**CONVOLUTION REPORT**

**Likitha Sree Yarabarla**

**Sai Mithra Vadepalli   
(GROUP 26)**

**Introduction:**

The objective of this project is to construct a convolutional neural network (CNN) capable of accurately and precisely distinguishing images of cats and dogs by recognizing their distinct characteristics. The experiment utilizes a dataset from Kaggle comprising 25,000 training images and 12,500 test images, with an equal distribution of cats and dogs.

**Problem statement:**

The Cats-vs-Dogs dataset is designed to classify images into either the dog or cat category.

**Methods used:**

Data:

The Cats-vs-Dogs dataset, totaling 543MB in compressed size, comprises 25,000 images evenly split between cats and dogs. After downloading and extracting the dataset, I created a new dataset containing three subsets:

Training dataset: Consisting of 1000 samples from each class

Validation dataset: Containing 500 samples

Test dataset: Also including 500 samples

Given the complexity of the problem at hand and the need for a broader image, we must enhance the size of our neural network. To achieve this, we incorporate an additional step into our existing Conv2D + MaxPooling2D architecture. This augmentation not only increases the network's capacity but also ensures that the feature maps do not become overly large as we approach the Flatten layer. Initially, the dimensions of our input images are 150x150. As we progress through the network's layers, the size of the feature maps gradually decreases until they reach 7x7 just before the Flatten layer. Although the choice of input size may seem arbitrary, it effectively serves the current task.

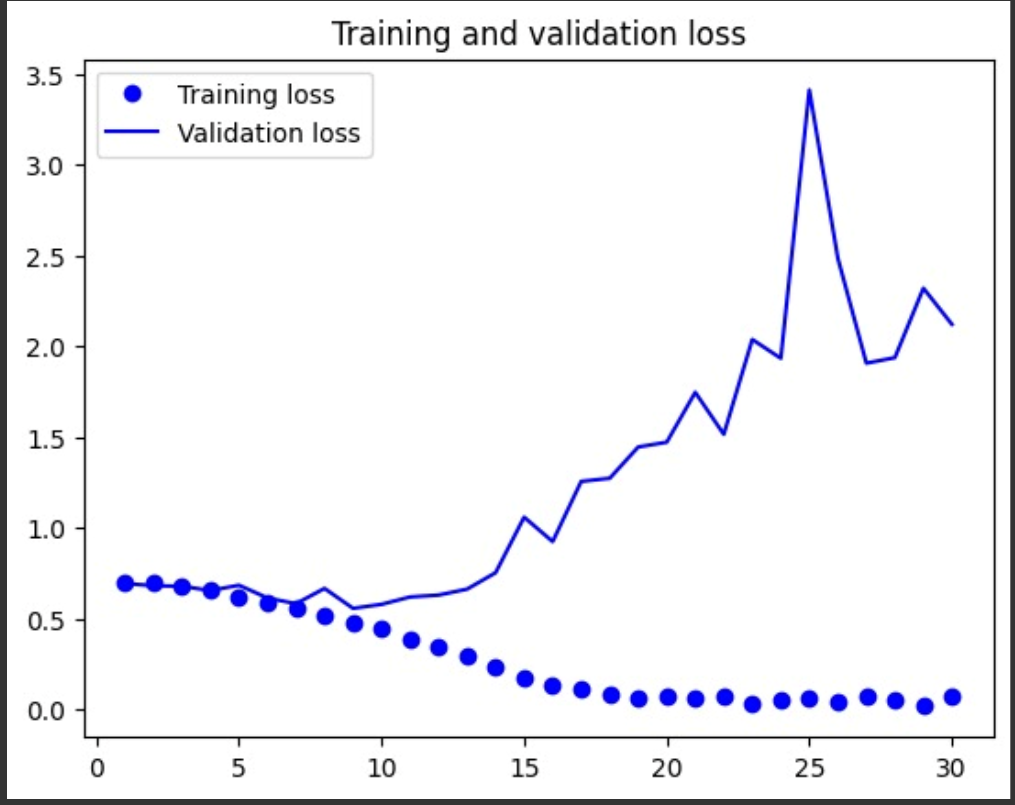
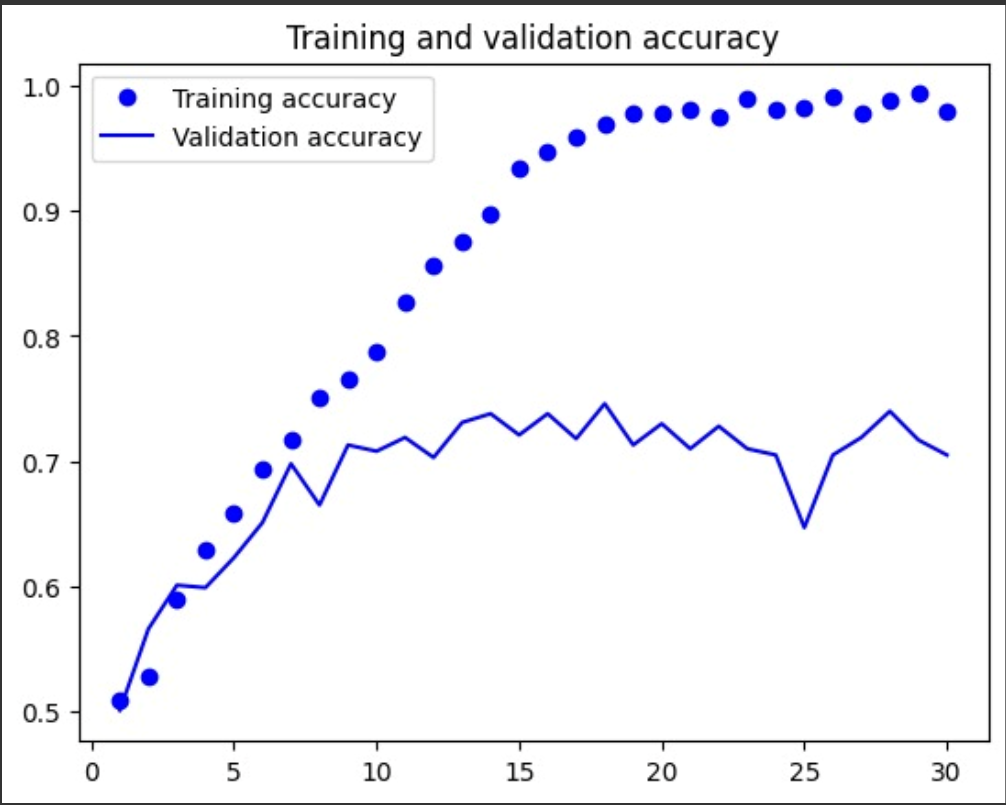
**Data Preprocessing:**

