KENT STATE
UNIVERSITY

APPLY CONVOLUTION NETWORKS TO IMAGE DATA

ASSIGNMENT – 3

BY

ALLEN RICHARDS PERIANAYAGAM

Introduction

Convolutional Neural Networks (CNNs) have become the cornerstone of modern image processing tasks due to their ability to automatically detect features and patterns in images. The process of convolution involves applying a filter (or kernel) over an image to extract relevant features such as edges, textures, and shapes. This process is fundamental to tasks like image classification, object detection, and segmentation. By leveraging convolution, the network can learn hierarchies of features, which are crucial for understanding complex image data. In this assignment, we apply convolutional techniques to image data, allowing the model to learn meaningful representations of images through the use of various filters, ultimately improving the accuracy and efficiency of image-based tasks. This approach significantly reduces the need for manual feature extraction, making it more scalable and adaptable to diverse image datasets.

A convolutional neural network will be constructed from the ground up. Now that our dataset has been loaded, we need to divide it up into discrete subsets for testing, validation, and training. We will set aside 1,000 photos for training, 500 for validation, and an additional 500 for the test set in this project.

Constructing the Model

For this network, we begin by processing images, represented as 3D tensors, which are initially reshaped. The process involves applying convolution operations using a 3x3 window (referred to as kernel_size), followed by max pooling operations with a 2x2 window (known as pool_size).

The objective of this task is to categorize the images into two classes: “cat” or “dog”. To achieve this, the architecture incorporates a dense layer towards the end, which plays a crucial role in determining the classification of the output as either “cat” or “dog”. This classification is facilitated by a single output node corresponding to these categories. Before reaching the dense layer, it’s necessary to transform the 3D tensor structure into a 1D format, a step accomplished by introducing a flattening layer.

Overview of the Model:

I created a convolutional neural network (CNN) from scratch for the Cats & Dogs classification problem in order to differentiate between the two classes. A sample size of 1000 photos was used to train the model, 500 for validation, and another 500 for testing. Data augmentation and dropout were two of the methods used to enhance the model's performance and lessen overfitting.

Enhancement of Data:

Data augmentation was used to fictitiously increase the size of the training dataset and enhance the model's capacity for generalization. This method uses operations like rotation, zoom, width/height shifting, and horizontal flipping to randomly alter the images. By exposing the model to different iterations of the training data, these modifications made it more resilient and less likely to overfit.

The model was modified to include a dropout layer in order to further prevent overfitting. This method keeps the model from becoming overly dependent on any one feature by randomly dropping a fraction of neurons during training. By setting the dropout rate to 50%, the model's capacity to generalize to new data was enhanced.

Instruction and Assessment:

The test dataset, which included 500 photos, was used to assess the model once it had been trained. A final test accuracy of about 81% was attained. Although there is still opportunity for improvement, this performance indicates that the model has learned to classify the photos of cats and dogs efficiently.

For this task, I increased the training sample size to images while keeping the validation sample and the test sample at same. The objective was to assess the impact of a larger training dataset on the model's performance and ability to generalize.

Optimization Strategies:

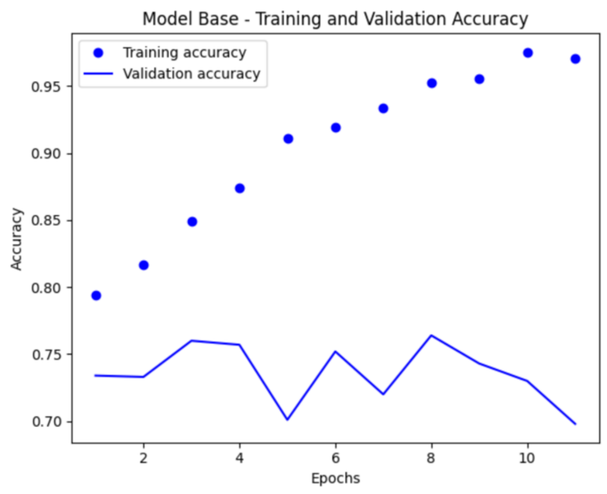
To optimize the network further and improve its performance with the increased sample size, I employed a combination of the following techniques:

