Reyyan Qureshi
169033850

Introduction

In this assignment, I explored the concept of transfer learning by using a pre-trained ResNet18 model to recognize my own handwritten digits. Transfer learning uses the knowledge gained from a large dataset to improve performance on a smaller dataset, like I am doing so in this assignment. This approach is mainly beneficial when limited data is available, as it reduces training time and enhances accuracy.

Dataset Preparation

To create a handwritten digit dataset, I wrote the numbers 0 through 9 twelve times on paper, scanned the sheets, and extracted individual digit images. Each image was preprocessed to match the MNIST dataset specifications: grayscale format and resized to 28x28 pixels. The dataset was organized into three subsets, a training set containing 80 images with 8 images per digit, and a validation set alongside a test set that each contained 20 images with 2 images per digit. This distribution made sure that the model had sufficient data to learn from while maintaining separate sets for validation and unbiased testing.

Model Selection

I selected ResNet18 for this assignment because of its compatibility with the handwritten digit dataset and its efficient architecture. Despite working with smaller, 28x28 pixel grayscale images, ResNet18's learning framework effectively captures essential features without overcomplicating the model, which is important given the limited dataset size of 80 training images. By modifying the first convolutional layer to accept single-channel input, ResNet18 worked well with the grayscale digit images.

Hyperparameter Tuning

Learning Rate

I experimented with learning rates of 0.001 for Model 1, where only the last layer was fine-tuned and 0.0001 for Model 2, where the last two layers were fine-tuned. A higher learning rate for Model 1 allowed for quicker convergence, while a lower learning rate for Model 2 provided finer adjustments during the more extensive fine-tuning process, which prevented overfitting.

Batch Size

Given the small size of the dataset, I decided to use a batch size of 8 for training and 2 for validation and testing. Smaller batch sizes made sense for small datasets as they allow the model to update weights more frequently and understand subtle patterns without overwhelming the memory.

Number of Epochs

I set the number of training epochs to 20 for both models. At first, I experimented with 10 epochs at first, but realized the overall accuracy it was giving me back was too low, which resulted in me switching back to the format the code in the slides presented. This duration was sufficient to see meaningful improvements in accuracy without leading to overfitting, as monitored through validation accuracy.

Challenges

One of the primary challenges was preventing overfitting, especially given the limited dataset size. Fine-tuning more layers increased the risk of the model memorizing the training data. To address this, especially in the fourth evaluation question, I had only the last two layers fine-tuned. I would also

adjust the learning rates and monitor validation accuracy to make sure the model reacted well to unseen data.

Training and Fine-tuning

Model 1: Fine-tuning Only the Last Layer

In Model 1, I froze all layers of ResNet18 except the final layer. Over 20 epochs, the training accuracy improved from 12.50% to 73.75%, and validation accuracy reached 60.00%. The final test accuracy was 45.00%, which showed moderate improvement, but suggesting room for further fine-tuning.

Model 2: Fine-tuning the Last Two Layers

Model 2 involved fine-tuning the last two layers. This allowed deeper adjustments to the model's feature so it could capture more detail of the handwritten digits. Training accuracy spiked from 11.25% to 95.00%, and validation accuracy averaged out to 80.00%. The final test accuracy achieved was 75.00%, showing significant improvement over Model 1 and highlighting the benefits of fine-tuning additional layers.

MNIST & Custom Test Sets:

The pre-trained ResNet18 model achieved accuracies of 10.12% on the MNIST test set and 20.00% on the custom handwritten digit test set. The low performance is due to ResNet18 being originally trained on the ImageNet dataset, which contains high-resolution RGB images, different from MNIST's grayscale, 28x28 images. The features optimized for complex, natural images in ImageNet fail to generalize effectively to MNIST's simpler digit patterns. In order to achieve better accuracy, training ResNet18 from scratch on MNIST would be needed and that requires a huge sum of resources and storage, as MNIST's dataset size of 60,000 training images would lead to lengthy training times. By using a pre-trained model and fine-tuning, storage and resource demands are reduced while using ResNet18 for more efficient learning focused towards smaller datasets like the custom handwritten digits.

Analysis and Insights

The pre-trained ResNet18 model showed low accuracy on both the MNIST and custom test sets, highlighting the inconsistency with its trained dataset and necessity of fine-tuning for task-specific adaptation. Fine-tuning only the last layer (Model 1) outputted a substantial increase in training and validation accuracy, yet the test accuracy remained moderate. In comparison, fine-tuning the last two layers (Model 2) significantly enhanced both validation and test accuracies, demonstrating adjustment with the deeper layers for improved feature extraction. The higher training accuracy in Model 2 suggests that the model successfully learned from the training data, while the consistent validation accuracy indicated stable performance.

Conclusion

Through this assignment, I successfully implemented transfer learning using ResNet18 for handwritten digit recognition. Fine-tuning additional layers proved more effective in improving model performance, which helped achieve a 75.00% accuracy on the second test set compared to 45.00% from the first one.