ASSIGNMENT 3: CONVOLUTIONAL NEURAL NETWORK (BA_64061_001)

Naveen Kumar Yadav

Introduction

This summary report is an attempt to provide insight into how the convolutional neural networks (CNNs) for binary image classification models are built and works to distinguish between images of cats and dogs. The aim of this assignment was to build various models by varying the training sample size to find the training sample size and technique to get the best performing model. To achieve the task of the best model, 9 models were built from scratch by taking various training sample sizes, models with augmentation, and used pretrained models while varying the size of the training sample. I started with a small training sample and gradually increased its size to examine how this progression affected the model's accuracy and robustness. The goal was to identify the optimal training sample size and technique for achieving high performance models, comparing the effectiveness of models trained from scratch with the pretrained models.

Dataset Description

The dataset used in this assignment is the smaller version of Cats and Dogs dataset that is available in Kaggle, a good resource for binary image classification tasks. It includes 4000 labelled images grouped into two classes: cats and dogs. For this Assignment, we used only a portion of the dataset. Each image was processed into a consistent format, so it could be used with convolutional neural networks. During the experiments, we varied the size of the training set to see how it affected the model's performance, while keeping the validation and test sets the same to allow fair comparisons. Though the size of validation and test sets were kept constant, I took different start and end indexes to validate and test the model on different samples.

Data Preprocessing

1. The image files were loaded using the image_dataset_from_directory() function, which automatically labels and batches the data based on folder structure.
2. Separate datasets were created for training, validation, and testing to ensure proper evaluation and generalization.
3. Each image was resized to **180 × 180 pixels** and grouped into batches of **32 images** to optimize model training.
4. Pixel values were normalized to the range [0, 1] to improve model performance and training stability.

5. TensorFlow's dataset operations were used to handle reshaping, batching, and mapping efficiently for further processing.

Model Training

1. Trained CNN models by varying training sample size and every new model were trained from scratch after validating and testing the previous model.
2. Models were trained with and without augmentation to observe the difference in their performance.
3. Implemented pretrained models with varying training sample size for comparison with the models built from scratch in this assignment earlier.

Performance Evaluation

1. Analyzed training and validation of accuracy and loss across multiple experiments to measure model performance.
2. Compared results obtained from different training sample sizes to understand how data quantity affects accuracy and stability.
3. Monitored validation accuracy to detect signs of overfitting and assess the model's ability to generalize unseen data.
4. Evaluated test set results to ensure consistent performance and verify the effectiveness of the chosen training approach.

Methodology & Performance Analysis

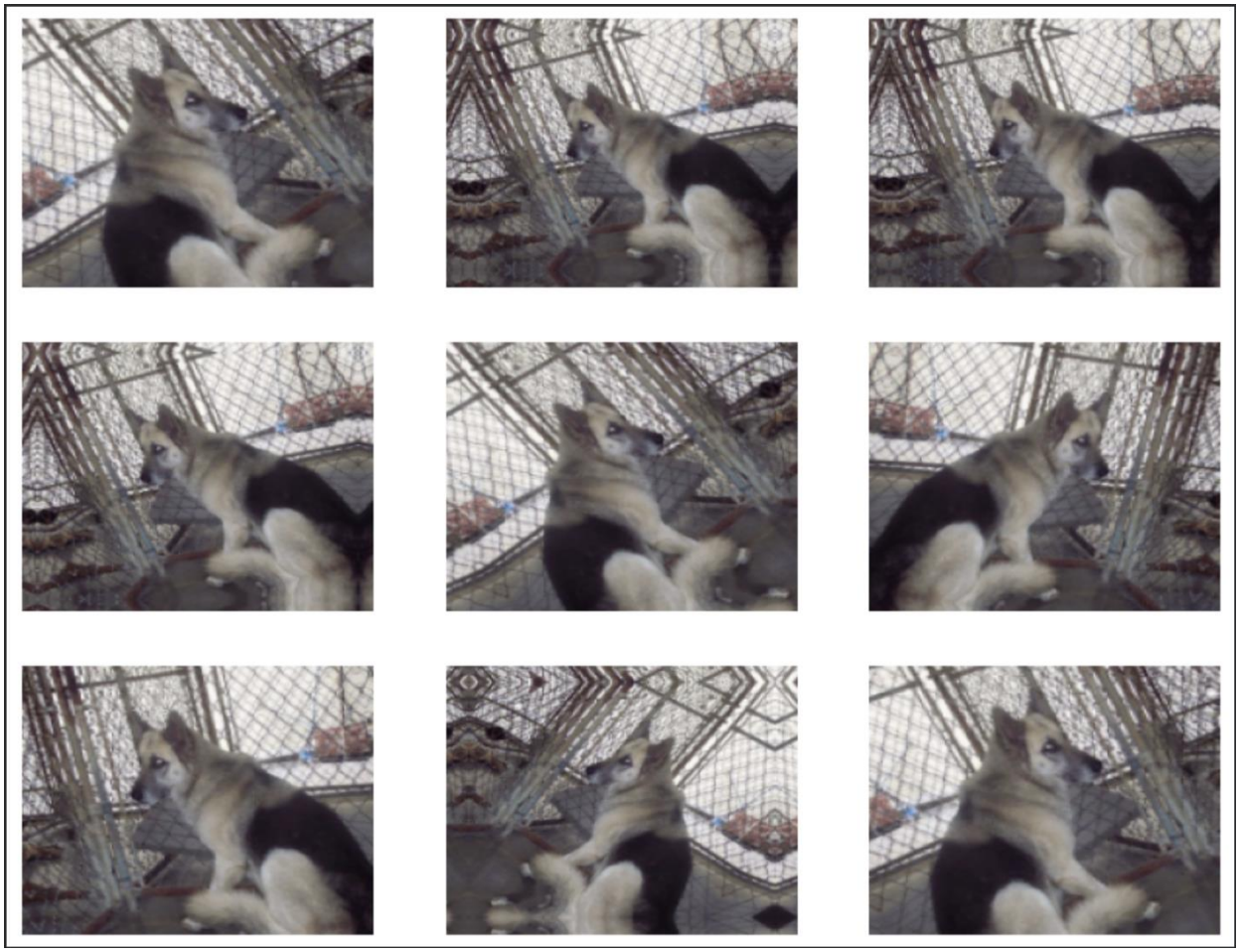
To study how the size of the training dataset influences model performance, I built and tested a series of convolutional neural network (CNN) models under consistent conditions. All models used the same CNN architecture, ensuring that differences in performance were due only to changes in training sample size or the use of pretrained weights.