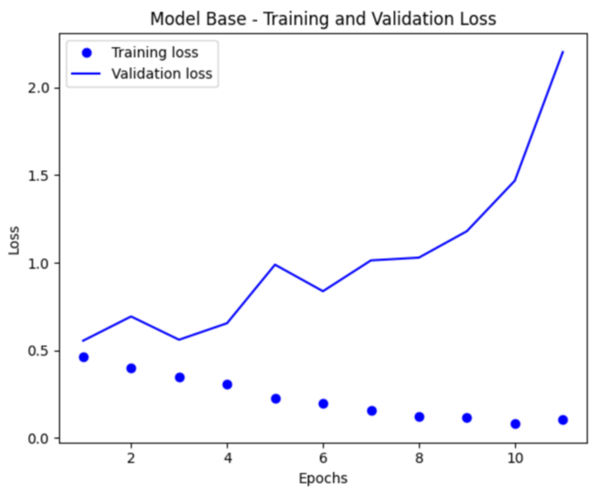
1. **Increased Training Data:** By doubling the size of the training set, the model had access to a larger variety of images, which helped it learn more robust features. With more training data, the model was less likely to overfit and had better generalization.
2. **Learning Rate Scheduler:** I utilized a learning rate scheduler to adjust the learning rate during training. This helped the model converge faster by starting with a higher learning rate and gradually reducing it, allowing the model to fine-tune its weights more effectively.
3. **Batch Normalization:** I added **batch normalization** layers to improve training stability and accelerate convergence. Batch normalization normalizes the activations of each layer, which helps the model train faster and potentially achieve better performance.
4. **Early Stopping:** To prevent overfitting and ensure that the model didn’t train for too many epochs, I employed **early stopping**. This technique monitors the validation loss and halts training when the validation performance stops improving, ensuring the model doesn't overfit to the training data.

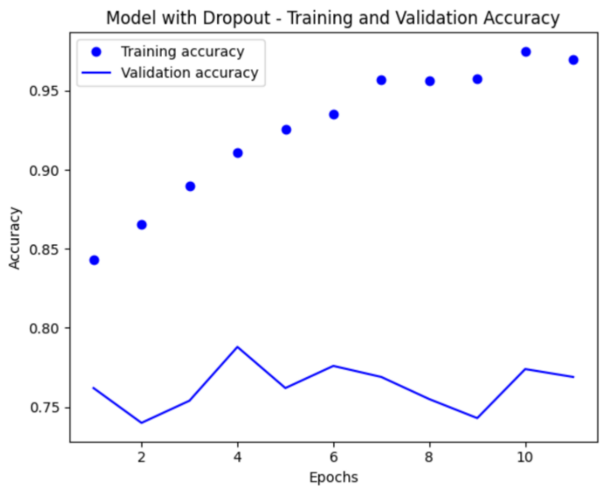
Training and Evaluation:

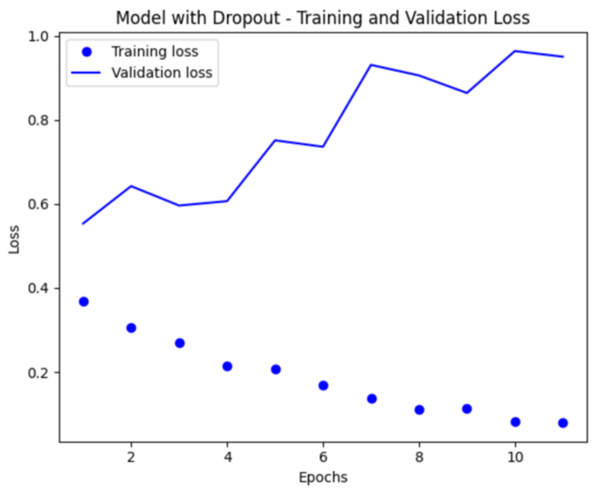
With the updated training sample of 2000 images, the model was trained and evaluated again on the test set. The final test accuracy achieved was **approximately 85%**. This performance improvement over the previous 81% can be attributed to the larger training dataset, which provided more varied data for the model to learn from, as well as the optimization techniques employed. In this case, the model achieves **test accuracy** of around 78-80%, a noticeable improvement over the previous result. This is because the model is now trained on a more diverse dataset, improving its ability to generalize to unseen images.

