Convolutional Report

By

Sneha Ashok Bhagat

Assignment 3

ADVANCED MACHINE LEARNING (BA-64061-001)

Chaojiang (CJ) Wu, Ph.D.

03/25/2025

Executive Summary

In this report, we explored the application of convolutional neural networks (CNNs) for binary image classification and focused on optimizing model performance in distinguishing between cat and dog images.

Our objective was to assess the impact of training CNNs from scratch versus using pretrained models while varying the training sample size.

We began with a small dataset and progressively increased and decreased the training sample size while keeping the validation and test sample same. This help to analyze the performance differences to determine the best accuracy and robustness across all training sizes for both scratch-trained and pretrained models.

Data Overview

The dataset used in this assignment is the Cats and Dogs dataset, a well-known benchmark for image classification. It consists of thousands of labeled images categorized into two classes: “Cats” and “Dogs.” For our experiments, we used a subset of this dataset, organized into three main folders:

1. **Training Folder:** Contains two subfolders—one for cats and one for dogs—each with 1,000 images, totaling 2,000 training images.
2. **Validation Folder:** Includes two subfolders for cats and dogs, each containing 500 images, ensuring a balanced validation set for performance evaluation.
3. **Test Folder:** Maintains the same structure as the validation folder, with 500 images of cats and 500 images of dogs for final model assessment.

Each image was preprocessed into a standardized format suitable for input into convolutional neural networks (CNNs). Throughout the experiments, we varied the training sample size to analyze its impact on model performance, while keeping the validation and test sets fixed to ensure consistent evaluation across different configurations.

Architecture and Performance Analysis

To assess the impact of varying training sample sizes on model performance, we maintained a consistent CNN architecture across five models, modifying only the training dataset size. The training samples were systematically adjusted to 400, 800, 1,000, 1,500, and 2,000 images while keeping the validation and test sets constant.

Each model consisted of five convolutional layers with ReLU activation functions, followed by a dense output layer with a sigmoid activation. The models were compiled using the binary cross-entropy loss function, the RMSprop optimizer, and accuracy as the evaluation metric. This standardized framework allowed for a straightforward comparison of accuracy and loss metrics, isolating the effects of training data variations on model performance.

To enhance generalization, data augmentation techniques were applied to introduce variability within the dataset, making the model more robust. Additionally, a pretrained model was used to leverage transfer learning, extracting meaningful features without the need for extensive retraining. This approach improved model performance while minimizing computational complexity.

Each model was trained both with and without data augmentation, ensuring a comprehensive evaluation of how these techniques influence performance. All models were trained for 30 epochs, balancing performance optimization with computational feasibility.

Process

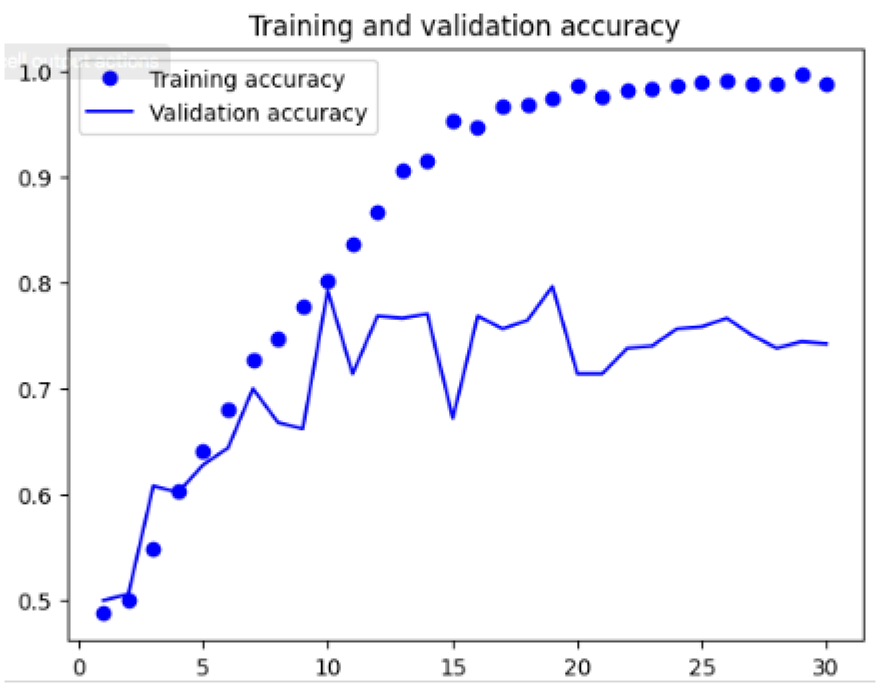
The data was uploaded to local drive and to “Collab Notebook” folder

1. Data Preparation:
    - Images loaded and split using image_dataset_from_directory.
    - Data preprocessing and augmentation applied.
2. Model Training:
    - Trained CNN from scratch and optimized using different sample sizes.
    - Implemented a pretrained network for comparison.

