**ADVANCED MACHINE LEARNING**

**Assignment 2- Convolution**

**Names: Nanaji Chalamalashetty &**

**Akhil Kornala**

**Student IDs: 811295529 & 811284142**

**Summary**

Our new convolutional neural network is being developed specifically for computer vision applications. The "Dog-vs-Cats" dataset from Kaggle is what we're using, but there's an issue with its tiny size. Convolutional neural networks, or convnets, are well-known for their superior performance in computer vision because of their capacity to identify and understand patterns in the spatial arrangement of images. This makes them perfect for tasks like segmentation, which is the process of dissecting an image into its constituent parts, and object detection and classification.

We believe that our convolutional neural network model can still produce good results with limited data available. Convolutional neural networks, or convnets, are well known for their ability to learn from and apply their knowledge to new situations, even with small amounts of data, because they are adept at identifying the important details in photos. Our plan is to train the model on the available data, then use transfer learning to make further refinements, and finally use targeted metrics to evaluate the model's performance. Essentially, we aim to build a convolutional neural network (convnet) that correctly sorts images in the "Dog-vs-Cats" dataset using the least amount of data possible.

**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**

The Cats-vs-Dogs dataset contains 25,000 images of dogs and cats (12,500 from each class). The new dataset we're putting together will consist of three subsets: a training set with 1000 samples per class, a validation set with 500 samples per class, and a test set with 500 samples each class. After being downloaded, each sample has been uncompressed. Because the problem we are attempting to solve is more complex and demands a wider viewpoint, our neural network must be larger. Conv2D + MaxPooling2D is our current setup; to handle the increased complexity of our problem, we're adding a stage. As we approach the Flatten layer, this modification increases the network's capacity and aids in controlling feature map sizes. The feature maps in our input images start out at 150x150 and progressively shrink as we proceed through the network layers, arriving at 7x7 just before the Flatten layer. Although the chosen input size appears somewhat arbitrary, it is effective for the task at hand.

**Preprocessing:**

Access the image files.

Decode the JPEG content into RGB pixel grids.

Transform them into floating-point tensors.

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

**Data Augmentation:**

Our objective is to use data augmentation methods to improve our model's accuracy. With data augmentation, we can obtain good results even with small datasets by adding random variations to preexisting training samples to create new data. This technique improves generalisation by ensuring the model sees variations of the images it hasn't seen previously during training. We intend to randomly apply different transformations, such as flipping, rotating, and zooming, to the training set images in order to accomplish our specific goal. This process yields multiple original image versions, which improves the dataset's diversity and strengthens our model's resilience.

**Pre-trained model:**

Numerous animal classifications, including various dog and cat breeds, are included in this dataset. VGG16 is a well-known and straightforward convnet design for ImageNet that serves as an illustration of this kind of network architecture.

A pretrained network can be used as a generic model and its features applied to a range of computer vision applications if the original dataset is large and diverse. One of deep learning's main advantages over other machine learning methods is its capacity to transfer learned characteristics to new tasks. Using the 1,000 different classes and 1.4 million annotated images in the ImageNet dataset, one can investigate a large-scale trained convolutional neural network as an example.

The two primary techniques for using a pretrained network are feature extraction and fine-tuning. In this case, we'll focus on feature extraction to enhance the results. Prior to adding data, we will first extract features without altering the original 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 32. We were able to ascertain that the validation accuracy was 77 and the test accuracy was 72.

A graph of a line graph

Description automatically generatedA graph of a graph

Description automatically generated

**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: 80

Test accuracy: 81

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 81.5% to 80.2%

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% and a test accuracy of 98.1%. 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).

A graph of blue dots

Description automatically generated

A graph of training and validation

Description automatically generated

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:**

A model's evaluation data set needs to be carefully selected. Because every dataset has a different level of complexity, findings from one sample set might not apply to other datasets in general.

The accuracy of the pre-trained model, which was 98% without data augmentation and 97% with data augmentation, is used to illustrate this.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Training Samples** | **Validation Accuracy** | **Test Accuracy** |
| Model 1 | 1000 | 77 | 72 |
| Model 2 | 1500 | 80 | 81 |
| Model 3 | 2000 | 80.5 | 81.2 |
| Model 4 | Pretrained Model without data augmentation | 98 | 98.1 |
| Model 4 (Final Model) | Pretrained Model with data augmentation | 98 | 97 |

**Conclusion:**

The performance of pre-trained and scratch-built models is examined in relation to data augmentation techniques, validation set size, and training data size. The following are the primary conclusions:

Reducing the size of the validation set or increasing the size of the training set will increase accuracy. This is valid for both scratch and pre-trained models.   
  
Data augmentation did not significantly improve accuracy for either model type.   
  
Pre-trained models generally outperform scratch models, especially in situations with limited data. Their use of previous task knowledge is the cause of this.