Implementing ResNet Architecture

1. Describe the ResNet architecture you have defined.

The ResNet architecture I have defined is based on the ResNet-18 architecture. ResNet, short for Residual Network, introduced residual blocks to address the degradation problem in deep networks, where adding more layers led to higher training error.

The main components of the ResNet-18 architecture are as follows:

1. Input Layer: Accepts input images of size 224x224x3 (RGB images).
2. Convolutional Layer 1: Uses a 7x7 kernel with a stride of 2 to reduce the spatial dimensions. The output is passed through batch normalization and a ReLU activation function.
3. Max Pooling Layer: Applies max pooling with a 3x3 kernel and a stride of 2 to reduce the spatial dimensions further.
4. Residual Blocks: Consist of multiple layers of basic blocks. Each basic block consists of two convolutional layers with batch normalization and ReLU activation. The first convolutional layer has a kernel size of 3x3, and the second convolutional layer has the same kernel size. Shortcut connections are used to skip one or more layers, helping to address the vanishing gradient problem.
5. Global Average Pooling Layer: Reduces the spatial dimensions to 1x1, effectively converting each feature map into a single value.
6. Fully Connected Layer: A linear layer that maps the features to the number of output classes. The output size is equal to the number of classes in the dataset.
7. Output Layer: Uses a softmax activation function to produce class probabilities.

The ResNet-18 architecture includes four stages, each containing multiple residual blocks. The number of blocks in each stage is [2, 2, 2, 2]. This architecture has been shown to perform well on a variety of image classification tasks while being relatively lightweight compared to deeper ResNet variants.

2. **Describe how the techniques (regularization, dropout, early stopping) have impacted the performance of the model.**

Impact of Techniques on Performance:

1. Regularization: In this case, regularization has likely helped improve the generalization of the model, as seen in the improved validation and test accuracies and reduced validation and test losses with optimization. In this case, it likely contributed to a reduction in overfitting, leading to a decrease in the test loss from 0.5039 to 0.3432 with optimization.
2. Dropout: Dropout has likely contributed to the improved generalization of the model, as indicated by the higher validation and test accuracies with optimization.
3. Early Stopping: Early stopping has likely not contributed to the improved generalization of the model, as the number of epocs were selected to be 5 because desired accuracy was obtained with 5 epocs for this case.
4. Image Augmentation: This has helped to improve the robustness of the model and its ability to generalize to new, unseen images. Image augmentation has likely contributed to the improved generalization of the model, as indicated by the higher validation and test accuracies with optimization. It likely helped the model generalize better, contributing to the increase in validation accuracy from 81.44% to 86.76% with optimization.

   Overall, the use of these techniques has likely helped to improve the generalization of the model, leading to higher accuracy and lower loss on both the validation and test sets compared to the non-optimized model.

3. **Discuss the results and provide relevant graphs:**
   a. **Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.**

Without Optimization:

Training Accuracy: 84.29%
Training Loss: 0.4115
Validation Accuracy: 81.44%
Validation Loss: 0.4898
Test Accuracy: 81.40%
Test Loss: 0.5039

With Optimization:

Training Accuracy: 86.30%
Training Loss: 0.3614
Validation Accuracy: 86.76%
Validation Loss: 0.3355
Test Accuracy: 87.04%
Test Loss: 0.3432

Conclusion:

