MLNS Deep Learning Assignment Part-2

Karan Nijhawan

1 Introduction

The goal of this project is to perform multi-class image classification using a pre-trained ResNet50 model. The dataset is divided into training, validation, and test sets, and various transformations are applied to enhance the model's generalization capability. A layer-wise learning rate policy is employed for fine-tuning the model, and the training process is evaluated on accuracy and loss metrics.

2 Dataset and Preprocessing

The dataset consists of three subsets (data0, data1, and data2), each containing images and their corresponding labels. These subsets are combined into a single dataset and split into training (80%), validation (10%), and test (10%) sets. The preprocessing steps include:

- Conversion to tensors.
- Repeating single-channel images to create 3-channel inputs for compatibility with ResNet50.
- Random rotation for data augmentation.
- Normalization using mean $[0.5, 0.5, 0.5]$ and standard deviation $[0.5, 0.5, 0.5]$.

The dataset is wrapped in a custom PyTorch Dataset class and loaded using DataLoader for efficient batch processing.

3 Approaches Tried

Several alternative approaches were attempted during this project:

- **Custom CNN:** A convolutional neural network was designed from scratch, but the performance was suboptimal compared to pre-trained models.
- **ResNet18:** A smaller variant of ResNet, ResNet18, was fine-tuned. However, it showed lower accuracy due to its reduced capacity compared to ResNet50.
- **LoRA CNN Tuning:** Low-Rank Adaptation (LoRA) was applied to tune the CNN layers, aiming to reduce the number of trainable parameters. While computationally efficient, this approach did not outperform the standard fine-tuning of ResNet50.
- **Segmentation and Individual Image Classification:** The images were segmented into individual parts, and each part was classified separately. This approach faced challenges in aggregating results effectively.
- **Autoencoder for Classification:** An autoencoder was used to classify images by reconstructing inputs and using the latent space for prediction. Additionally, the sum of reconstructed predictions was used to infer class labels. While innovative, this approach did not achieve satisfactory accuracy.

4 Final Model Architecture

We use a pre-trained ResNet50 model from torchvision's model zoo. The model's fully connected (fc) layer is replaced with a new layer having 37 output nodes, corresponding to the number of classes. The architecture is fine-tuned using layer-specific learning rates to optimize different parts of the network effectively.

5 Training Procedure

The training process is conducted using the following:

- **Loss Function:** CrossEntropyLoss, suitable for multi-class classification.
- **Optimizer:** Adam optimizer with weight decay for regularization.
- **Learning Rates:** Layer-specific learning rates, increasing from earlier to later layers.
- **Epochs:** The model is trained for 50 epochs.

During each epoch:

1. The model is set to training mode and the optimizer's gradients are zeroed.
2. Forward and backward passes are performed on training batches to update the model weights.
3. Validation data is evaluated without gradient updates to compute loss and accuracy.
4. If the validation accuracy improves, the model is saved as 'resnet-step.pth'.

6 Evaluation

The trained model is evaluated on the test dataset. The process involves a forward pass through the model to predict class probabilities, followed by computation of accuracy based on the predicted and true labels.

7 Results

The training process produces the following metrics:

- **Train Loss and Accuracy:** Monitored during each epoch.
- **Validation Loss and Accuracy:** Used to determine the best-performing model.
- **Test Accuracy:** Final metric to evaluate the generalization capability of the model.

The best model achieved a validation accuracy of **92.33%** and a test accuracy of **91.77%**.

8 Conclusion

This project demonstrates the effective use of transfer learning with ResNet50 for multi-class image classification. Data augmentation and a layer-specific learning rate policy significantly improved the model's performance. Future work could involve exploring other pre-trained models, additional data augmentation techniques, or hyperparameter tuning.