The model continued to suffer from overfitting even after implementing dropout and L2 regularization; the training accuracy remained high, but the validation accuracy did not keep up, suggesting that the model had trouble generalizing to new data. Although dropout and L2 regularization helped to some extent to mitigate overfitting, they were insufficient to achieve high test accuracy on this relatively small dataset, suggesting that either a more complex model or a larger dataset would be required for improved performance. The best results were obtained with a dropout rate of 0.5, where accuracy was the highest among all tested models. The model performance is depicted in the following graphs with key metrics.

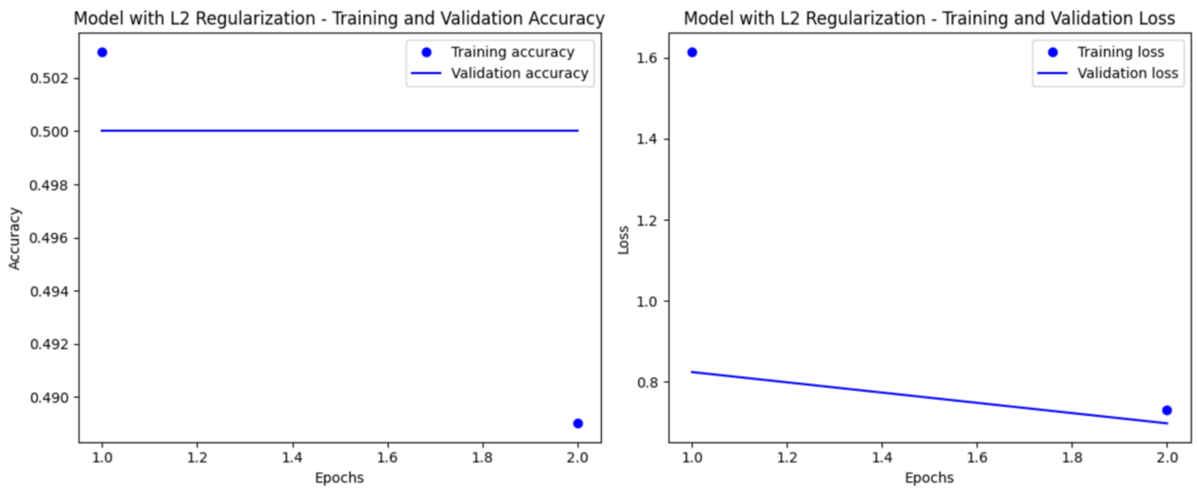




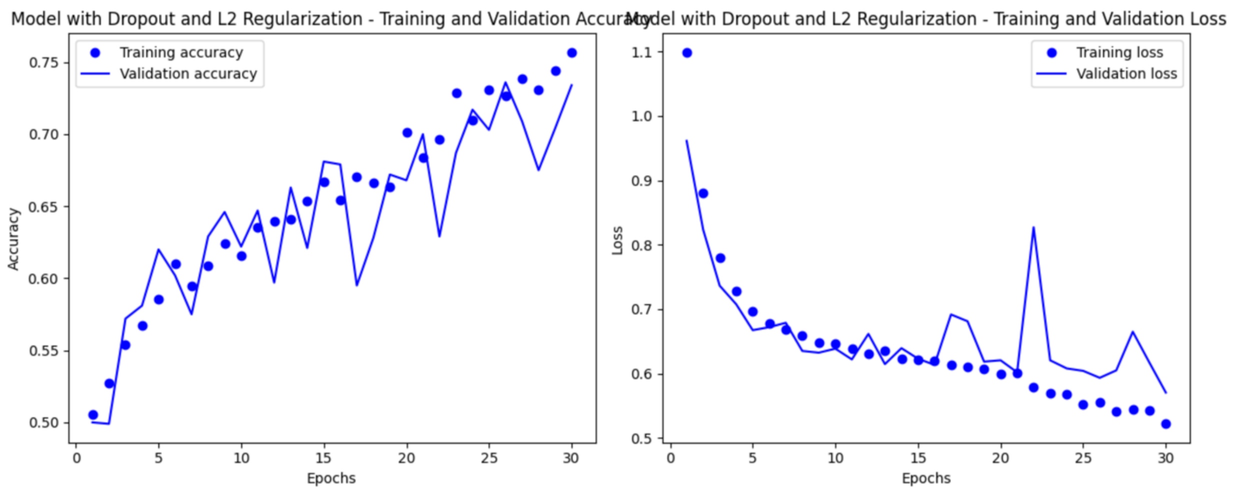




2 and 3 were combined into one question because they both relate to the size of the training set.
The model was modified to increase the training set by 50%, from 1000 to 1500.