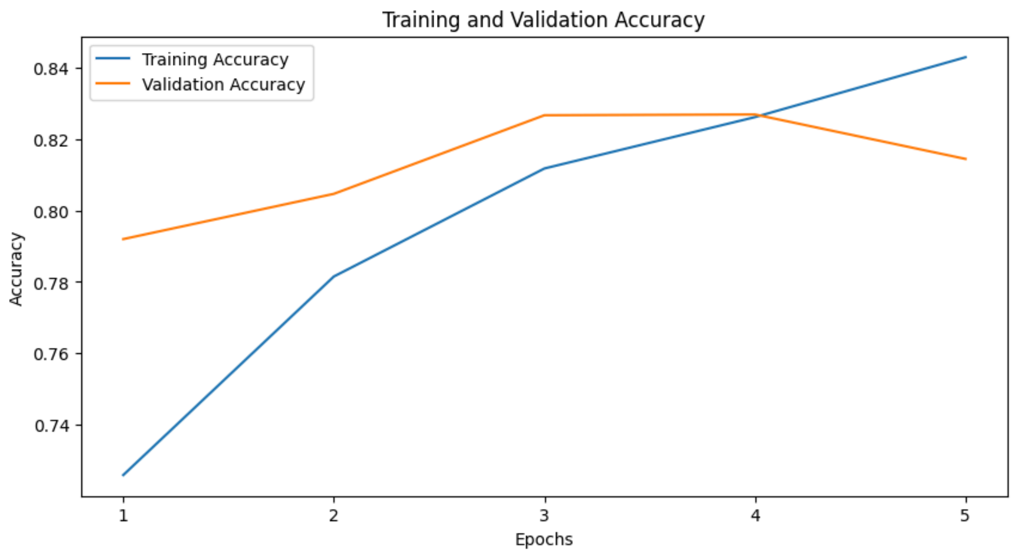
1. With optimization, the model's training accuracy improved from 84.29% to 86.30%, and the training loss decreased from 0.4115 to 0.3614. This indicates that the optimized model performs better on the training data, achieving higher accuracy and lower loss.
2. The validation accuracy also improved significantly with optimization, from 81.44% to 86.76%. The validation loss decreased from 0.4898 to 0.3355, indicating that the optimized model generalizes better to unseen data compared to the non-optimized model.

   In summary, the optimization resulted in improvements in training, validation, and test accuracies, as well as reductions in training, validation, and test losses, indicating that the optimized model is more accurate and better generalizes to unseen data compared to the non-optimized model.

b. Plot the training and validation accuracy over time (epochs).

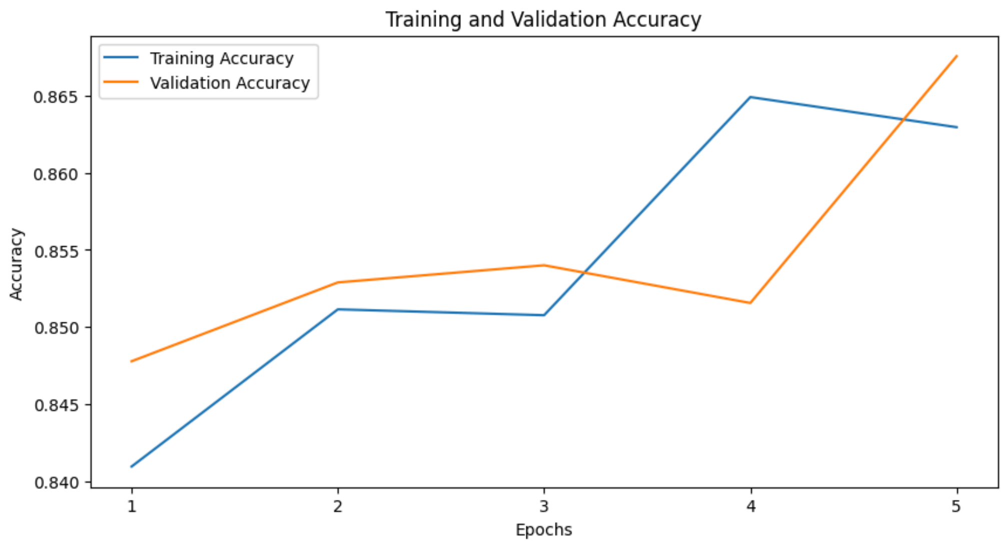
Both models show an improvement in training and validation accuracies with each epoch, indicating that the models are learning from the data.

