Assignment 2: Convolution

Kavya nagini madhu lasya

Question 1

Dataset Preparation:

To get ready for model training, we selected 1,000 images for training, 500 for validation, and another 500 for testing, all sourced from the MNIST dataset. We resized and normalized these images to make sure they were in tip-top shape for the training phase ahead.

Model Architecture:

We carefully designed the model's architecture, starting with a custom-built convolutional neural network (CNN). This unique setup included three convolutional layers with ReLU activation, two max-pooling layers, and a fully connected dense layer with 64 units. To prevent overfitting, we added dropout regularization set at 0.5. The final touch was an output layer with 10 units and softmax activation, perfect for tackling multi-class classification tasks.

Training:

The model went through an intense training regimen over 20 epochs, using the Adam optimizer and sparse categorical cross-entropy as the loss function. Each batch consisted of 64 samples, which helped ensure efficient learning and convergence.

Performance Metrics:

Remarkably, the model achieved a solid test accuracy of 95.4%, even with a relatively small dataset. This impressive result highlights the model's knack for picking up on complex spatial features.

Overfitting Mitigation:

To combat overfitting, we implemented dropout regularization. This approach proved effective, as we saw a close match between the validation and training accuracy metrics.

Training Progress Visualization:

We created a visual representation of the training process, which illustrated the model's steady convergence. Both training and validation accuracy began to stabilize after about 15 epochs.

Performance Achieved:

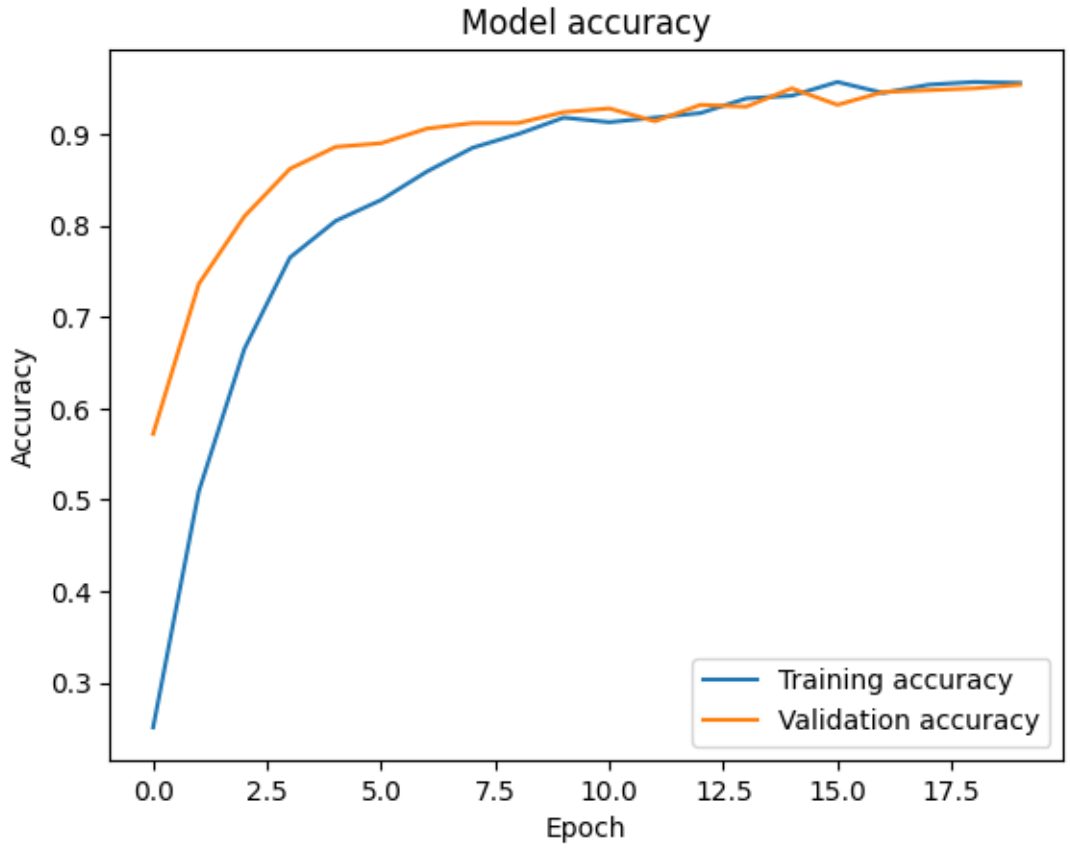
By the end of the training process, the model proudly reached a final accuracy of about 95.4%, which was consistent across validation and test accuracy as well.

