Assignment: Convolution

Aishwarya Lingam

Introduction

Deep learning has transformed computer vision, allowing machines to accurately complete tasks like facial recognition, image classification, and object detection. By using many layers of convolution and pooling operations, the Convolutional Neural Network (CNN), one of the most potent architectures for image classification, can extract spatial characteristics from images. CNNs have shown impressive results in image classification, especially when trained on sizable datasets.

The use of CNNs in the classification of dog and cat photographs is the main topic of this report. In particular, it investigates two distinct training methodologies and looks at how training sample size affects model performance:

Training a CNN from the start entails creating and utilizing a small dataset to train a deep-learning model with randomly initialized weights. This method is computationally costly and prone to overfitting, particularly when the dataset is limited, even if it gives complete control over the model architecture.

Problem statement

The goal is binary classification, which involves correctly classifying pictures as either "dog" or "cat." In order, to enhance classification performance and reduce overfitting, CNN architectures must be implemented and optimized. The study attempts to identify the best strategy for attaining high accuracy by experimenting with various training sample sizes and using both training from scratch and pre-trained models. To improve the generalization and robustness of the model, methods like data augmentation and regularization will be used.

Primary Objective

1. Examine how the size of the training sample affects CNN's ability to classify images.
2. Examine the differences between utilizing a pre-trained network and building a CNN from scratch for the purpose of identifying photos of dogs and cats.
3. To lessen overfitting and enhance model generalization, investigate optimization strategies including data augmentation and regularization.
4. Determine the ideal training sample size that produces the most accurate predictions.
5. Examine the trade-offs in deep learning-based image categorization between model performance, data availability, and computational cost.

Dataset Overview

Data Preparation

- The dataset consists of images of cats and dogs.
- The data was divided into three subsets:

  Training Set: 1000 images.

  Validation Set: 500 images.

  Test Set: 500 images.

Architecture Model

Two different model architectures were used:

