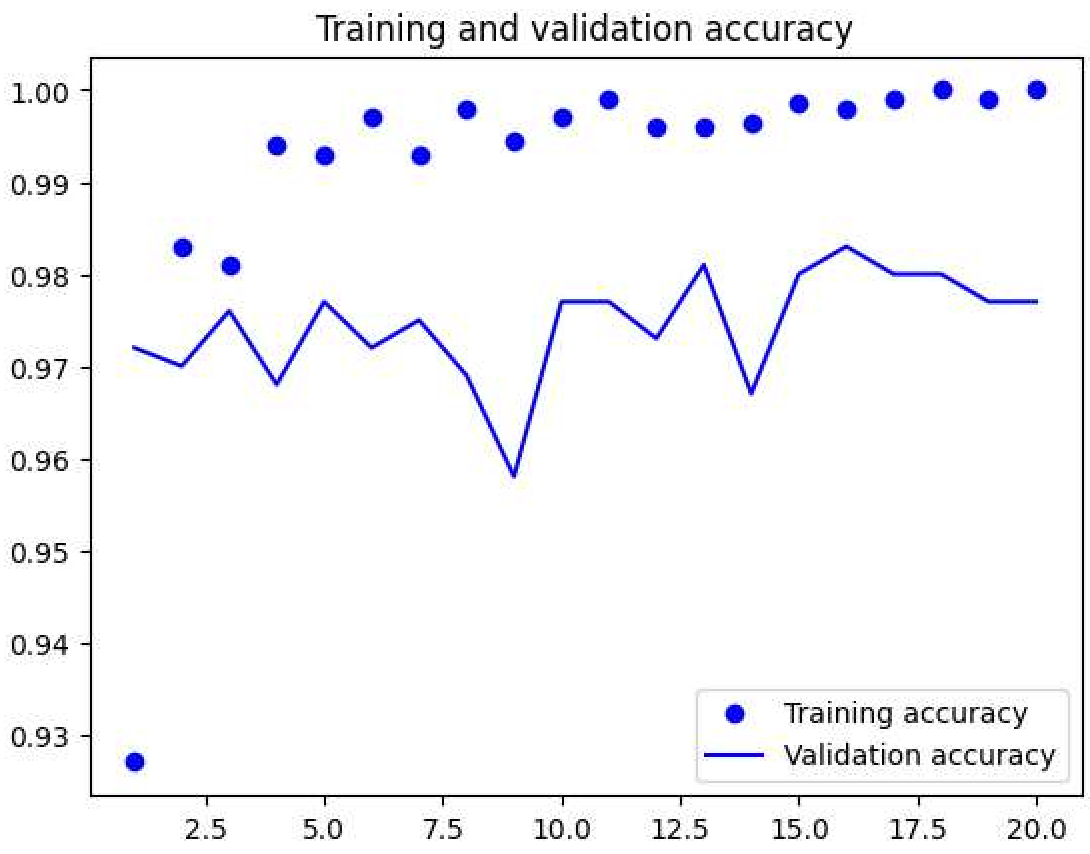
The new sample size is 2,000.

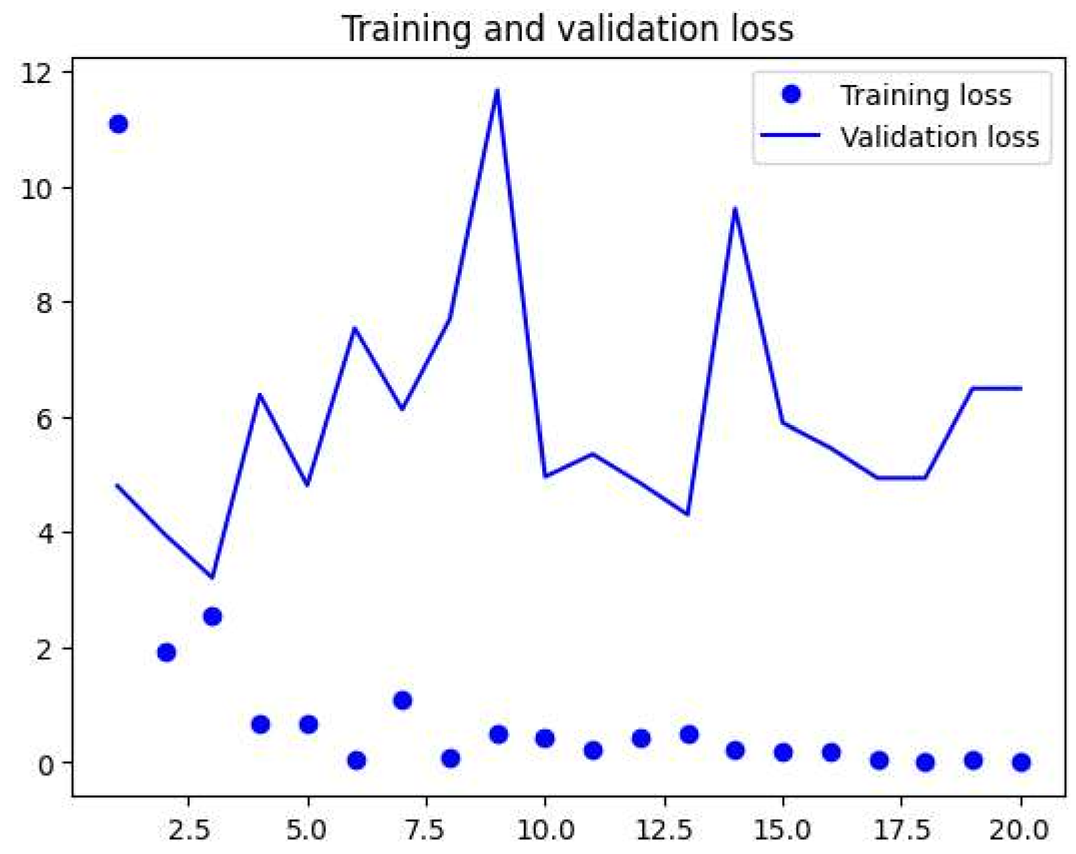
The model will perform better if the data size is increased by 500. By increasing the sample size by 500 and adding new data, we may improve the performance of our model from 97.

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 the best performance.

Pre-trained model without augmentation:

The accuracy we achieved during the initial training phase of the small model is lower than our train accuracy of 98.2%. We have a 96.5% validation accuracy. The plots show that we are fast overfitting even with an average dropout rate.





Pre-trained model with augmented data:

When analyzing a model, samples must always be taken! Strong findings from one sample set might not be applicable to other sets due to the possibility that different sample sets provide unique obstacles.

Results:

Sample	Train Accuracy	Validation Accuracy	Data Augmentation
1000	96.15	85.9	No
1500	95	85.3	Yes
2000	97.56	97.7	Yes
Pre-trained	97.7	97.3	No
Pre-trained	98.2	96.5	Yes
Fine - tuned	97.6	94.3	Yes

Conclusion:

The prior results make it evident that the data drives the model. By increasing the training sample size from 1000 to 2000, test accuracy increases from 74.6 to 85%. Improved accuracy can also be obtained with a trained model and additional data. Overall, we can conclude that our capacity to forecast and generally generalize is enhanced when we use data augmentation with additional data points.