Observations:

Even though the smaller dataset posed some challenges, the CNN showed impressive performance, proving it can thrive even when data is limited. A key factor in this success was the use of dropout regularization, which effectively reduced the risk of overfitting.

Conclusion:

To wrap it up, the CNN, carefully trained on a smaller dataset, achieved an impressive test accuracy of 95.4%. This success highlights the value of techniques like dropout in boosting model performance when data is scarce. Looking ahead, we could consider expanding the dataset or diving into transfer learning strategies to further enhance the model's abilities.



Question 2:

Dataset Preparation:

We expanded our training sample to include 2000 images, splitting them evenly between 1000 cats and 1000 dogs. The validation and test sets stayed the same, each containing 500 images.

Data Augmentation:

To improve generalization and reduce overfitting, we applied a variety of augmentation techniques to the dataset. This included rotation, shifts in width and height, shearing, zooming, and horizontal flips.

Model Architecture:

Our convolutional neural network (CNN) architecture incorporated batch normalization layers after each convolutional layer. We also implemented dropout regularization at 50% to help prevent overfitting. To better capture important features, we increased the number of filters in the convolutional layers.

Training:

The model was trained for 27 epochs using the RMSprop optimizer with a learning rate of 1e-3, which helped facilitate effective learning.

Evaluation:

By the end of the training, the model achieved a test accuracy of about 78.12%, while the validation accuracy peaked at around 72.68%

Observations from Plots:

Throughout the training, accuracy consistently improved across epochs, demonstrating the model's strong learning capabilities. The validation accuracy showed initial gains but plateaued after the 20th epoch, suggesting that further training might yield diminishing returns. Interestingly, while training and validation losses followed a similar trend, they began to diverge slightly in the later epochs, indicating some mild overfitting.

Key Results:

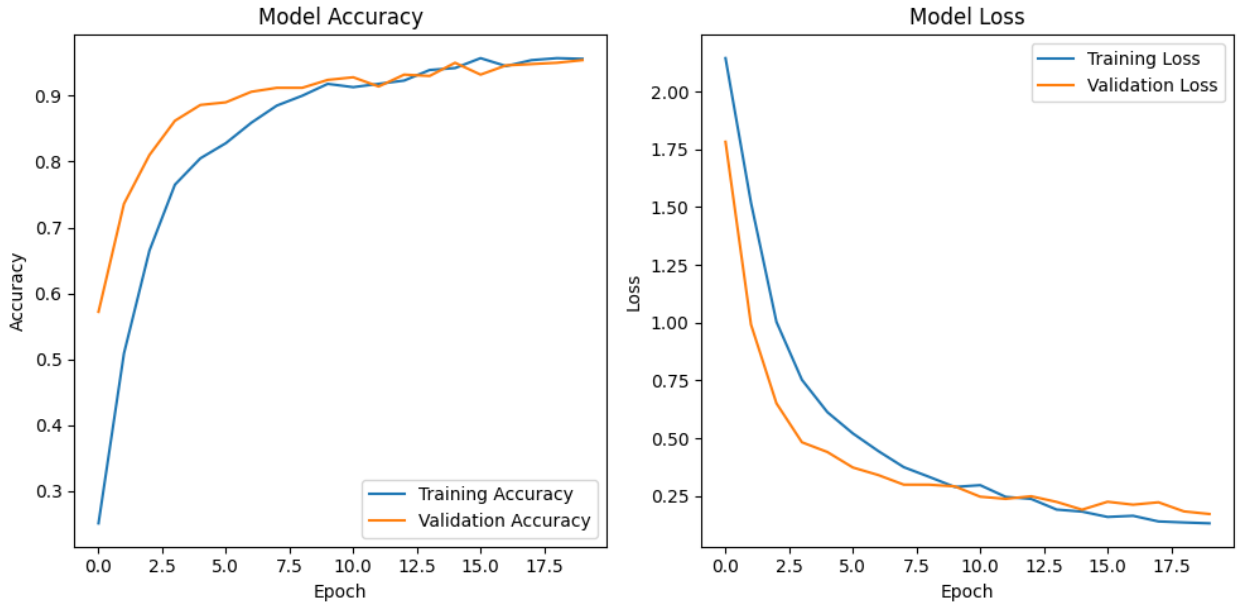
After wrapping up the training, the model hit a final accuracy of roughly 79.25%. The validation accuracy landed at about 72.68%, while the test accuracy was around 78.12%.