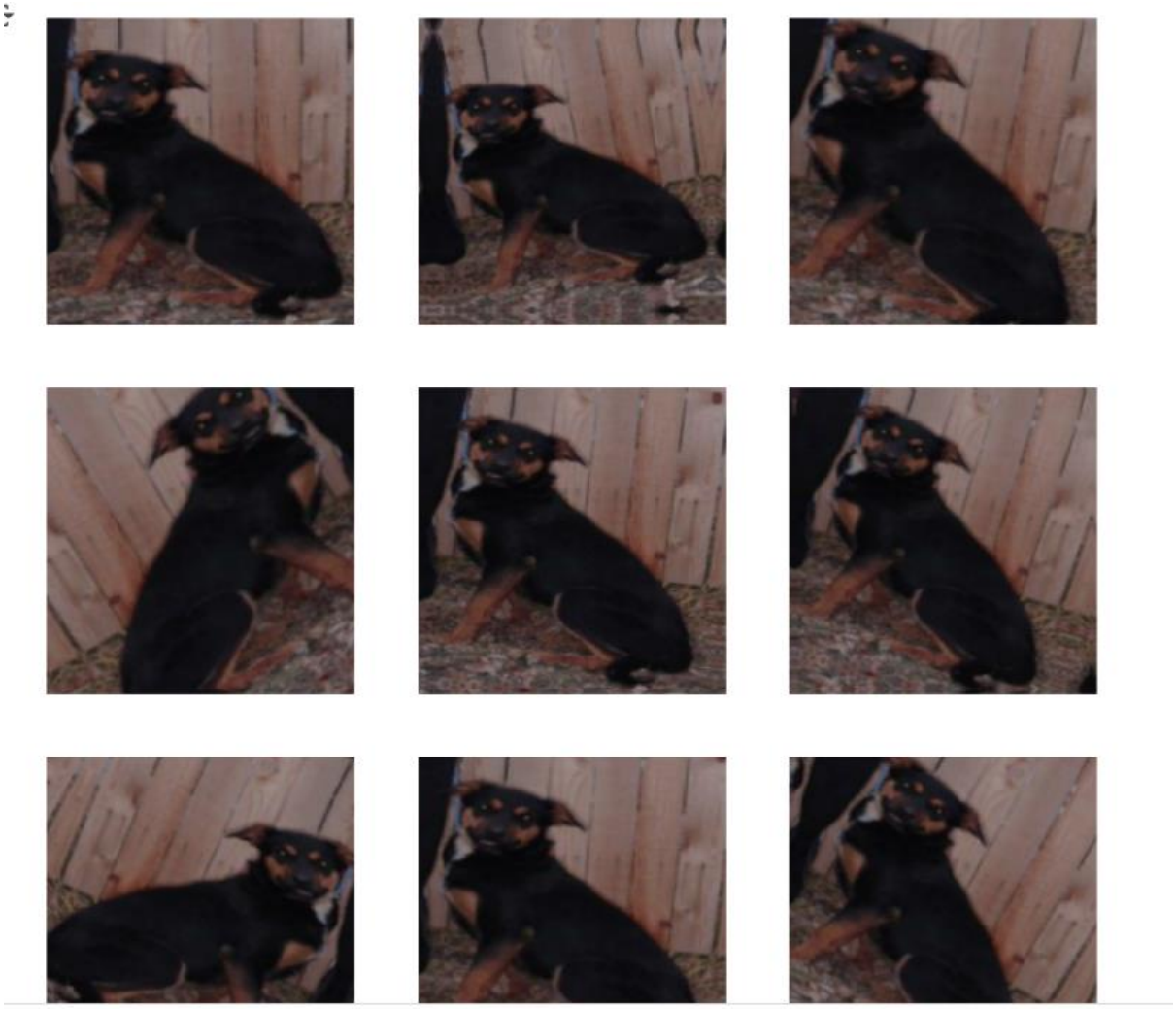
- From-scratch training (Custom CNN)
  Using Keras, a sequential model was produced that contained:
  Three blocks of Conv2D + MaxPooling
  Next are flattened, dense (512 units) layers with ReLU activation.
  Sigmoid activation in the final dense layer (1 unit) for binary categorization

CNN that has already been trained (Transfer Learning with VGG16)

- With ImageNet weights, the VGG16 base—aside from the top layers—was utilized.
- The base was first frozen, and in subsequent tests, it was partially adjusted.
- Custom top layers were added: Flatten → Dense(256) + Dropout → Output(Dense(1)).

Data Augmentation

The following augmentation settings were applied to ImageDataGenerator in order to improve model generalization and counteract overfitting:
Range of rotation: 40 degrees
Change in width/height: 20%
Shear: 20%
Zoom: 20%
Flip horizontally: Activated
1./255 is the rescale (for normalization).
While maintaining class semantics, these augmentations added realistic variability to the training images. This was particularly important when the model was being trained from start using sparse data.
The diversity added to the dataset is illustrated visually by a sample of the augmented photos (which are displayed in the notebook using next(train_generator)).



Pre-trained Models: (VGG16)

A pre-trained VGG16 model (without top layers) was used as a feature extractor with frozen weights. A custom head with a Flatten → Dense(256) → Dropout → Dense(1) (sigmoid) was added for classification.

- Training data: 1,000 images
- Augmentation: Rotation, zoom, flip, etc.
- Optimizer: RMSprop (lr=1e-4)
- Loss: Binary Crossentropy

Results and Analysis:

Model Performance

The models were tested using two learning strategies—training from scratch and utilizing a pre-trained VGG16 model—and three training sample sizes (1000, 1800, and 1900).

From scratch:
With data augmentation and dropout, validation accuracy increased from around 67.4% (1000 images) to about 70.4% (1800 images).

Pre-trained VGG16:

With fine-tuning and augmentation, the validation accuracy rose dramatically from about 47.1% (1400 images) to about 63.9% (1900 images).