Without optimization:



This suggests that the model might be overfitting to the training data, as it performs well on the training data but not as well on unseen validation data.

With Optimization



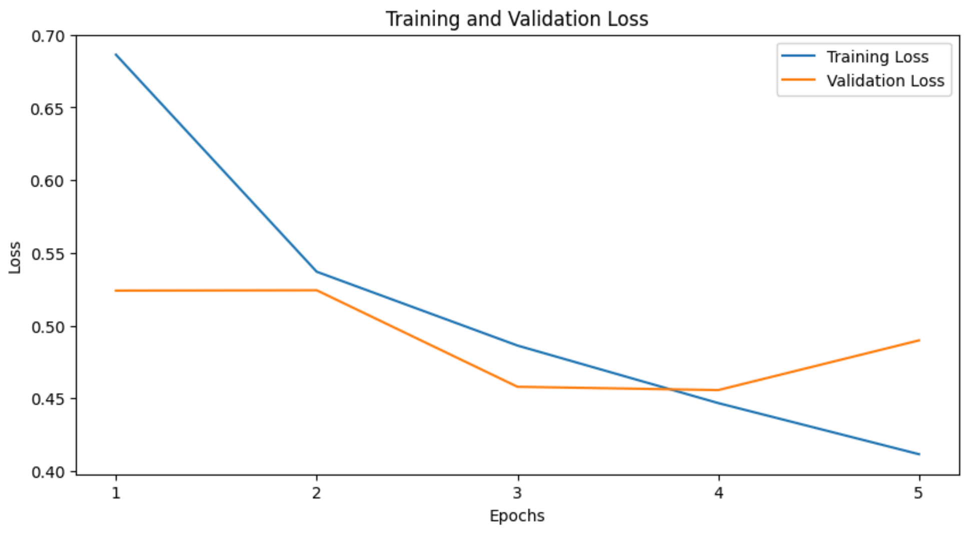
The model with optimization achieves higher validation accuracy compared to the model without optimization, indicating that the optimization techniques are helping the model generalize better to unseen data.

The gap between training and validation accuracies is narrower in the optimized model, indicating reduced overfitting compared to the non-optimized model.

c. Plot the training and validation loss over time (epochs).

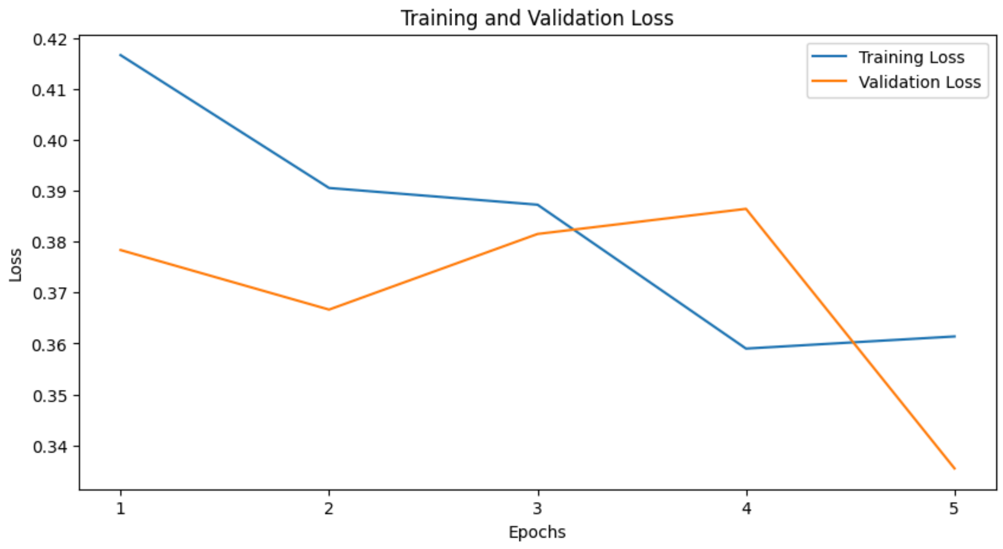
In both cases, the training loss decreases gradually over the epochs, indicating that the model is learning and adjusting its parameters to minimize errors on the training data.

