**ADVANCED MACHINE LEARNING**

**Assignment 2- Convolution**

**Name: Vijay Charan Reddy Gottam**

**Student ID: 811292996**

**Summary**

We are developing a novel convolutional neural network especially for computer vision applications. We are using the "Dog-vs-Cats" dataset from Kaggle, but its small size presents a problem. The ability of convolutional neural networks, or convnets, to recognise and comprehend patterns in the spatial arrangement of images makes them well-known for their superior performance in computer vision. This makes them ideal for tasks like object detection, object classification, and segmentation—that is, breaking up an image into its component parts.

We think our convnet model has a chance to deliver good outcomes even with limited data availability. Because they are skilled at identifying the crucial details in photos, convolutional neural networks (convnets) are renowned for their capacity to learn from and apply their knowledge to new situations, even with small amounts of data. Our strategy is to use the available data to train the model, then apply transfer learning to further refine it, and lastly assess the model's performance using targeted metrics. In essence, we want to create a convolutional neural network (convnet) that uses the least amount of data possible to accurately sort images in the "Dog-vs-Cats" dataset.

**Problem**

Deciding if an image belongs in the dog or cat category is the goal of the Cats-vs-Dogs dataset binary classification task.

**Techniques**

**Dataset**

There are 25,000 photos of dogs and cats in the Cats-vs-Dogs dataset (12,500 from each class). Three subsets will make up the new dataset we're assembling: a test set with 500 samples per class, a validation set with 500 samples per class, and a training set with 1000 samples per class. All of the samples have been downloaded and uncompressed. Our neural network must be larger because the issue we are trying to solve is more complex and necessitates a wider viewpoint. We're adding a stage to our current Conv2D + MaxPooling2D setup to handle the increased complexity of our problem. This change improves the network's capacity and helps regulate feature map sizes as we get closer to the Flatten layer. Our input images are 150x150 at first, and the feature maps gradually get smaller as we go through the network layers, reaching 7x7 prior to the Flatten layer. The input size that was selected seems a little arbitrary, but it works well for the task at hand.

**Preprocessing:**

Access the image files.

Decode the JPEG content into RGB pixel grids.

Transform them into floating-point tensors.

In order to get the pixel values (which range from 0 to 255) to lie inside the [0, 1] range, scale them (since smaller input values are better for neural networks).

**Data Augmentation:**

Our goal is to apply data augmentation techniques to increase the accuracy of our model. Data augmentation creates new data from preexisting training samples by adding random variations, allowing us to get good results even with small datasets. This method makes sure the model sees different versions of the images it hasn't seen before during training, which enhances generalisation. To achieve our particular goal, we plan to haphazardly apply various transformations, including flipping, rotating, and zooming, to the images in the training set. This procedure produces a variety of original image versions, enhancing the diversity of the dataset and bolstering the robustness of our model.

**Pre-trained model:**

This dataset includes many animal classifications, such as different breeds of dogs and cats. One well-known and simple convnet design for ImageNet is VGG16, which is an example of this type of network architecture.

If the original dataset is large and diverse, a pretrained network can be used as a generic model and its features applied to a variety of computer vision applications. The ability of deep learning to transfer learned characteristics across different tasks is one of its primary advantages over other machine learning techniques. A large-scale trained convolutional neural network can be examined as an example using the ImageNet dataset, which contains 1,000 different classes and 1.4 million annotated images.