9 Figures

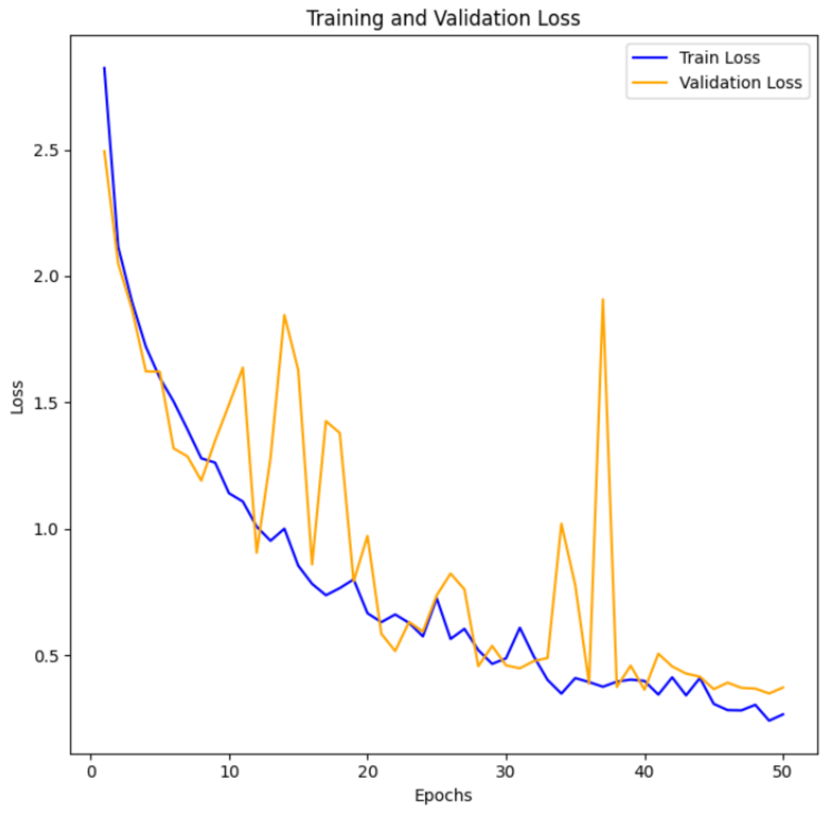

Figure 1: Sample training loss and validation loss.

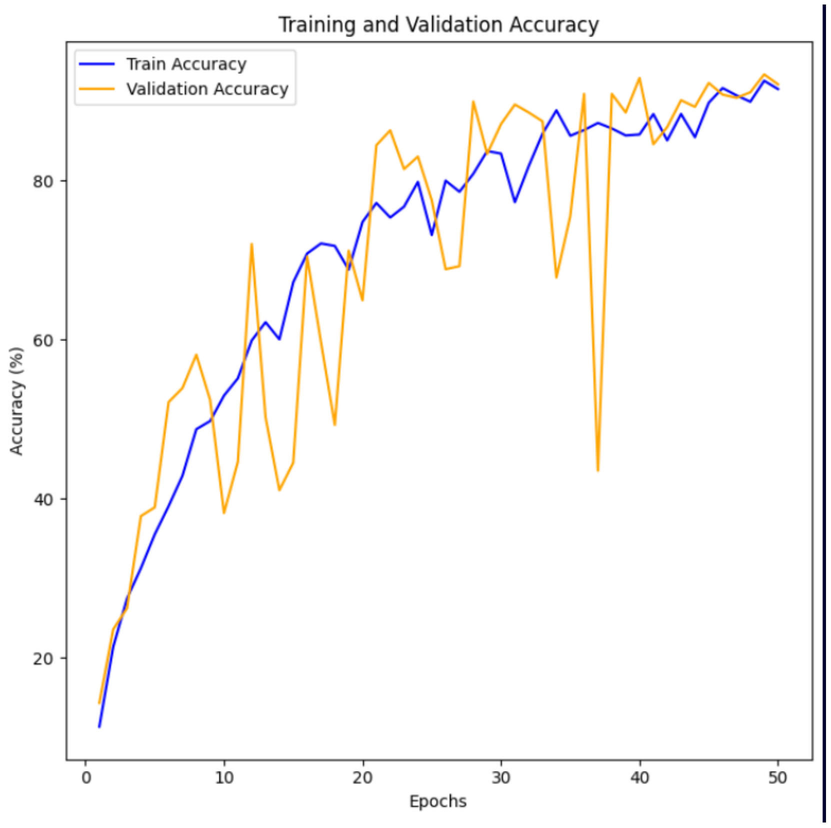

Figure 2: Sample training accuracy and validation accuracy curves.