**ASSIGNMENT 2**

**APPLYING CONVOLUTION NETWORKS (COVNETS) TO**

**IMAGE DATA**

In this assignment, we worked with the cats vs dogs dataset, which was obtained from Kaggle through the Kaggle API. The dataset, as it came from Kaggle, initially consisted of 25,000 images featuring both cats and dogs.

The assignment revolved around exploring the impact of training sample size on the choice of training approach, namely starting from scratch or utilizing a pre-trained convolutional neural network (ConvNet). The key questions we aimed to answer were as follows:

We began by working with a smaller training sample and implemented techniques to combat overfitting while striving to enhance model performance when building a ConvNet from the ground up. The results of this approach were analyzed.

We then scaled up the training sample size while keeping the validation and test datasets consistent with scenario 1. In doing so, we optimized the architecture of the neural network to observe how performance evolved with a larger training dataset.

We explored the notion of determining the ideal training sample size by experimenting with both an increase and a decrease in the training sample. For this assignment, the focus was on augmenting the training sample.

The final part of the assignment delved into the utilization of pre-trained models. We employed different network architectures and optimization techniques to fine-tune these pre-trained models for our specific task.

Throughout this assignment, we aimed to elucidate how the size of the training sample influences the performance of a given model. Furthermore, we sought to establish the relationship between the efficacy of training a model from scratch and the utilization of a pre-trained ConvNet.

Let’s discuss each case separately. The findings are as below: -

* 1. **Model 1 - Training sample of 1000, a validation sample of 500, and a test sample of 500.**

In this instance, a training sample of 1000 images was used, along with 500 images for both the validation and test sets. The objective was binary classification, distinguishing between images of cats and dogs. The model underwent training for 50 epochs, employing the RMSprop optimizer and the ReLU activation function. Notably, the convent architecture was designed to incorporate data augmentation exclusively, without the inclusion of dropout or any other regularization techniques. The aim was to assess the impact of relying solely on data augmentation in achieving accuracy. The results are detailed below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Optimizers and techniques** | **Training Accuracy** | **Training Loss** | **Validation Accuracy** | **Validation Loss** | **Test Accuracy** |
| **1**  **Training -1000**  **Validation –**  **500**  **Test - 500** | **RMSprop Data**  **Augmentation** | **0.99** | **0.6008** | **0.70** | 3.68 | **0.68** |

The observed pattern indicates that the model is suffering from overfitting. This is evident as the training accuracy continues to rise, while the validation accuracy plateaus after a certain number of epochs. This overfitting trend is further underscored by the behavior of the validation loss, which consistently increases. The test accuracy, at 71.4%, provides additional confirmation of the overfitting issue. This suggests that the model struggled to generalize effectively to new, unseen data, as discussed earlier.

* 1. **Model 2 - Training sample of 3000, a validation sample of 500, and a test sample of 500.**

In this scenario, I expanded the training sample by adding 3000 more data points, while maintaining the same validation and test sample sizes as in the previous case. The model underwent 50 epochs of training with the same optimizers. However, in this iteration, I implemented several techniques aimed at enhancing the model's performance. These techniques included Dropout, adjusting the Learning Rate, and employing Early Stopping. Early stopping was used to halt the optimization process when it ceased to provide further improvement, with a patience value set at 10. The outcomes of this approach are detailed below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Optimizers and techniques** | **Traini ng**  **Accur acy** | **Traini ng**  **Loss** | **Validation Accuracy** | **Validation Loss** | **Test Accuracy** |
| **2**  **Training -3000**  **Validation – 500**  **Test - 500** | **RMSprop**  **Data**  **Augmentation Dropout**  **Learning Rate**  **Early Stopping** | **0.9602** | **0.1280** | **0.882** | **0.3873** | **0.844** |

Based on the results, it's apparent that the model achieved a high level of accuracy when tested on the training dataset, but it fell short in terms of validation performance. In comparison to the first model, this iteration demonstrated improved performance, primarily due to the enlargement of the training dataset and the incorporation of data augmentation, along with additional techniques like dropout, learning rate adjustments, and early stopping. Nevertheless, the validation accuracy and loss figures still suggest overfitting. However, the combination of an increased sample size and these techniques led to a better outcome, accompanied by a slight reduction in loss. In contrast to the initial model, which lacked regularization, this model exhibited significant progress, as evidenced by the increase in test accuracy.