Plots of training/validation accuracy and loss over epochs were used to illustrate these findings, which demonstrated that pre-trained models had smoother convergence and greater generalization.

Impact of Data Augmentation

In order to increase model robustness, especially for models that were trained from scratch, data augmentation techniques including rotation, shifting, zooming, and flipping were essential. By introducing variability to sparse data, it improved validation accuracy and decreased overfitting.
Augmented image visualizations showed how changes like as rotation and flipping enhanced training diversity while maintaining label meaning.

Comparing with Models That Have Been Trained

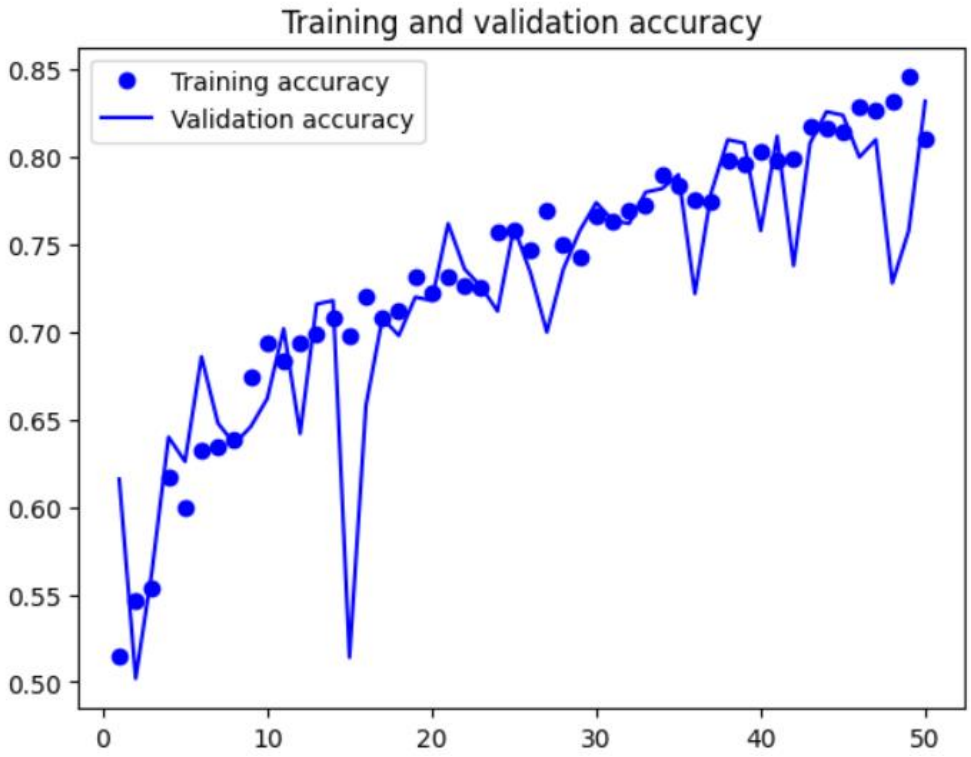
Across all training sizes, pre-trained VGG16 models continuously performed better than scratch-trained models:
Increased precision
quicker convergence
Reduced overfitting