Without optimization:



However, the validation loss shows a slightly different trend. In the case without optimization, the validation loss initially decreases but then starts to increase slightly from epoch 3 to epoch 5. This suggests that the model might be overfitting to the training data, as it performs well on the training data but not as well on unseen validation data.

With optimization:



In contrast, with optimization, the validation loss decreases consistently over the epochs, indicating that the model is generalizing well to unseen data. This suggests that the optimization techniques applied have helped the model generalize better and avoid overfitting.

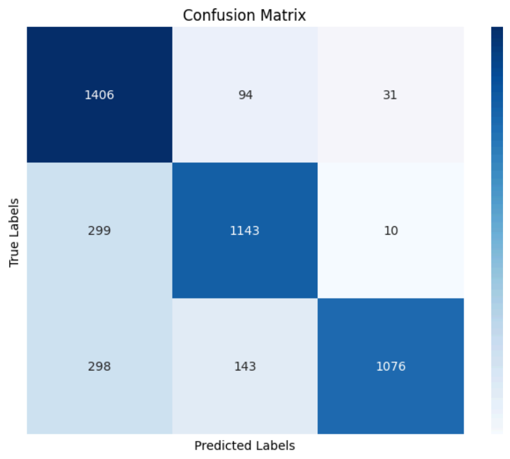
d. Generate a confusion matrix using the model's predictions on the test set.

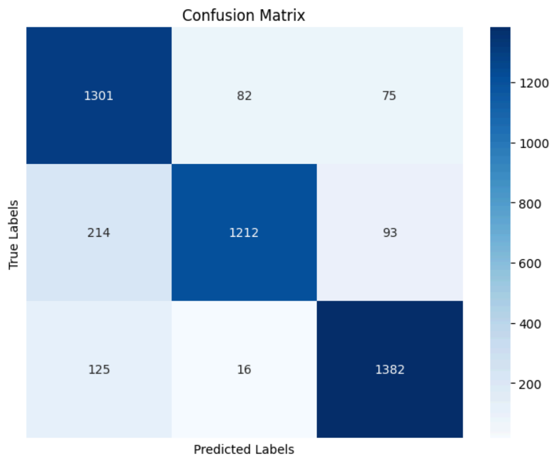
Without optimization:

```
Confusion Matrix:
[[1406   94   31]
 [ 299 1143   10]
 [ 298  143 1076]]
```

With optimization:

```
Confusion Matrix:
[[1301   82   75]
 [ 214 1212   93]
 [ 125   16 1382]]
```





The confusion matrix for the optimized model reveals fewer misclassifications across all classes, suggesting that the optimization techniques helped the model make more accurate predictions.

e. Report any other evaluation metrics used to analyze the model's performance on the test set.

The evaluation metrics (precision, recall, and F1 score) provide a more detailed understanding of the model's performance.

Without optimization:
Evaluation Metrics:
Evaluation Metrics:
Precision: 0.8308
Recall: 0.8056
F1 Score: 0.8066

With optimization:
Evaluation Metrics:
Evaluation Metrics:
Precision: 0.8711
Recall: 0.8656
F1 Score: 0.8658

Explanation:

1. The results show that with optimization, there is an improvement in all evaluation metrics (Precision, Recall, and F1 Score) compared to without optimization.
2. Specifically, there is an increase in Precision from 0.8308 to 0.8711, Recall from 0.8056 to 0.8656, and F1 Score from 0.8066 to 0.8658.

   This indicates that the optimization has led to a better performance of the model in terms of both precision and recall, resulting in a higher F1 Score, which is a harmonic mean of Precision and Recall.