* 1. **Model 3 - Training sample of 9000, a validation sample of 500, and a test sample of 500.**

For the third model, I continued to expand the training sample by adding an additional 900 data points while keeping the validation and test samples at their previous sizes. I retained the same techniques and approaches used in the second model, without introducing any additional methods. The primary focus in this case was to ascertain the optimal training sample size for achieving accurate prediction results.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Optimizers and techniques** | **Traini ng**  **Accur acy** | **Traini ng**  **Loss** | **Validation Accuracy** | **Validation Loss** | **Test Accuracy** |
| **3**  **Training -9000**  **Validation – 500**  **Test - 500** | **RMSprop Data**  **Augmentation Dropout**  **Learning Rate**  **Early Stopping** | **0.9526** | **0.1448** | **0.904** | **0.3043** | **0.895** |

Based on the observations, it can be deduced that the model effectively grasped the nuances of the training data, although there might be some minor signs of overfitting, as indicated by the validation accuracy being slightly lower than the training accuracy. However, the test accuracy implies that the model is adept at generalizing to new, unseen data. It can be reasonably stated that expanding the training sample consistently benefits the model's performance, as this particular model significantly outperformed the previous ones.

* 1. **Pretrained Models - VGG16 Pretrained Convnet Network, ResNet50V2 convolutional base and MobileNetV2**

In this particular case, I opted for pre-trained models instead of building one from the ground up. The training sample consisted of 2000 data points, while both the validation and test samples contained 1000 each. I experimented with three distinct pre-trained models, each employing different optimizers and techniques. The outcomes of these experiments are detailed below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Pretrained Model** | **Optimizers and techniques** | **Trainin g**  **Accura cy** | **Trainin g Loss** | **Validation Accuracy** | **Validation Loss** | **Test Accuracy** |
| **Pretrained Model 1 - VGG16**  **Convnet Network Training -2000**  **Validation – 1000**  **Test - 1000** | **RMSprop Data**  **Augmentation Dropout**  **Learning Rate Model**  **Checkpointing**  **VGG16**  **convolutional base** | **0.9669** | **0.1175** | **0.8780** | **0.4750** | **0.904** |
| **Pretrained Model**  **2: ResNet50V2 convolutional**  **base**  **Training -**  **2000**  **Validation – 1000**  **Test - 1000** | **Adam Optimizer**  **Data**  **Augmentation**  **Early Stopping Dropout** | **0.9844** | **0.0821** | **0.9780** | **0.2688** | **0.976** |
| **Pretrained Model 3: MobileNetV2 Training -2000**  **Validation – 1000**  **Test - 1000** | **RMSprop Data**  **Augmentation**  **Early Stopping Model**  **Checkpointing Dropout** | **0.9614** | **0.1058** | **0.9850** | **0.0578** | **0.9860** |

As per the results, all three models demonstrated exceptional performance on both the training and validation datasets, achieving accuracy levels surpassing 96%. Among them, the MobileNetV2 model stood out with the highest test accuracy at 98.6%, closely followed by the ResNet50v2 model. The VGG16 model also displayed commendable performance, securing a test accuracy of 90.4%, albeit slightly lower in comparison. It is evident that the choice of utilizing a pre-trained model, in addition to the optimizer and techniques employed during training, significantly influences the model's performance.

In summary, employing a pre-trained model can notably reduce training time and enhance the accuracy of image classification tasks when contrasted with constructing custom models from scratch. The integration of regularization techniques, such as data augmentation, dropout, learning rate adjustments, and early stopping, played a crucial role in averting overfitting and elevating the models' ability to generalize. Furthermore, the expansion of the training sample size contributed to enhanced learning when training models from the ground up. It was observed that the Adam optimizer outperformed the RMSprop optimizer in fine-tuning pre-trained models. These collective insights suggest that a combination of pre-trained models and diverse optimization techniques can yield high accuracy and robust image classification models.