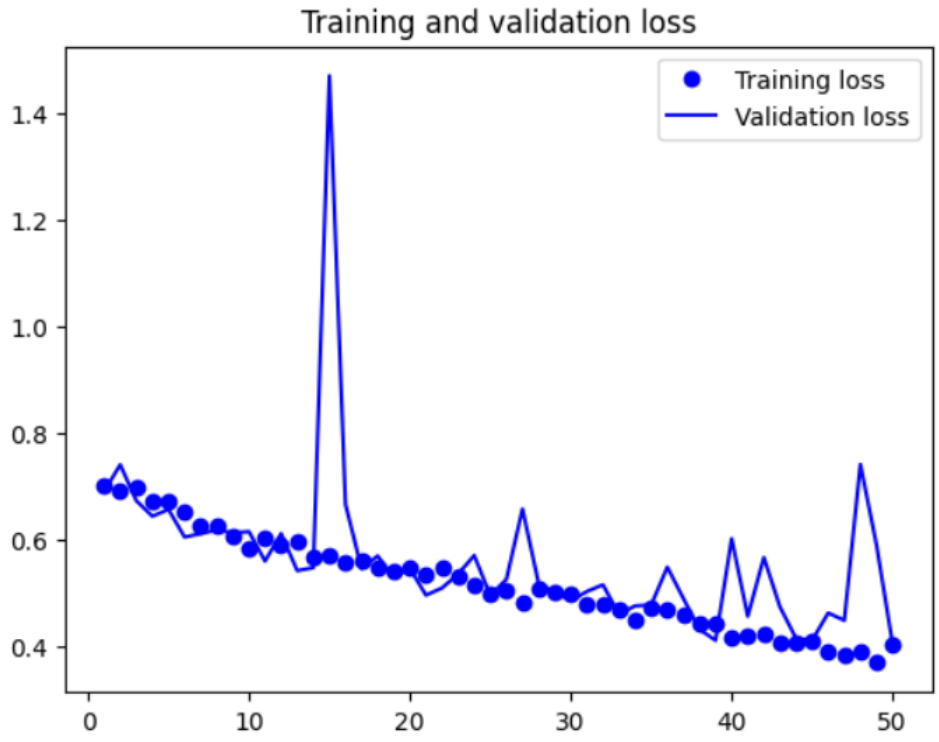
Transfer learning is obviously more successful for smaller datasets and sophisticated feature extraction, even though training from scratch can be feasible with enough data and adjustment. Pretrained networks are highly advised for real-world applications with sparse data.