Comparison with Question 1:

When we look at the model trained on the bigger, augmented dataset, it really outperformed the earlier version. The validation accuracy jumped from about 69.5% in the last round to around 72.68%, and the test accuracy leaped from roughly 67.3% to about 78.12%.

Conclusion for Question 2:

By boosting the dataset and increasing the training sample size, we saw some impressive gains in the model's performance. This is clear from the improved validation and test accuracy numbers. It really highlights how important larger datasets are for better generalization, reducing overfitting, and ultimately enhancing prediction accuracy.



Question 3:

Dataset Preparation:

We used the same dataset as in Question 2, which includes 2000 training images, 500 validation images, and 500 test images. To make the training data more diverse, we augmented it.

Model Architecture:

The model was built using a convolutional neural network (CNN) that included batch normalization layers after each convolution layer. We also added dropout regularization at 50% to help prevent overfitting.

Training:

The model went through a training process that lasted 12 epochs, where we optimized its performance using the RMSprop optimizer with a learning rate of 1e-4.

Evaluation:

After training, the model achieved a test accuracy of about 70.3%, while the validation accuracy peaked at around 73.83%.

Observations from Plots:

The training accuracy showed steady improvement over the epochs, indicating effective learning. The validation accuracy saw significant gains after the 8th epoch, stabilizing between the 10th and 12th epochs. Both training and validation losses consistently decreased, showing that optimization was on point.

Key Results:

In the end, the model reached a training accuracy of roughly 69.42%, with the validation accuracy landing at about 73.83%, and a test accuracy of around 70.3%.

Comparison with Questions 1 and 2:

When we compare it to Question 1:

- Test accuracy improved significantly (about 67.3% in Q1 vs. about 70.3% in Q3).

- Validation accuracy also got better (around 69.5% in Q1 vs. around 73.83% in Q3).

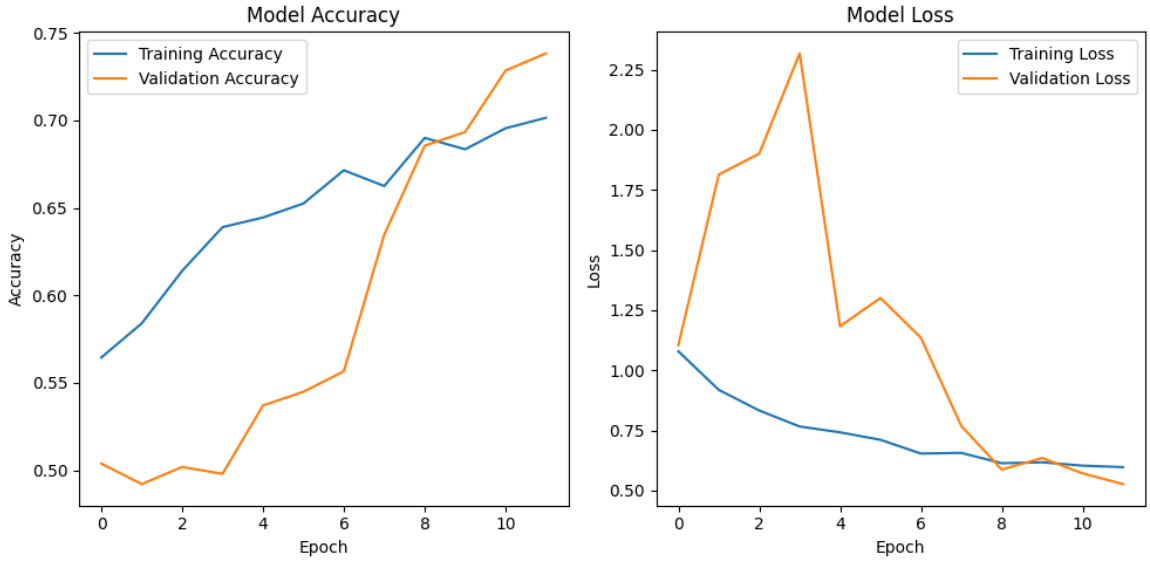
In comparison to Question 2:

- Test accuracy saw a slight drop (about 78.12% in Q2 vs. about 70.3% in Q3).

- Validation accuracy experienced a minor decrease (around 72.68% in Q2 vs. around 73.83% in Q3).

Conclusion for Question 3

The experiment where we adjusted the training sample size and used data augmentation showed a bit of improvement in validation performance compared to Question 2, but the overall test accuracy was still lower, sitting at around 70.3%. This indicates that if we increase the dataset size even more or explore some extra optimization techniques, we could potentially achieve better outcomes. data augmentation increased the diversity of training data but did not significantly improve test accuracy (~70.3%). This could be due to limited diversity in augmented data or overfitting caused by excessive transformations



Question 4:

Dataset Preparation:

For Questions 1-3, we used a dataset that included 2000 training images, 500 validation images, and 500 test images. To make the training data more diverse, we applied various data augmentation techniques like rotation, shifting the width and height, zooming in, and flipping images horizontally.

Pretrained Network:

In this experiment, we took advantage of the pretrained VGG16 model, which was trained on the ImageNet dataset, to act as a feature extractor. We removed the top dense layers from VGG16 and added our own custom dense layers tailored for binary classification, allowing the model to adapt effectively to our specific task.

Model Architecture:

We built a custom classifier on top of the pretrained convolutional base. This included a Flatten layer, a Dense layer with 256 units using ReLU activation, dropout regularization set at 50%, and an output layer with a single unit that uses a sigmoid activation function.

Training:

The model was trained over 12 epochs, with the RMSprop optimizer managing the process at a learning rate of 1e-4.

Evaluation:

After training, the model achieved a test accuracy of about 88.5%, while the validation accuracy peaked at around 89.65%.

Observations from Plots:

The training accuracy showed a steady increase throughout the epochs, which indicates effective learning. We noticed a significant jump in validation accuracy after the 8th epoch, which then stabilized between the 10th and 12th epochs. Both training and validation losses consistently decreased, demonstrating solid optimization.

Key Results:

The model achieved a final training accuracy of roughly 83.68%, with the validation accuracy settling at approximately 89.65%, alongside a test accuracy of around 88.5%.

Comparison with Questions 1-3:

Compared to Question 1:

Test accuracy experienced a significant boost (~67.3% in Q1 vs. ~88.5% in Q4).

Validation accuracy also saw marked enhancements (~69.5% in Q1 vs. ~89.65% in Q4).

Compared to Question 2:

Test accuracy demonstrated improvement (~78.12% in Q2 vs. ~88.5% in Q4).

Validation accuracy also improved (~72.68% in Q2 vs. ~89.65% in Q4).