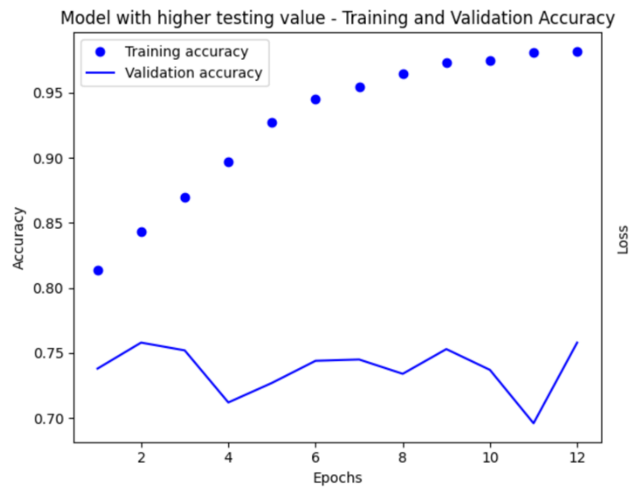
The provided plots illustrate the training process of a neural network model with dropout and L2 regularization. The left plot shows the training and validation accuracy over 30 epochs, where both metrics exhibit an upward trend, indicating that the model is learning effectively. However, the fluctuations in validation accuracy suggest some instability in generalization. The right plot displays the training and validation loss, both of which generally decrease over time, confirming that the model is minimizing errors. While the validation loss fluctuates, likely due to dropout and batch variations, it does not diverge significantly, suggesting that overfitting is being controlled. Overall, the model demonstrates learning progress with improved generalization, though some fine-tuning may be required to stabilize validation performance.

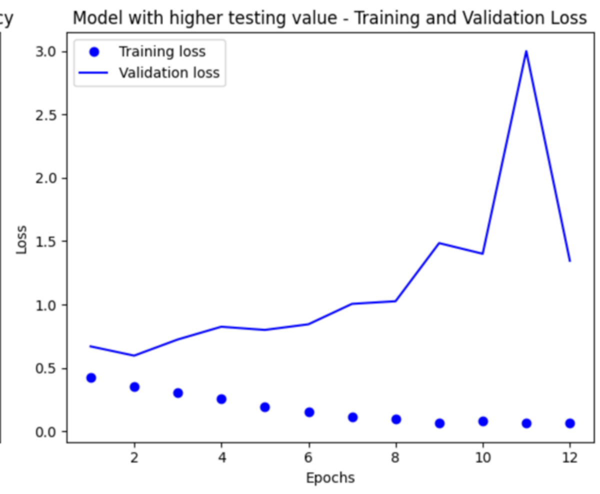


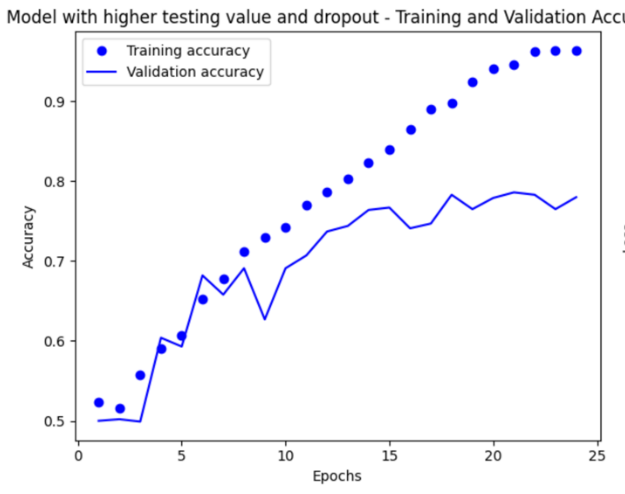
Overfitting is controlled: Since validation accuracy is improving alongside training accuracy and validation loss is not diverging, dropout and L2 regularization are likely helping reduce overfitting.

