**ADVANCED MACHINE LEARNING**

**Assignment 2- Convolution**

**Raju Rishikesh Gollapally**

**Student ID:811285826**

**Sai Sharanya Devambatla**

**Student ID:811305424**

**Summary**

We plan to build a neural network with convolution methods using which computer vision practices can be carried out. Our dataset of origin is the "Dog-vs-Cats" one obtained from Kaggle, with the only downside being its micro size. This ability is one of the core reasons convolutional neural networks (CNNs) are so often the leading choice to model tasks in the computer vision domain because of their feature of distinguishing and understanding patterns in the structural form of pictures. For these reasons, they are supreme at tasks such as segregation, which encompasses the division of the scene into major parts, or object recognition and categorization as well.

We believe that our convolutional neural network model could probably generate positive outcomes despite the small amount of data ready. Convnets which are the most successful in photo recognition can generalize on relatively low data capacity. Therefore, they become known for understanding feature relevance and for using their knowledge in an unfamiliar situation. The model that is built is trained with the data provided and then performance is evaluated using a few picked indicators after transfer learning is used to strengthen the model. It is the development of a convnet that makes the best possible use of the least amount of input and correctly combines images in the "Dog-vs-Cats" dataset that is the main objective here.

**Problem**

Determining if a photograph belongs in the dog or cat category is the aim of the Cats-vs-Dogs dataset binary classification task.

**Techniques**

**Dataset**

The Cats-vs-Dogs dataset consists of 25000 images of dogs and cats made up of 12500 images of each class. We will be creating a new dataset consisting of three subsets: training data with 1000 samples per class, validation data with 500 samples per class, and a test data with 500 samples per class. Users can download and unzip the files and all the samples are accessible. The complexity of our problem is greater, and our knowledge base must scale accordingly. To handle the added problem complexity, we insert a layer to our current Conv2D + MaxPooling2D architecture design. By the time we get to the Flatten layer, this modification works to govern feature map sizes and increase the network's strength. On the way through the network layers, the feature maps are sequentially shrunk 50x50 and 7x7 before being flattened in the final Flatten layer. Being a bit strange but that will work out for the purpose in mind.  
**Preprocessing:**

Open the image files.

Turn the pictures into a grid of colored dots.

Change these grids into numbers that computers can understand.

Make these numbers easier for computers to use by adjusting them so they're between 0 and 1 instead of 0 to32, which is what they start as.

**Data Augmentation:**

It has many animal classifications in it, for instance various breeds of dogs and cats. There is one such well-known and simple convnet design called VGG16 and it belongs to this type of network architecture.

Furthermore, if the original dataset size is large and contains many classes, then a pretrained model could be used as a general model and transferred to different computer vision projects. The biggest benefit of deep learning among other machine learning techniques is the capability of effective transfer learning across different tasks. An enormous trained convolutional neural network can be thoroughly scrutinized based on the ImageNet dataset, which consists of 1,000 different classes as well as 1.4 million annotated images.

Feature extraction and fine-tuning are the two core principles of a generic network. In this case, we'll focus on the process of feature extraction in order to enhance accuracy. First, we will extract features directly from the data without enhancements, and then we will add data.  
  
**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 (like in the text). Use any technique to reduce overfitting and improve performance in developing a network that you train from scratch. What performance did you achieve?**

For Cats & Dogs Data Set, we used a training sample of 1000 (validation = 500 and test = 500) during the sampling process. As the training dataset is a priori artificially created with the size of 1000 and may exhibit overfitting tendency, I have adopted a 50% dropout to deal with the issue.

**Hyper tuning parameters:**

With the data flattening method, the data transformation is done, and the batch size is 32 is set. From this, we noticed the validation accuracy was 70.4 and the test accuracy was 71.3.

A graph with blue dots

Description automatically generated

A graph of a line graph

Description automatically generated with medium confidence

**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: 85.2

Test accuracy: 84.8

The results show that they were better than before, and I'll explain why (Question 1).

Adding 500 more training samples, bringing the total to between 1000 and 1500, made our model work better. Both the accuracy during training and when checking against new data went up by over 10%. We also made the pictures change a bit to help the model learn better, and that made a big difference too.

**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.**

Selecting the appropriate sample size may be challenging, even while increasing the quantity of training data is a tried-and-true way to improve model performance.

In this case, the model performance increased from 81.8% to 81,5% after data augmentation techniques were applied and 500 samples were added to the data set.   
An apparent illustration of this phenomenon appears to be the model's limited capacity to learn new information, even with the improved data and greater sample size inside the designated convolutional architecture.

This finding implies that more approaches to improving the usefulness of the model need to 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's test accuracy was 97.3 percent, while its validation accuracy was 97.1 percent. Compared to the first training of a smaller model, the test accuracy is positive, but there is a worrying tendency of overfitting.

Plots show this overfitting even with dropout regularization carried out at a respectably high dropout rate.

Despite the dropout plots indicating that overfitting is occurring early in the training phase and perhaps impeding the model's capacity to generalize to new data, the T model is doing well on the validation data (data used to fine-tune hyperparameters).

A graph of a line graph

Description automatically generated with medium confidence

A graph of a line graph

Description automatically generated with medium confidence

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

The evaluation data set for a model must be carefully chosen. The intricacy of each dataset varies, therefore conclusions drawn from one sample set may not generalize to other datasets.

This is demonstrated by the pre-trained model's accuracy, which was 99.7% without data augmentation and 96.9% after data augmentation.

***Table for Model from Scratch***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model no | Train Size | Validation and Test sample size | Data Augmentation | Test Accuracy% | Validation  Accuracy% |
| Model 1 | 1000 | 500,500 | NO | 71.3 | 70.4 |
| Model 1a | 1000 | 500,500 | YES | 81.5 | 81.8 |
| Model 2 | 1500 | 500,500 | NO | 75.8 | 74.6 |
| Model 2a | 1500 | 500,500 | YES | 84.8 | 85.2 |
| Model 2b | 1500 | 500,500 | YES | 52.4 | 50.1 |
| Model 2c | 1500 | 500,500 | NO | 75.2 | 75.1 |

***Table for Pre-Trained Models***

|  |  |  |
| --- | --- | --- |
| **Data Augmentation** | **Train Accuracy %** | **Validation Accuracy%** |
| NO | 97.3 | 97.1 |
| YES | 96.9 | 97.2 |

**Conclusion:**

The tables above show the settings of our model and the sizes of the training, testing, and validation sets. We've included results for different scenarios: with and without adding extra data, training the model from scratch, and adjusting the sizes of the training and validation sets. We're comparing how accurate the model is, how well it performs on new data, and how much the added data helps when using a pre-trained model.

After analyzing the data, we found that improving the model's performance wasn't necessarily the consequence of adding more data to the training set. Increasing the size of the training set or adjusting the validation set's size did, however, improve the model's accuracy. We found that pre-made models did not appear to be more accurate or more adept at handling new data when we added additional data to them. In general, it is more effective to use pre-made models rather than create them from scratch, especially in situations when there is a lack of initial training data.