ASSIGNMENT 3

APPLYING CONVULUTION NETWORKS (COVNETS) TO IMAGE DATA

In this assignment, we have used the cats vs dogs dataset. I have downloaded the dataset from Kaggle using the Kaggle API. In the next step, I unzipped the data and then I created a new subset of the dataset for use in training and testing a ConvNet model. The dataset downloaded from Kaggle had 25000 images of cats and dogs.

In this assignment, I have examined the relationship between training samples and the choice of training model from scratch, versus using a pretrained convnet. Specifically, I have answered the following questions: -

1. Taking a small training sample and using techniques to reduce overfitting and improve performance in developing a network from scratch and determining the results.
2. Increasing the training sample size while keeping the validation and test samples same as in scenario 1 and optimizing the network.
3. Finding the ideal training sample by increasing or decreasing the training sample. (In this assignment, I have increased the training sample)
4. Using pretrained models to train by using different network models and optimization techniques.

We will discuss how the size of the training sample affects how a particular model performs and determine the relationship between training your model from scratch versus using a pretrained covnet.

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 case, I have taken a training sample of 1000 and 500 each for validation and test samples. The model is being trained in binary classification task of clarifying images of cats and dogs. The model was trained on 50 epochs using RMSprop optimizer and Relu activation function. I have defined convnet that includes only data augmentation without using dropout or any other regularization techniques. I wanted to see how using just data augmentation will define accuracy. The results are as follows: -

Model	Optimizers and techniques	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Test Accuracy
1 Training -1000 Validation – 500 Test - 500	**RMSprop Data Augmentation**	**0.9920**	**0.0290**	**0.74**	**3.1450**	**0.7140**

From the above, we can see that the model is overfitting as the training accuracy keeps increasing but the validation accuracy does not after few epochs. The overfitting can also be interpreted by the validation loss as it keeps increasing. The test accuracy is only 71.4, which suggests overfitting as discuss above as the model did not generalize well with the new data.

2.

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

In this case, I have increased the training sample by 3000 and I kept the validation and test sample the same as in the previous case. This model too was trained on 50 epochs using the same optimizers, but I also added a few techniques to improve the model's performance. I have tried Dropout, Learning Rate and Early stopping. I used the early stopping to stop optimization when it isn't helping anymore. The patience value is equal to 10. The results were as follows: -

Model	Optimizers and techniques	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Test Accuracy
2 Training -3000 Validation – 500 Test - 500	**RMSprop Data Augmentation Dropout Learning Rate Early Stopping**	**0.9640**	**0.1304**	**0.8230**	**0.5831**	**0.8230**

From the above, we can infer that the model performed well on the training sample but did not perform well on the validation sample. Compared to the 1st Model, this model performed well because we have increased the sample size and used data augmentation with other techniques like dropout, learning rate and early stopping. The validation accuracy and loss indicate the model has overfit. However, increasing the sample size and using these techniques got me a better result while the loss is slightly reduced. In contrast to the first mode, which was not regularized, this model performed quite well as the test accuracy increased.

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

In the 3rd model, I have further increased the training sample by 900 by keeping the validation and test sample the same. I have used the same techniques that I used in the 2nd model. I have not used any other techniques as the main objective in this case is to determine the ideal training sample to get the prediction results.

Model	Optimizers and techniques	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Test Accuracy
3 Training -9000 Validation – 500 Test - 500	**RMSprop Data Augmentation Dropout Learning Rate Early Stopping**	**0.9604**	**0.1410**	**0.9150**	**0.3707**	**0.912**

From the above, we can conclude that the model has learned the training data well but there might be some overfitting to the training data as the validation accuracy is slightly lower than the training data. However, the test accuracy suggests that the model is generalizing well to the unseen data. We can say that increasing the training sample always helps the model as this model performed way better than the other models.