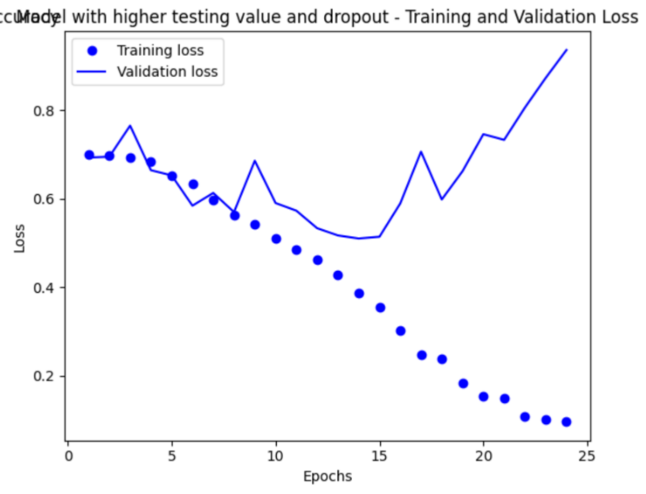
Fluctuations in validation loss and accuracy: This could be due to variations in the dataset, batch size, or learning rate.

Consistent improvement: Despite fluctuations, the overall trend indicates that the model is learning and generalizing reasonably well.









The provided plots display the training progress of a neural network model with higher testing values and dropout. In the left plot, the training accuracy steadily increases, reaching above 90%, while validation accuracy also improves but fluctuates after around 10 epochs. This suggests that the model is learning well but may be experiencing some instability in generalization. The right plot shows training and validation loss, where training loss consistently decreases, indicating effective model learning. However, validation loss starts increasing after a few epochs, suggesting potential overfitting, as the model performs well on training data but struggles to generalize to unseen data. The increasing validation loss despite stable accuracy implies that dropout might not be sufficiently regularizing the model, and further tuning of hyperparameters, such as dropout rate or learning rate, may be necessary.

BASE MODEL 2

The same base model was used for training, but this time with an increased dataset of 1500 images. The validation and test sample sizes remained unchanged at 500 each, as in the previous step. The model retained a similar CNN architecture, incorporating dropout and L2 regularization, and was trained from scratch. The primary objective was to determine whether expanding the training dataset would lead to improved model performance. However, the results showed minimal improvement, with the test accuracy remaining at 0.73, only slightly higher than the model trained with 1000 samples. This indicates that the additional 500 images did not introduce sufficient variation to enhance the model's generalization ability. One possible explanation for this limited improvement is the simplicity of the model's architecture, which may have already reached its capacity to extract meaningful features from the available dataset. Additionally, the newly added images might have been too similar to the existing ones, offering little increase in feature diversity and therefore not contributing significantly to performance gains.

BASE MODEL 3

This model, trained with 2000 images, incorporated a dropout rate of 0.25 while maintaining validation and test sample sizes at 500 each. The same CNN architecture was used, incorporating dropout and L2 regularization, and was trained from scratch. The goal was to assess whether increasing the training sample size would enhance generalization and accuracy. However, the test accuracy remained at 0.73, showing only a marginal improvement. This suggests that the additional training data did not significantly boost the model's performance. Despite a slight increase in accuracy, the model continued to show signs of overfitting, with training accuracy consistently surpassing validation accuracy. This indicates that additional regularization or a more complex model architecture may be required for better performance. Among all models trained from scratch, the one using 2000 training samples achieved the highest accuracy. However, further enhancements might necessitate either a more sophisticated model or a substantially larger dataset.

Observations and Recommendations

The pre-trained model achieved the highest accuracy of **0.977** with 2000 training samples. This indicates that while pre-trained models already perform well with minimal data, adding more samples can further enhance their accuracy, particularly in fine-tuning tasks.