- Inspect the image files.

- Transform the JPEG data into RGB pixel arrays.

- Convert the pixel arrays into floating-point tensors.

The pixel values, ranging from 0 to 255, need to be normalized to fall within the [0, 1] range to optimize neural network performance, as networks tend to perform better with smaller input values.

I chose a batch size of 255 and employed data flattening to convert the data transformation. After running for 30 epochs, we found that the validation accuracy reached 70.5%, while the test accuracy achieved 73.7%.

**Question 2:**

**Data Augmentation:**

Utilizing this strategy can enhance the model's accuracy. Data augmentation is a technique that enables obtaining reliable results even with limited datasets. It involves applying random modifications to the existing training samples to generate additional data. This approach ensures that the model is exposed to a diverse range of images during training, thereby enhancing its ability to generalize effectively.

All subsequent results are based on a training sample of 1500 and a validation set of 500 images.

The trained augmented pictures 

Test accuracy is 80.9% greater than results in question1 and validation accuracy is 80.3%

Possible Reasons for the Improvement in Model Performance:

The enhancement in the model's performance can be attributed to the following factors:

* The addition of 500 (from 1000 to 1500) training samples contributed to nearly a 10% increase in both test and validation accuracy.
* Furthermore, incorporating data augmentation alongside the convolutional layer facilitated improved feature extraction, thereby resulting in enhanced performance.

**Question 3:**

We cannot definitively ascertain the ideal sample size, as it's widely recognized that utilizing larger datasets tends to improve model performance.

* In our experiment, we utilized test sets consisting of 500 samples and 2000 training samples with validation. Upon comparison, we observed that test accuracy is higher when utilizing 1500 images compared to both 1000 and 2000 training samples.
* With 1000 training samples, we noticed an improvement in training accuracy.
* We increased the training sample size to 2000 while keeping the validation and test sets at 500 samples each.