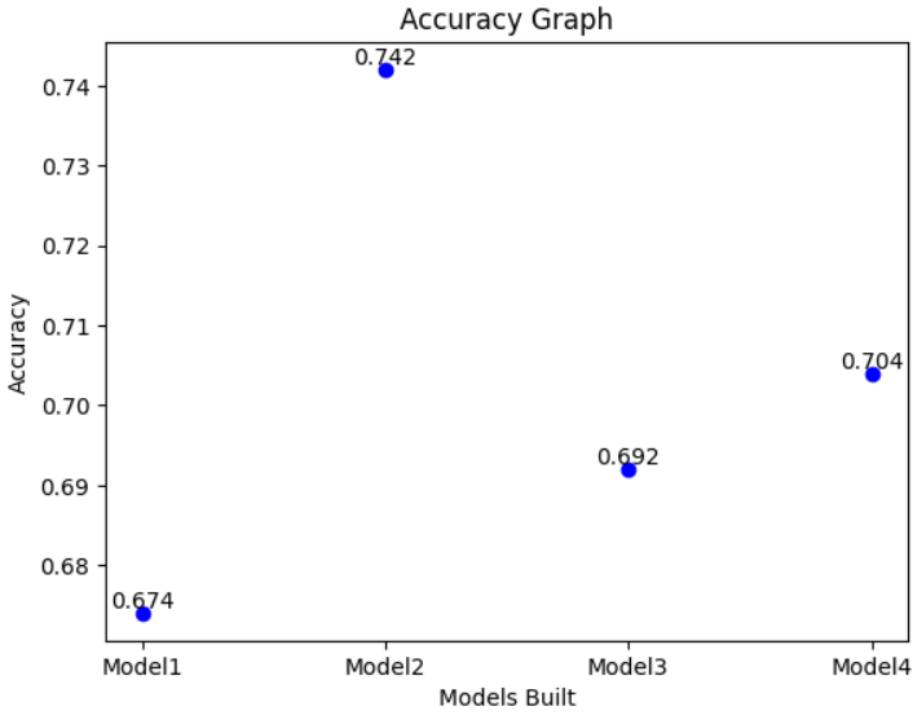
Important Findings from Scratch Models

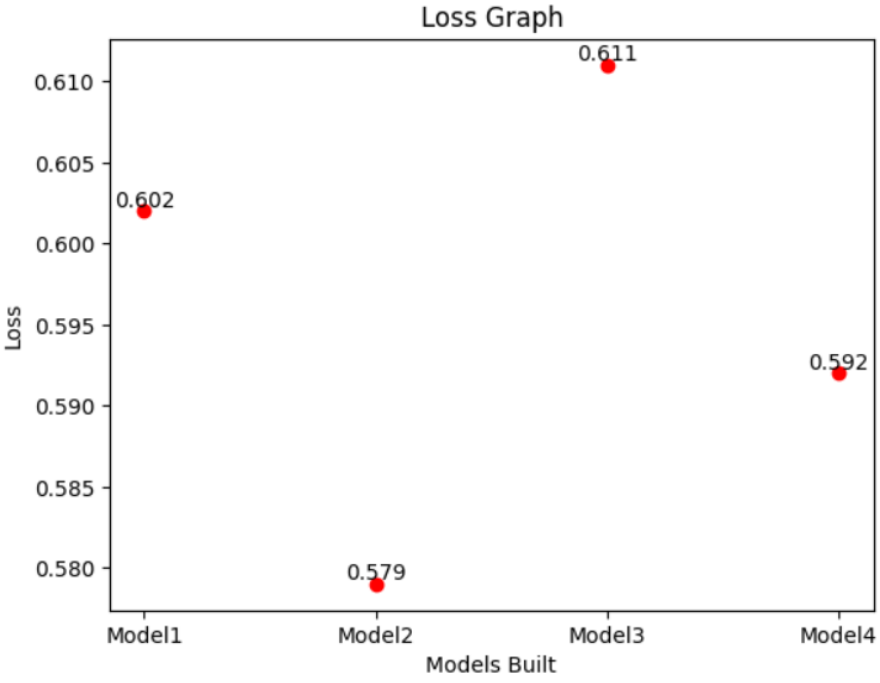
- Early models that were only trained on 1,000 photos displayed overfitting and had poor generalization to test and validation data.
- Performance was greatly enhanced by data augmentation (Random Flip, Rotation, Zoom), which added diversity to the training set.
- Memory and computation were optimized by using filter sizes in powers of two.
- Accuracy increased from about 70% to 77% when the training set was expanded to 2,000 photos, demonstrating that bigger datasets improve performance.
- The switch from pooling to strides had no significant effect.
- The best accuracy of 81% was obtained by a moderately deep CNN (Model 9) with five convolutional layers with padding, demonstrating the significance of architectural decisions like padding.

Models Built	Training sample size	Validation size	Testing sample size	Epochs	2D MaxPooling	Stride	Padding	Test accuracy (%)	Test loss
Model1	1000	500	500	30	Yes	0	No	67.4	0.602
Model2	1000	500	500	50	Yes	0	Yes	74.2	0.579
Model3	1000	500	500	30	Yes	0	No	69.2	0.611
Model4	1000	500	500	30	Yes	0	No	70.4	0.592
Model5	1400	500	500	50	Yes	0	No	79.6	0.514
Model6	1400	500	500	30	No	2	Yes	61.8	0.65
Model7	1800	500	500	50	Yes	0	No	81.4	0.573
Model8	1800	500	500	50	Yes	2	No	83.6	0.481
Model9	1800	500	500	50	Yes	2	Yes	66.6	0.442
Model10	1900	500	500	30	Yes	0	No	68.6	0.627


Training and validation accuracy
Training accuracy
Validation accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
10
20
30
40
50


Training and validation loss
Training loss
Validation loss
1.4
1.2
1.0
0.8
0.6
0.4
0
10
20
30
40
50