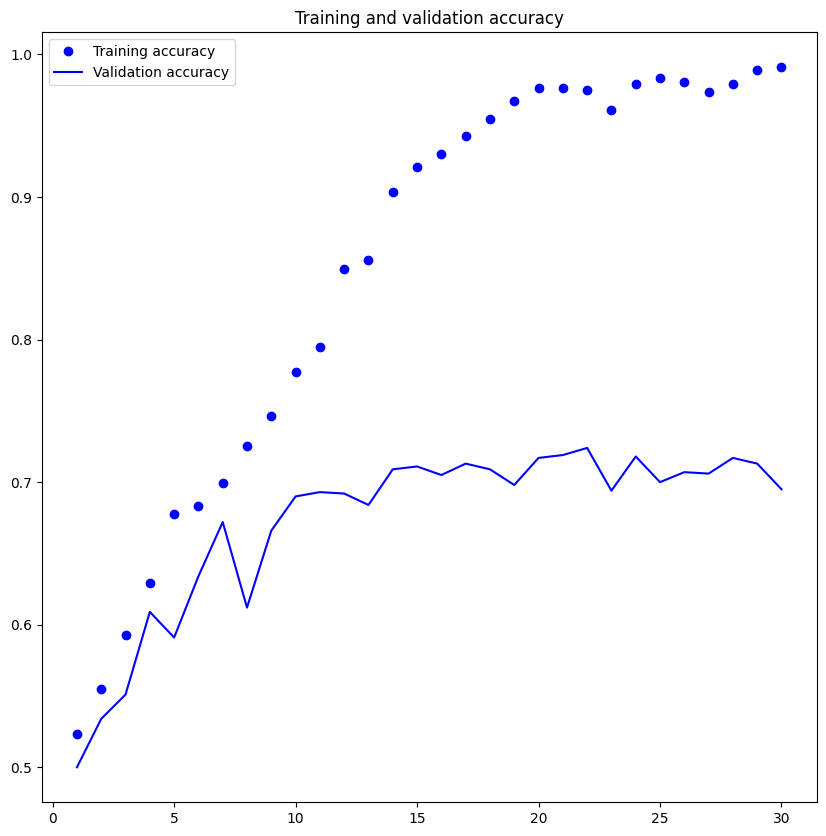
Feature extraction and fine-tuning are the two main methods for applying a pretrained network. We'll concentrate on feature extraction in this instance to improve the outcomes. First, we will extract features without augmentation of data, and then we will add data.

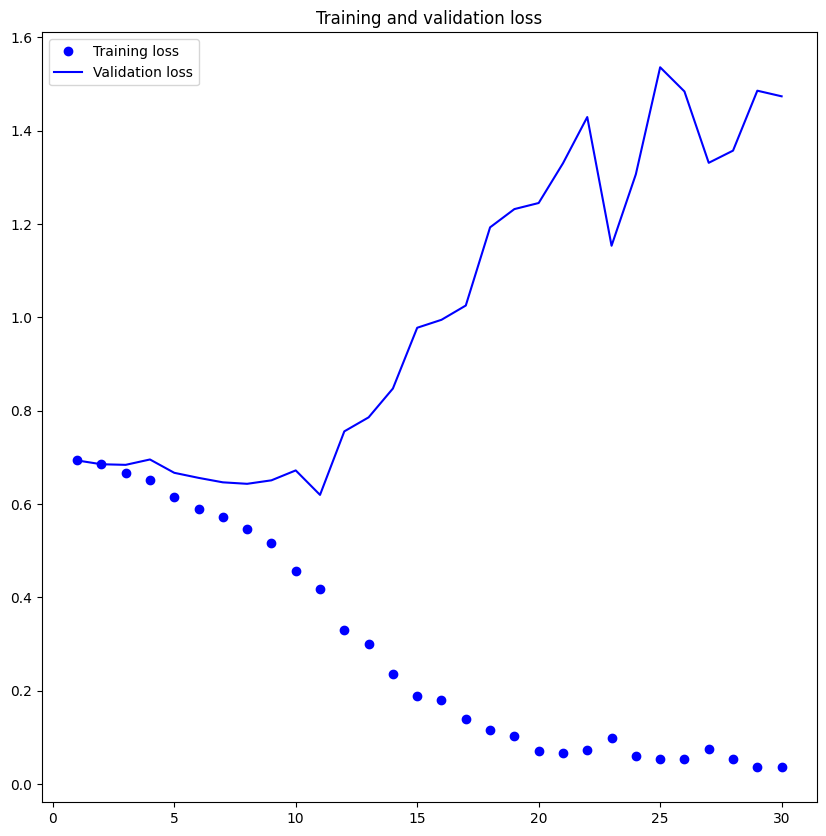
**Question 1: Consider the Cats & Dogs example. Start initially with a training sample of 1000, a validation sample of 500, and a test sample of 500 (half the sample size as the sample Jupiter notebook on Canvas). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?**

The training sample of 1000 (validation = 500 and test = 500) was taken into consideration for the Cats & Dogs Data Set. Given that the training sample size of 1000 has a tendency to be overfit, I have employed a 50% dropout strategy to address this problem.

**Hypertuning parameters:**

We've converted the data transformation using the data flattening technique, and I've set the batch size at 255. We were able to ascertain that the validation accuracy was 69.5 and the test accuracy was 67.3.