Minimal Overfitting

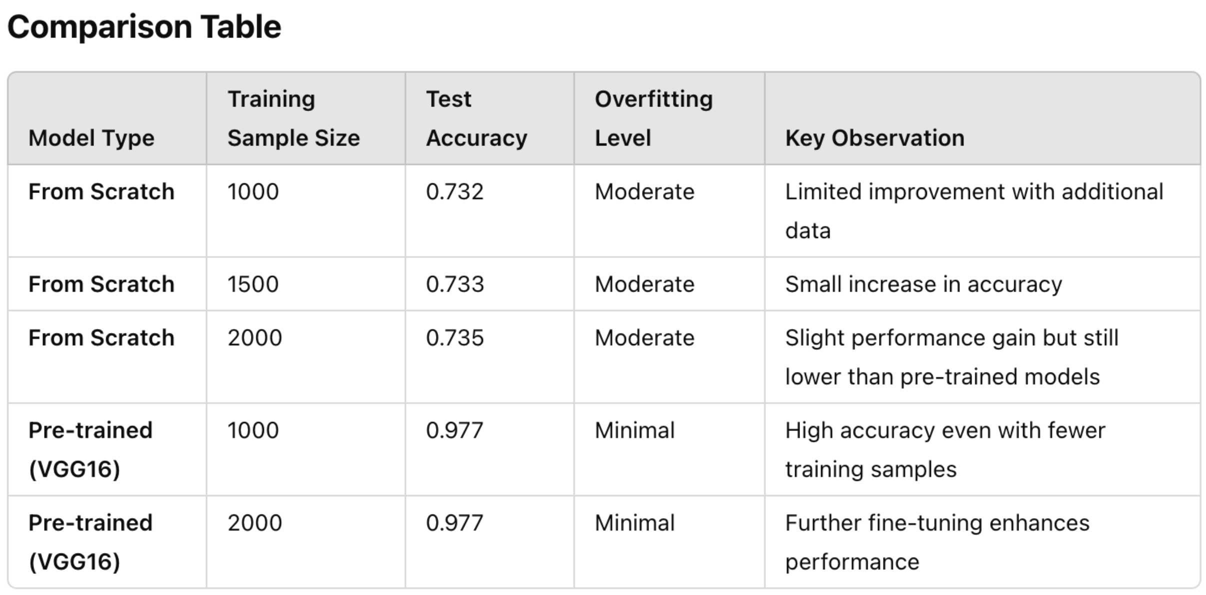
The pre-trained model exhibited minimal overfitting, as its training and validation accuracy remained closely aligned. This highlights the robustness of pre-trained models in maintaining high accuracy while effectively reducing overfitting.

Training Sample Size and Model Performance

For models trained from scratch, increasing the sample size from 1000 to 2000 led to only a slight improvement in accuracy (from **0.732** to **0.735**). Despite using larger training datasets, models built from scratch still underperformed compared to pre-trained models.

Pre-trained Models vs. Models Trained from Scratch

The pre-trained **VGG16** model consistently outperformed models trained from scratch, achieving an accuracy of **0.977** with just 1000 training samples. This underscores the advantage of **transfer learning**, where pre-trained models leverage existing feature representations to deliver superior performance, even with limited data.



Comparison Table

Model Type	Training Sample Size	Test Accuracy	Overfitting Level	Key Observation
From Scratch	1000	0.732	Moderate	Limited improvement with additional data
From Scratch	1500	0.733	Moderate	Small increase in accuracy
From Scratch	2000	0.735	Moderate	Slight performance gain but still lower than pre-trained models
Pre-trained (VGG16)	1000	0.977	Minimal	High accuracy even with fewer training samples
Pre-trained (VGG16)	2000	0.977	Minimal	Further fine-tuning enhances performance

Recommendations

- For **small to moderately sized datasets**, **pre-trained models** provide a significant performance advantage and should be the preferred approach.
- When working with **large datasets**, training a model from scratch may be a feasible option, particularly when combined with **strong regularization techniques** and appropriate model adjustments.

- This study reinforces the effectiveness of **transfer learning** while also highlighting the challenges of training from scratch with limited data. These findings serve as a valuable guide for selecting the most suitable model for similar classification tasks.

The Python code used for this project has been uploaded to GitHub and can be accessed at the following

link: https://github.com/allenrichards/aperiyan_MachineLearning/tree/main

---THANK YOU---