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

In this scenario, I have used pretrained models rather than training a model from scratch. The training sample was 2000 while the validation and test sample were 1000 each. I have used 3 different models with different optimizers and techniques. The results were as follows: -

Pretrained Model	Optimizers and techniques	Training Accuracy	Training 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**	**0.9683**	**0.1325**	**0.9180**	**0.4393**	**0.904**
Pretrained Model 2: ResNet50V2 convolutional base Training - 2000 Validation – 1000 Test - 1000	**Adam Optimizer Data Augmentation Early Stopping Dropout**	**0.9873**	**0.0787**	**0.9820**	**0.1678**	**0.982**
Pretrained Model 3: MobileNetV2 Training -2000 Validation – 1000 Test - 1000	**RMSprop Data Augmentation Early Stopping Model Checkpointing Dropout**	**0.9611**	**0.1041**	**0.9810**	**0.0552**	**0.9860**

From the above, all three models performed very well on both the training and validation sets, with accuracy above 96. The MobileNetV2 model achieved the highest test accuracy of 98.6, followed closely by the ResNet50v2 model. The VGG16 model too performed well with a test accuracy of 90.4 but is comparatively lower than the others. It can be said that using a pretrained model as well as the choice of optimizer and techniques used in the training will have a significant impact on the performance of the model.

To have a clear understanding of what we did, I have summarized all the models into a combined table. The table is as follows: -

Model	Optimizers and techniques	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Test Accuracy
1 Training - 1000 Validation – 500 Test - 500	RMSprop Data Augmentation	0.9920	0.0290	0.74	3.1450	0.7140
2 Training - 3000 Validation – 500 Test - 500	RMSprop Data Augmentation Dropout Learning Rate Early Stopping	0.9640	0.1304	0.8230	0.5831	0.8230
3 Training - 9000 Validation – 500 Test - 500	RMSprop Data Augmentation Dropout Learning Rate Early Stopping	0.9604	0.1410	0.9150	0.3707	0.912
Pretrained Model 1 - VGG16 Convnet Network Training - 2000 Validation – 1000 Test - 1000	RMSprop Data Augmentation Dropout Learning Rate Model Checkpointing VGG16 convolutional base	0.9683	0.1325	0.9180	0.4393	0.904
Pretrained Model 2: ResNet50V2 convolutional base Training - 2000 Validation – 1000 Test - 1000	Adam Optimizer Data Augmentation Early Stopping Dropout	0.9873	0.0787	0.9820	0.1678	0.982
Pretrained Model 3: MobileNetV2 Training - 2000 Validation – 1000 Test - 1000	RMSprop Data Augmentation Early Stopping Model Checkpointing Dropout	0.9611	0.1041	0.9810	0.0552	0.9860

Findings and Conclusion: -

- The 1st model had the highest training accuracy but the lowest validation and test accuracy which indicates overfitting.

- The 2nd model was a regularized model and used various techniques such as data augmentation, dropout, learning rate, and early stopping to reduce overfitting, which resulted in a slightly better test accuracy compared to the 1st model.
- The 3rd model used the same techniques as the 2nd model but had a larger training sample of 9000. This resulted in the highest test accuracy among the 3 models.
- The three pretrained models (VGG16, ResNet50V2, MobileNetV2) achieved high training and validation accuracies and low losses, indicating that they have learned useful features from large datasets and can be fine tuned for specific tasks with relatively small amounts of data.
- Among the pretrained models, MobileNetV2 achieved the highest test accuracy.
- Using Data Augmentation in all the models and dropouts helped improve generalization performance and prevent overfitting.
- Using the RMSprop optimizer helped achieve a decent test accuracy, but it was not as effective as the Adam optimizer.
- Learning rate tuning and early stopping have also helped in better generalization and prevention of overfitting.
- Incorporating a pre-trained model into the training process can reduce the required training time and minimize the effort required to develop a custom model architecture.
- Pre- trained models tend to outperform custom- built convolutional neural networks and offer better accuracy and results.
- A larger sample size typically improves the model's learning performance when training from scratch.
- In a pre-trained model, we can improve the model performance by fine-tuning the top layers and jointly training the newly added layers with frozen layers.

In conclusion, we can say that using a pretrained model could significantly reduce training time and provide better accuracy of image classification tasks compared to custom built models. Using regularization techniques such as data augmentation, dropout, learning rate, and early stopping helped prevent overfitting and improved the generalization performance of the models. Increasing the sample size can lead to better learning performance when training from scratch. The Adam optimizer was found to be more effective than the RMSprop optimizer for fine-tuning pre-trained models. Overall, these findings suggest that a combination of pre-trained models and various optimization techniques can lead to high accuracy and robust image classification models.