**RESULT :**

|  |  |  |  |
| --- | --- | --- | --- |
| Training samples | Validation Accuracy | Test Accuracy | Data Augmentation |
| 1000 | 70.5 | 73.7 | NO |
| 1500 | 80.3 | 80.9 | YES |
| 2000 | 98.1 | 97.9 | YES |

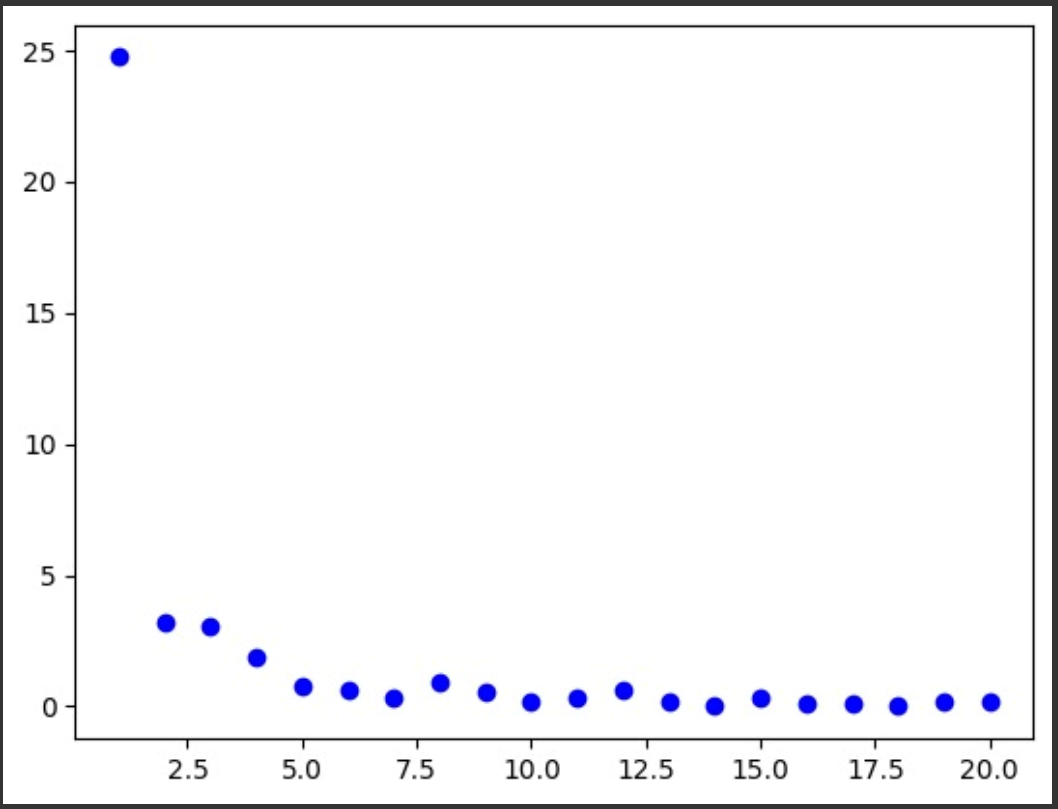
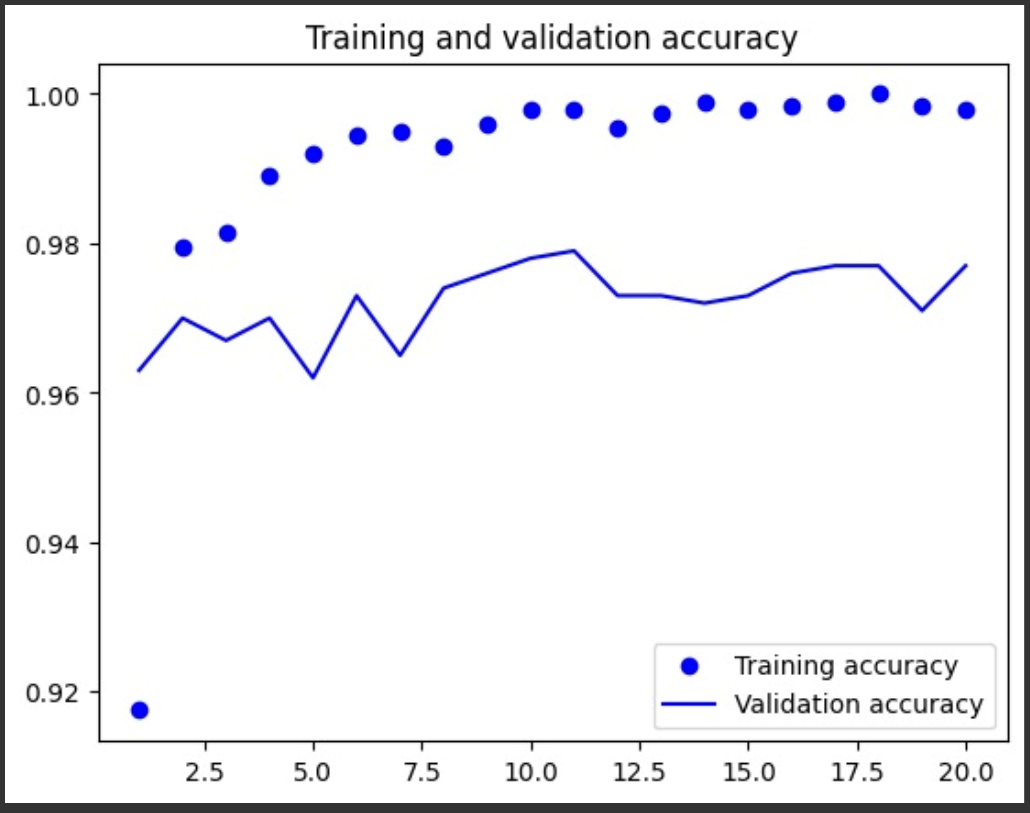
**Question 4:**

