**ADVANCED MACHINE LEARNING**

**CONVOLUTION**

**ASSIGNMENT-2**

**Rishitha Rapuri**

**811283595**

**Summary**

Our project is creating a new convolutional neural network specifically designed for computer vision tasks. We're utilizing the "Dog-vs-Cats" dataset from Kaggle, which poses a challenge due to its limited size. Convolutional neural networks, or convnets, are known to be good at computer vision because they can find and understand patterns in how spaces are arranged within images. This makes them great for things like separating different parts of an image (segmentation), finding specific objects (detection), and figuring out what's in the image (classification).

We believe that despite the restricted data availability, our convnet model has the potential to produce satisfactory results. Convnets are known for being able to learn from and apply their knowledge to new situations even with small amounts of data, because they're good at picking out the important details in images. Our plan is to train the model on the data we have, then improve it further using a technique called transfer learning, and finally check how well it performs with specific measurements. We basically want to build a convnet that can accurately sort images in the "Dog-vs-Cats" dataset, all while using as little data as possible.

**Problem**

The aim of the Cats-vs-Dogs dataset binary classification task is to ascertain whether an image falls into the category of either dogs or cats.

**Techniques**

**Dataset**

The Cats-vs-Dogs dataset, includes 25,000 images of dogs and cats (12,500 from each class). The new dataset we're assembling will consist of three subsets: a training set containing 1000 samples per class, a validation set containing 500 samples per class, and a test set containing 500 samples per class, all downloaded and uncompressed. We need to increase the size of our neural network because the problem we're tackling is more intricate and demands a broader perspective. To meet the increased complexity of our problem, we're expanding our current Conv2D + MaxPooling2D setup by adding another stage. This modification helps control the size of feature maps as we approach the Flatten layer, while also enhancing the network's capacity. Our input images are initially 150x150 in size, and as we traverse the network layers, the feature maps progressively shrink until they reach 7x7 before the Flatten layer. Although the chosen input size may seem somewhat arbitrary, it suits the task at hand effectively.

**For preprocessing:**

Access the image files.

Decode the JPEG content into RGB pixel grids.

Transform them into floating-point tensors.

Scale the pixel values (ranging from 0 to 255) to fall within the [0, 1] range (since neural networks perform better with smaller input values).

**Data Augmentation:**

To improve our model's accuracy, we aim to utilize data augmentation techniques. By introducing random variations, data augmentation generates additional data from existing training samples, enabling us to achieve strong results even with limited datasets. This approach ensures that during training, the model encounters variations of the images it hasn't seen before, thus improving its ability to generalize. Our intention is to randomly apply transformations to the images in the training set, such as flipping, rotating, and zooming, to serve our specific objective. This process generates diverse versions of the original images, enriching the dataset's diversity and strengthening our model's resilience.

**Pre-trained model:**

This dataset encompasses numerous animal classifications, including various dog and cat breeds. An example of such a network architecture is VGG16, a popular and straightforward convnet design for ImageNet.

A pretrained network may be used as a generic model and its features used to a wide range of computer vision applications if the original dataset is large and diverse. One of the main advantages of deep learning over other machine learning methods is its capacity to transfer learnt characteristics across various tasks. Using the ImageNet dataset, which has 1.4 million annotated pictures and 1,000 distinct classes, a large convolutional neural network that has been trained may be analyzed as an example.

There are two primary approaches to utilizing a pretrained network: feature extraction and fine-tuning. In this case, we'll focus on feature extraction to enhance the results. Initially, we'll perform feature extraction without data augmentation, followed by incorporating 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.

Hyper tuning parameters:

I've set the batch size at 255, and we've converted the data transformation using the data flattening approach. We were able to determine the test accuracy to be 72.22 and the validation accuracy to be 72.7.

A graph with blue dots

Description automatically generated

A graph with blue lines and dots

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: 71.10

Test accuracy: 73.4

For the reasons listed below, the findings show that they were better than the prior (Question 1)

The model's performance has improved as a result of our 500 (1000–1500) training sample increase. As we can see, the train and validation accuracy have increased by over 10%. Additionally, we used data augmentation in addition to the convolution layer, which allowed us enhance the featured extractions and produce higher 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 volume of training data is a well-established approach to enhancing model performance, determining an optimal sample size can be challenging.

In this case, using data augmentation techniques and increasing the data quantity by 500 samples resulted in a clearly better model performance, going from 68.8% to 70.1%.

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 observation suggests the potential need to explore alternative strategies for optimizing the model's performance.

**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 98.0% and a test accuracy of 96.8%. While the test accuracy is encouraging compared to the initial training of a smaller model, there is a concerning trend of overfitting.**

This overfitting is evident despite employing dropout regularization with a relatively high dropout rate, as visualized in the plots.

T model is performing well on the validation data (data used to fine-tune hyperparameters) but may not generalize well to unseen data, as indicated by the dropout plots suggesting overfitting is occurring early in the training process.

A graph of a training and validation accuracy

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:

It is crucial to carefully choose the data that is used to assess a model. Good outcomes on one sample set might not apply to other datasets in general, especially when taking into account the variable degrees of complexity that each dataset has.

This is demonstrated by comparing the accuracy of the pre-trained model, which was 98.6% without data augmentation and 97.7% with data augmentation.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Training samples | Validation accuracy | Test accuracy |
| Model 1 | 1000 | 72.1 | 72.7 |
| Model 2 | 1500 | 71.1 | 73.4 |
| Model 3 | 2000 | 68.8 | 70.1 |
| Model 4 | Pretrained Model without data augmentation | 98.0 | 96.8 |
| Model 4 | Pretrained Model with data augmentation | 98.6 | 97.7 |

**Conclusion:**

The report explores how training data size, validation set size, and data augmentation techniques affect the performance of scratch-built and pre-trained models. Here are the key takeaways:

Increasing training data or adjusting validation set size can improve accuracy. This applies to both scratch and pre-trained models.

Data augmentation didn't significantly improve accuracy for either model type.

Pre-trained models generally outperform scratch models, especially with limited data. This is because they leverage knowledge from previous tasks.