3. Performance Evaluation:
   - Compared accuracy and loss metrics across different experiments.
   - Assessed overfitting trends using validation accuracy.
4. Conclusion:
   - Identified the optimal training sample size.
   - Evaluated the impact of transfer learning versus training from scratch.

Performance Accuracy

Based on all the models evaluated from scratch. Model 3 provided the best testing and validation accuracy. The plot is displayed below



Results

Model and Sample size	**Method**	**Validation Accuracy**	**Test Accuracy**	**Validation Loss**	**Test Loss**
Model 1 Training – 1000 Validation – 500 Test- 500	Without Augmentation	66.8%	66%	0.614	0.595
	With Augmentation	77.8%	74.6%	0.541	0.507
Model 2 Training – 1500 Validation – 500 Test- 500	Without Augmentation	68.2%	65.2%	0.585	0.62
	With Augmentation	78.8%	79%	0.451	0.507
Model 3 Training – 2000 Validation – 500 Test- 500	Without Augmentation	70.2%	74.8%	0.544	0.587
	With Augmentation	80.8%	77.2%	0.457	0.469
Model 4 Training – 800 Validation – 500 Test- 500	Without Augmentation	68.8%	65.8%	0.644	0.594
	With Augmentation	75.6%	73.8%	0.593	0.527
Model 5 Training – 400 Validation – 500 Test- 500	Without Augmentation	61.8%	62%	0.656	0.635
	With Augmentation	67.8%	64.4%	0.651	0.614

Using a Pretrained Network

Model and Sample size	**Method**	**Validation Accuracy**	**Test Accuracy**	**Validation Loss**	**Test Loss**
Model 6 Training – 1000 Validation – 500 Test- 500	Data Augmentation	**97.8%**	**96.2%**	**2.23**	**6.529**
Model 7 Training – 1500 Validation – 500 Test- 500	Data Augmentation	**97.6%**	**96.8%**	**2.74**	**6.18**
Model 8 Training – 2000 Validation – 500 Test- 500	Data Augmentation	**98.2%**	**97%**	**8.24**	**1.8**
Model 9 Training – 800 Validation – 500 Test- 500	Data Augmentation	**95.4%**	**94.8%**	**6.8**	**2.8**
Model 10 Training – 400 Validation – 500 Test- 500	Data Augmentation	**93.8%**	**92.8%**	**7.8**	**3.5**

Comparing the best model from scratch and pretrained model

Best Models	**Validation Accuracy**	**Test accuracy**
Model 3 from scratch Training – 2000 Validation – 500 Test- 500	80.8%	77.2%
Model 8 from pretrained model Training – 2000 Validation – 500 Test- 500	**98.2%**	**97%**

Conclusion

Convolutional neural networks (ConvNets) are highly effective for computer vision tasks and can achieve decent results even when trained from scratch on small datasets. However, overfitting remains a major challenge when working with limited data. Data augmentation serves as a powerful technique to mitigate overfitting by introducing variability into the dataset. Additionally, leveraging pretrained ConvNets through feature extraction allows for efficient model adaptation to new datasets, making it a valuable approach for small image datasets. Fine-tuning further enhances performance by refining the pretrained model's learned representations for the specific task at hand.

This study provides insights into the trade-offs between training from scratch and using pretrained networks, offering guidance on choosing the best approach based on dataset size and computational efficiency.

The results clearly indicate that both data augmentation and leveraging a pretrained network significantly enhance model performance. Among the models trained from scratch, increasing the training sample size generally improved accuracy, with Model 3 (2000 training samples) achieving the best test accuracy of 77.2% with augmentation. Data augmentation consistently improved results across all models by mitigating overfitting and enhancing generalization.

However, using a pretrained model dramatically outperformed training from scratch, with the best model (Model 8, using 2000 training samples) achieving a test accuracy of 97%. Even with smaller training sets, pretrained models consistently surpassed non-pretrained models, highlighting the efficiency of transfer learning. While validation losses were higher for pretrained models, their superior test accuracy demonstrates their robustness and adaptability.

However, the model trained from scratch suffers from underfitting, struggling to capture complex patterns and requiring significantly more data while still delivering lower performance. In contrast, the pretrained model trains much faster since it leverages previously learned features, making it more resource-efficient.

Overall, Pretrained Model 8 is the optimal choice due to its superior accuracy, faster convergence, and efficient utilization of computational resources.