Initially, I trained **five models from scratch**, using training sample sizes of **1000, 1500, 1600, 1900, and 2100 images of cats and dogs**. The validation and test datasets were kept the same size across all runs, but the sample indexes were varied to ensure fair comparisons. Each model without image augmentation was trained for **30 epochs**, while models that included augmentation were trained for **100 epochs** to allow more time for learning from the expanded data variations. The CNN architecture consisted of five convolutional layers with ReLU activation functions, followed by a fully connected layer with a sigmoid activation for binary classification. The models were compiled using the **binary cross-entropy loss function**, the **RMSprop optimizer**, and **accuracy** as the main evaluation metric. To improve generalization, we used **data augmentation** techniques such as random flipping, rotation, and zooming. These transformations introduced variety into the dataset, helping the model become more robust with changes in image orientation and scale.

In the next phase, we trained **four additional models using pretrained networks** to explore the effect of transfer learning. These pretrained models were trained with sample sizes of **1000, 1500, 1900, and 1800 images**, while keeping the validation and test sets constant in size. The pretrained models also used **image augmentation** and were trained for **100 epochs** to allow the model to adapt to the new dataset effectively. This setup allowed for a detailed comparison between models trained from scratch and those built on pretrained architectures. By keeping most parameters consistent and varying only the training data size and augmentation methods, we were able to clearly observe how these factors influenced accuracy, loss, and overall model robustness.

Implementation of Data Augmentation

The goal of data augmentation in this project is to improve model accuracy by expanding the diversity of the training data. This technique generates new images from existing ones by applying random transformations, allowing the model to learn from slightly different versions of the same data. By exposing the model to varied examples, it becomes better at recognizing patterns and generalizing unseen images. In this work, random flipping, rotation, and zooming were applied to the training images to introduce variation and reduce the risk of overfitting. This approach increased dataset variety and helped the model achieve more stable and reliable performance.



Model Performance Analysis

Model & Sample Size	Method	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
Model-1 Training Sample- 1000 Validation Sample- 500 Test Sample- 500	Without Augmentation	71.6%	0.588	68.2%	0. 629
	With Augmentation	80.0%	0.480	77.2%	0.584
Model-2 Training Sample- 1500 Validation Sample- 500 Test Sample- 500	Without Augmentation	68.8%	0.576	71.4%	0.652
	With Augmentation	77.2%	0.475	76.0%	0.476
Model-3 Training Sample- 1600 Validation Sample- 500 Test Sample- 500	Without Augmentation	71.6%	0.555	68.0%	0.602
	With Augmentation	81.0%	0.407	81.6%	0.445
Model-4 Training Sample- 1900 Validation Sample- 500 Test Sample- 500	Without Augmentation	72.2%	0.560	69.4%	0.568
	With Augmentation	81.4%	0.464	78.6%	0.544
Model-5 Training Sample- 2100 Validation Sample- 500 Test Sample- 500	Without Augmentation	73.0%	2.055	68.0%	2.927
	With Augmentation	81.4%	0.436	78.8%	0.488

Pretrained Model Performance Analysis

Model & Sample Size	Method	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
Model-6 Training Sample- 1000 Validation Sample- 500 Test Sample- 500	With Augmentation	97.6%	3.27	97.2%	8.239

Model-7 Training Sample- 1500 Validation Sample- 500 Test Sample- 500	With Augmentation	98.0%	1.063	97.2%	2.12
Model-8 Training Sample- 1900 Validation Sample- 500 Test Sample- 500	With Augmentation	98.2%	2.492	97.5%	2.492
Model-9 Training Sample- 2200 Validation Sample- 500 Test Sample- 500	With Augmentation	98.2%	1.442	97.4%	1.072

Conclusion

The results show a clear relationship between training sample size, data augmentation, and the use of pretrained ConvNets. Among the models trained from scratch, **Model 3**, trained on 1600 samples with data augmentation, performed the best overall. It achieved **81.0% validation accuracy** and **81.6% test accuracy**, with the lowest loss values, indicating strong generalization and effective learning. Increasing the training size beyond this point provided only small improvements in accuracy while slightly increasing loss, suggesting that more data alone does not always lead to better generalization.

When pretrained ConvNets were used, performance improved noticeably. **Model 8**, trained with 1900 samples and augmentation, reached the **highest test accuracy (97.5%)**, while **Model 9** achieved a similar accuracy with a lower loss. These results highlight the strength of pretrained models in learning complex visual patterns efficiently, even with limited data.

Overall, increasing the training sample size helps up to an optimal level, but data augmentation remains essential for controlling overfitting. Pretrained ConvNets consistently outperform models trained from scratch, showing that transfer learning is a more effective and efficient approach when working with smaller datasets.