**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?**

The results are:

Validation accuracy: 81.4

Test accuracy: 81.7

The results demonstrate that they were superior to the previous for the reasons mentioned below (Question 1)

Our 500 (1000–1500) training sample increase has improved the model's performance. The train and validation accuracy have both increased by more than 10%, as can be seen. In addition to the convolution layer, we also used data augmentation, which helped us improve the featured extractions and achieve 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.**

While increasing the amount of training data is a tried-and-true method of improving model performance, figuring out the right sample size can be difficult.

In this instance, utilising data augmentation techniques and adding 500 samples to the data set produced a notably improved model performance, which increased from 81.8% to 80.1%

Despite the enhanced data and larger sample size inside the specified convolutional architecture, the model shows a restricted ability to acquire new information, which seems to be an obvious example of this phenomena.

This finding raises the possibility that other approaches to improving the model's performance should be investigated.

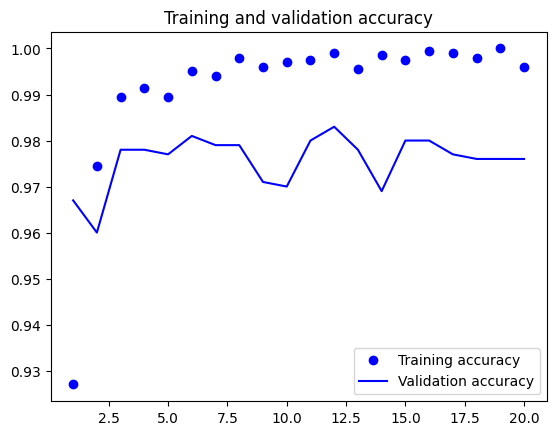
**Question 4: Repeat Steps 1-3, but now using a pretrained network. The sample sizes you use in Steps 2 and 3 for the pretrained network may be the same or different from those using the network where you trained from scratch. Again, use any and all optimization techniques to get best performance.**

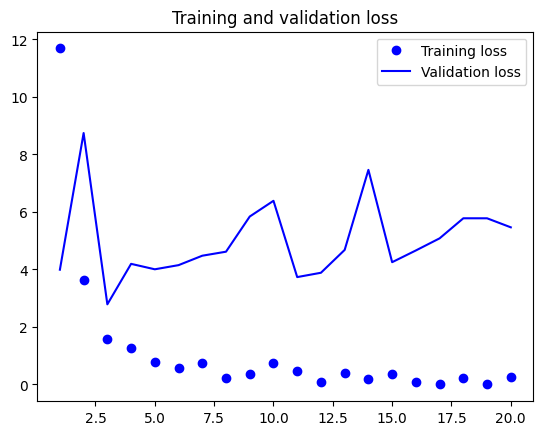
Pre-Trained model without Augmentation

**The model achieved a validation accuracy of 98.50% and a test accuracy of 97.5%. While the test accuracy is encouraging compared to the initial training of a smaller model, there is a concerning trend of overfitting.**

Plots illustrate this overfitting even with dropout regularisation applied at a relatively high dropout rate.

Although the dropout plots, which suggest overfitting is occurring early in the training process, suggest that the T model may not generalise well to unseen data, it is performing well on the validation data (data used to fine-tune hyperparameters).





Pre-Trained model with Data Augmentation:Pre-Trained model with Data Augmentation:Pre-Trained model with Data Augmentation:A graph of training and validation

Description automatically generatedTop of Form

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

The data that is used to evaluate a model must be carefully chosen. Given the varying degrees of complexity present in each dataset, good results on one sample set may not translate to other datasets in general.

The accuracy of the pre-trained model, which was 98.5% without data augmentation and 98.3% with data augmentation, is used to illustrate this.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Training Samples** | **Validation Accuracy** | **Test Accuracy** |
| Model 1 | 1000 | 69.5 | 67.3 |
| Model 2 | 1500 | 81.4 | 81.7 |
| Model 3 | 2000 | 81.8 | 80.1 |
| Model 4 | Pretrained Model without data augmentation | 98.5 | 97.5 |
| Model 4 | Pretrained Model with data augmentation | 98.3 | 97.3 |

**Conclusion:**

The study investigates the effects of data augmentation methods, validation set size, and training data size on the performance of pre-trained and scratch-built models. The main conclusions are as follows:

Accuracy can be raised by expanding the training set or reducing the size of the validation set. This holds true for both pre-trained and scratch models.

Accuracy was not appreciably increased by data augmentation for either model type.

In general, pre-trained models perform better than scratch models, particularly when there is little data. This is as a result of their utilisation of prior task knowledge.