**Pre-trained Model:**

Pretrained networks are primarily employed for feature extraction and fine-tuning purposes. When a pretrained network has been trained on a large and diverse dataset, it can serve as a versatile model whose learned features can be applied to various computer vision tasks. One of the key advantages of deep learning is its ability to transfer learned characteristics across different tasks, distinguishing it from other machine learning methods.

An exemplary dataset used for analyzing a pretrained convolutional neural network is ImageNet, which contains 1.4 million annotated images across 1,000 different classes, including numerous animal categories such as various breeds of dogs and cats. One commonly utilized architecture in this context is VGG16, a straightforward convolutional network design tailored for the ImageNet dataset.

In this scenario, we will leverage feature extraction to improve the results, initially without data augmentation and subsequently incorporating data augmentation techniques.



**Pre-Trained model without Data Augmentation:**

• validation accuracy is 97.8%

• train accuracy is 99.8%

**Pre-Trained model with Data Augmentation**:

• validation accuracy is 98.1%

• train accuracy is 98.5%

• test accuracy: 97.9%.

**Fine-tuning a pretrained model**

• validation accuracy is 98.1%

• train accuracy is 99.75%

• test accuracy: 98.0%.

**Conclusion:**

We obtained a training accuracy of 98% using a short training set of 1000

Samples.

**Methods for Mitigating Overfitting:**

* Expanding the training dataset may not always be feasible. One approach to maximizing the utility of limited training data is through data augmentation.
* The extent of overfitting is influenced by the number of learnable parameters in the model, including the number of layers and units within those layers.
* Regularizing the distribution of weights by constraining them to very small values can help prevent or mitigate overfitting, thereby limiting the complexity of the network.
* Introducing dropout during training is an effective method for reducing overfitting. Dropout involves randomly setting a proportion of the layer's output features to zero, with the dropout rate indicating the fraction of features that are zeroed out.

The previously presented tables outline the model configurations and sample sizes for the training, testing, and validation datasets. We present results both without data augmentation for the initial model and with augmentation for models trained with varying sizes of training and validation sets, or with an increase in the training size. Additionally, for the pretrained model, we compare validation accuracy, overall accuracy, and the impact of data augmentation.

Increasing the training set size or adjusting the validation set size both lead to improvements in the model's accuracy. However, when comparing the pretrained model with and without data augmentation, we observed no enhancement in either accuracy or validation accuracy through augmentation. Notably, in scenarios where training data is limited, pretrained models tend to outperform models constructed from scratch.