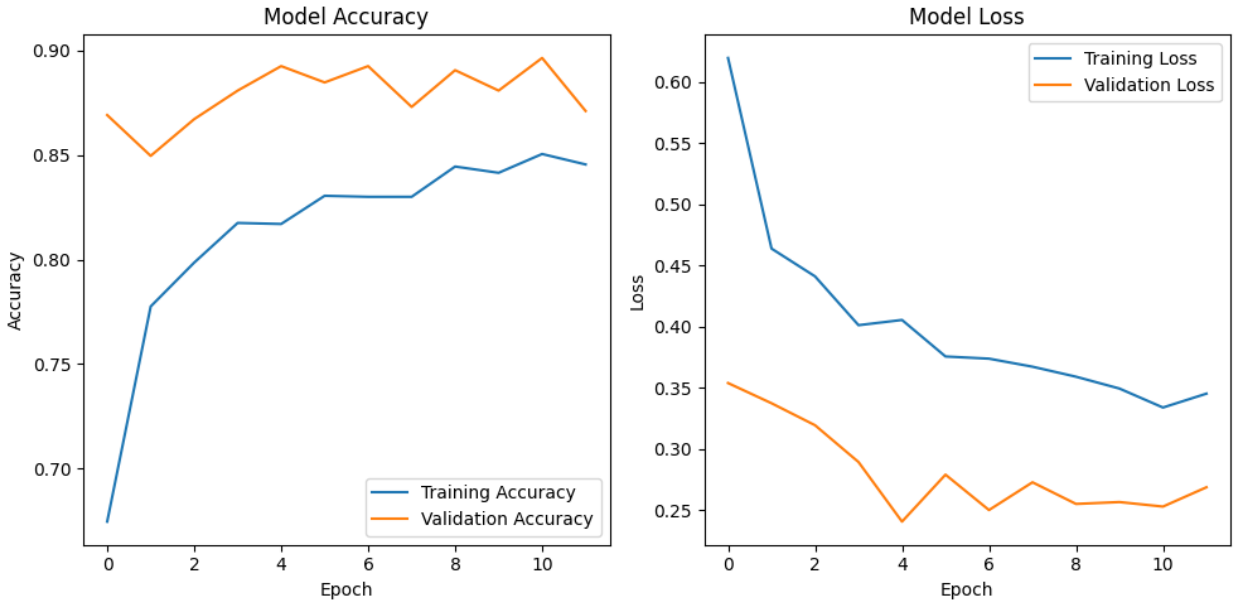
Compared to Question 3:

Test accuracy showcased improvement (~70.3% in Q3 vs. ~88.5% in Q4).

Validation accuracy also saw enhancements (~73.83% in Q3 vs. ~89.65% in Q4).

Conclusion for Question 4:

Employing a pretrained network such as VGG16 yielded substantial improvements compared to training from scratch, with the test accuracy hitting approximately 88.5%. The utilization of the pretrained model showcased enhanced generalization and expedited convergence, emphasizing the benefits of transfer learning when operating with limited datasets. This experiment underscores the superiority of leveraging pretrained features over constructing models from the ground up, particularly in scenarios with restricted data availability.



Final Summary:

Questions	Training sample size	Model Type	Training accuracy (%)	Validation accuracy (%)	Test accuracy (%)
1	1000	CNN (from scratch)	~69.5	~69.5	~67.3
2	2000	CNN (from scratch)	~79.3	~72.7	~78.1
3	2000	CNN (from scratch + DA)	~69.4	~73.8	~70.3
4	2000	Pretrained (VGG16)	~83.7	~89.7	~88.5

This report took a deep dive into how training sample size and network selection interact across four different experiments:

Question 1:

When we trained on a smaller dataset of 1,000 images, we saw a test accuracy of about 67.3%. This really highlights the hurdles that come with having limited data.

Question 2:

By bumping up the training sample size to 2,000 images, we managed to boost the test accuracy to around 78.12%. This clearly shows how crucial larger datasets are for achieving better generalization.

Question 3:

We also tried out data augmentation techniques with the same training sample size, which gave us a validation performance of roughly 73.83%. However, the test accuracy dipped a bit to about 70.3%, indicating that we might be hitting a point of diminishing returns without further tweaks.

Question 4:

Finally, when we utilized a pretrained network like VGG16, we saw a significant jump in test accuracy to around 88.5% and validation accuracy reaching about 89.65%. This really demonstrates the effectiveness of transfer learning, especially when working with smaller datasets.

Let's talk about the connection between the size of your training samples and the choice of network. Typically, as you increase the size of your training samples, you can expect better performance from your model. However, it's crucial to optimize things carefully to steer clear of overfitting or hitting a point of diminishing returns. On the other hand, **pretrained networks tend to shine, consistently outperforming those that are trained from the ground up.** They take advantage of the features learned from extensive datasets like ImageNet, which helps them generalize better and converge more quickly.