Convolutional Neural Networks Assignment2 Cats & Dogs

Objective:
The objective of this project is to develop a convolutional neural network (CNN) capable of accurately recognizing cat and dog images. Additionally, we aim to investigate the impact of training size on the model-building process.

Model Building:
In total, we constructed 15 models with diverse configurations, including different layers, nodes, optimizers, and other hyperparameters. The models can be categorized into two groups: Scratch Models and Pre-Trained Models.

For the Scratch Models, we trained the CNN from scratch without utilizing any pre-existing weights or feature extraction. Various hyperparameters were adjusted. For the Pre-Trained Models, we employed transfer learning by utilizing pre-trained models such as VGG16, ResNet50, or InceptionV3 as the base architecture. We fine-tuned these models by adapting them to our specific task of cat and dog image recognition.

Hyper Tunning Parameters: (Scratch Models)

#No.	Input Layers	Filters	Filter Size	Optimizer	Training	Validation & Test	Dropout
Model 1	5	32 to 256	3	Adam	1000	500 & 500	-
Model 2	5	32 to 256	3	Adam	1000	500 & 500	0.5
Model 3	6	32 to 512	3	Adam	1000	500 & 500	0.5
Model 4	5	64 to 1024	3	Adam	1000	500 & 500	0.6
Model 5	5	32 to 256	3	Adam	2000	500 & 500	0.5
Model 6	5	32 to 256	3	Adam	2000	500 & 500	0.5
Model 7	5	32 to 256	3	Adam	3000	500 & 500	0.5
Model 8	5	32 to 256	3	Adam	3000	500 & 500	0.5
Model 9	**5**	**32 to 256**	**3**	**Adam**	**5000**	**500 & 500**	**0.5**

#No.	Max Pooling	Strides	Padding	Augmented Images	Test Performance (Loss, Accuracy)
Model 1	Yes (Pool Size = 2)	-	-	-	(0.645, 0.614)
Model 2	Yes (Pool Size = 2)	-	-	Yes	(0.601, 0.712)
Model 3	Yes (Pool Size = 2)	-	-	Yes	(0.609, 0.692)
Model 4	Yes (Pool Size = 2)	-	-	Yes	(0.666, 0.652)
Model 5	Yes (Pool Size = 2)	-	-	Yes	(0.445, 0.860)
Model 6	-	Yes (Strides = 2)	-	Yes	(0.608, 0.650)
Model 7	Yes (Pool Size = 2)	-	-	Yes	(0.495, 0.818)
Model 8	Yes (Pool Size = 2)	Yes (Strides = 2)	-	Yes	(0.425, 0.856)
Model 9	**Yes (Pool Size = 2)**	-	-	**Yes**	**(0.182, 0.920)**

Findings:

- When the training size was 1000 samples, the first four models did not achieve high accuracy. However, Model 2, which utilized augmented images, showed the best accuracy among the four models. This indicates that data augmentation is an effective technique for improving model performance.
- Models 3 and 4 introduced changes to the filters and layers. Model 3 had 6 layers with filters ranging from 32 to 512, while Model 4 had 5 layers with filters ranging from 64 to 1024. However, increasing the number of filters and layers did not lead to significant improvements in model performance.
- Increasing the training size to 2000 samples for Models 5 and 6 resulted in improved accuracy. Model 5 achieved 86% accuracy and a loss of 44.5%, compared to 71.2% accuracy with 1000 training samples. This suggests that providing the model with more training samples, including augmented versions, enables better learning and recognition.
- Model 6 utilized strides instead of a pooling layer to decrease spatial dimensionality. However, this change did not contribute to a significant improvement in model performance.
- Model 7, which replicated Models 2 and 5 with an increased training sample size of 3000, showed a decrease in accuracy from 86% to 81.8%. This indicates that simply increasing the sample size does not guarantee improved performance. The selection of an appropriate sample size is crucial for effective training and generalization.
- Model 8, which incorporated both max pooling and strides, demonstrated improved accuracy, increasing from 81.8% to 85.6%.
- Finally, when the sample size was increased to 5000, the models that achieved the highest accuracy with different sample sizes (Models 2 and 5) were selected. This led to an accuracy of 92% and a loss of 18.2%.

Pre-Trained Network:

VGG-16, a widely-used pre-trained network, was employed for the image recognition task. This model has been trained on a vast dataset comprising thousands and hundreds of thousands of images across various categories. Its extensive training makes VGG-16 a powerful tool for image recognition tasks.

Hyper Tunning Parameters: (Pre-Trained Models)

#No.	Dense Layer	Training Size	Optimizer	Validation & Test	Dropout
Model 1	1 (256 Nodes)	1000	rmsprop	500 & 500	0.5
Model 2	1 (256 Nodes)	1000	rmsprop	500 & 500	0.5
Model 3	1 (256 Nodes)	1000	rmsprop	500 & 500	0.5
Model 4	1 (256 Nodes)	5000	Adam	500 & 500	0.5
Model 5	1 (256 Nodes)	5000	Adam (1e-5)	500 & 500	0.5

#No.	Pre-Trained Weights Update Set to False	Freeze Layers	Augmented Images	Test Performance (Loss, Accuracy)
Model 1	No	False	No	(12.350, 0.956)
Model 2	Yes	False	Yes	(5.111, 0.958)
Model 3	Yes	True	Yes	(9.085, 0.966)
Model 4	Yes	False	Yes	(0.236, 0.986)
Model 5	Yes	True	Yes	(0.083, 0.994)

Findings:

- The size of the training sample played a significant role in determining the learning characteristics and performance of the models using the pre-trained network. A larger training sample size generally resulted in better model performance on unseen data.
- While rmsprop is a highly effective optimizer function for building convolutional neural networks, Adam optimizer demonstrated superior performance due to its unique combination of Momentum and rmsprop. This combination allows for more efficient optimization of the neural network.
- By setting the trainable parameter of the pre-trained network to False, we prevented the pre-trained network from updating its weights during training. This approach allowed the model to focus more on training the densely connected classifier layer at the end. This technique proved to be effective in improving the performance of Models 2 and 3, as it helped prevent overfitting.
- Freezing the initial layers of the pre-trained network enabled the model to concentrate specifically on the image recognition task at hand, as opposed to the broader categorization layers for which the network was originally trained. This strategy was beneficial in the fine-tuned models (Model 3 and Model 5), resulting in higher accuracy

compared to other models within their respective sample sizes.

Conclusion:

Increasing the sample size for pre-trained networks, along with providing augmented versions of the images, resulted in improved model performance and minimal loss values. Freezing the initial layers of the pre-trained network and preventing weight updates during training proved effective in controlling overfitting and promoting generalization on unseen data. These findings highlight the importance of sample size and fine-tuning strategies in optimizing the accuracy and performance of pre-trained network models.