Conclusion

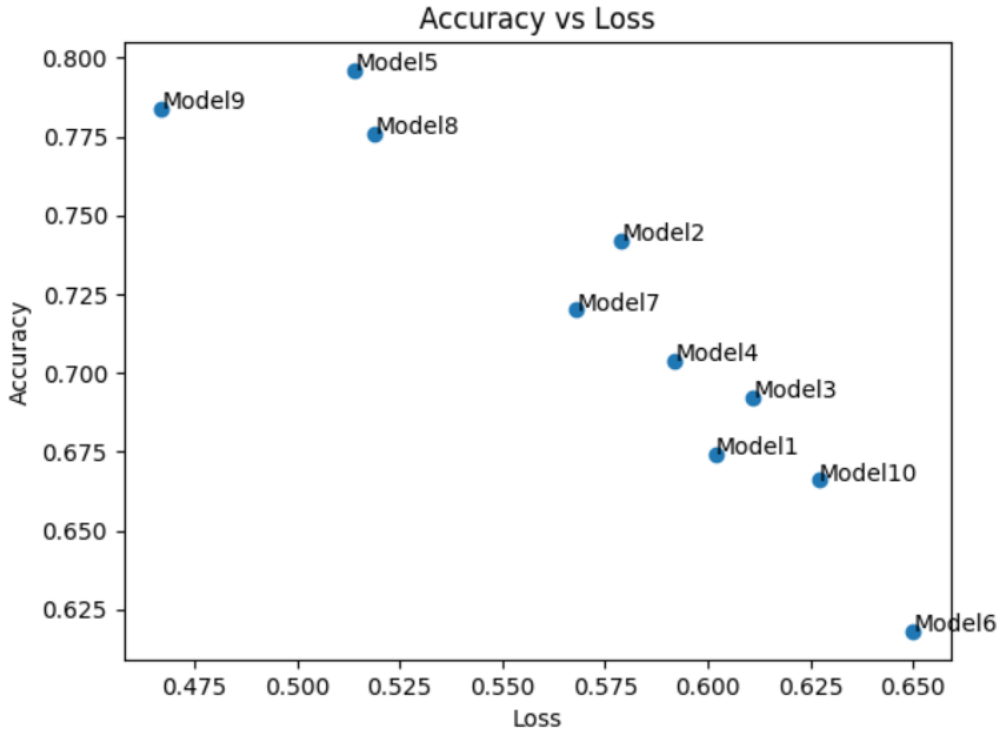
Convolutional neural network training from scratch revealed that augmentation methods and dataset size had a significant impact on model performance. The models had limited accuracy and were prone to overfitting due to the tiny dataset (1,000 samples). Nevertheless, results were much enhanced by using data augmentation and expanding the training size to 3,000 photos; the best model achieved 81% accuracy. Improved feature extraction was also a result of architectural choices like employing multiple convolutional

layers and adding padding. Overall, more data and thorough regularization are needed to match the performance of pretrained networks, even if scratch-trained models can perform well with the right tuning.

Important findings from trained models:

The VGG16 architecture, which was pre-trained on ImageNet, was used in this research to implement transfer learning. To examine the effects of data augmentation and layer freezing, experiments were carried out under two configurations—VGG Model 1 and VGG Model 2—in which the top layers of the network were swapped out for a custom classifier.

Models Built	Training sample size	Validation size	Testing sample size	Epochs	Data Augmentation	Test accuracy (%)	Test loss (%)
Model	1400	500	500	50	Yes	63.9	0.67
VGG Model1	1800	500	500	50	Yes	62.7	0.69
VGG Model2	1900	500	500	30	Yes	47.1	0.79



Conclusion

- Both models demonstrated the effectiveness of transfer learning by achieving high accuracy (~64%) even with only 1400 training data.
- Compared to VGG Model1, VGG Model2 with frozen layers and augmentation was more resilient and less likely to overfit.
- By tailoring higher-level features to the Cats vs. Dogs classification assignment, fine-tuning further improved performance.