Visual Recognition using Deep Learning - Homework 1

110550108

GitHub Repository: [Link]

1 Introduction

The goal of this homework is to develop an image classification model for a dataset containing 100 classes. The task requires training a deep learning model to recognize images from different categories. My approach leverages a **pre-trained ResNet-152** model and fine-tunes it for improved classification accuracy. I also explore the effects of different loss functions on model performance.

2 Method

2.1 Data Preprocessing

To prepare the dataset for training, I applied the following preprocessing steps:

- **Training Data Augmentation:**
  - RandomResizedCrop(224)
  - RandomHorizontalFlip
  - RandomRotation(15 degrees)
- **Validation and Test Data Transformation:**
  - Resize(256)
  - CenterCrop(224)
- **Normalization:** ImageNet mean and standard deviation values:
  - Mean: [0.485, 0.456, 0.406]
  - Std: [0.229, 0.224, 0.225]

2.2 Model Architecture

I employed **ResNet-152** as the backbone, initialized with **ImageNet pre-trained weights (IMAGENET1K_V2)**. The final fully connected layer was replaced with a new classification head:

- Dropout (0.5)
- Fully connected layer with 100 output classes

2.3 Training Strategy

- **Loss Functions Explored:**
  - CrossEntropyLoss (baseline)
  - Focal Loss to handle class imbalance
  - Label Smoothing Loss to improve generalization
- **Optimizer:** AdamW with an initial learning rate of 10^{-3}
- **Learning Rate Scheduler:** ReduceLROnPlateau (factor = 0.2, patience = 3)

- **Batch Size:** 128
- **Gradient Accumulation:** 2 steps
- **Number of Epochs:** 100

2.4 Training Procedure

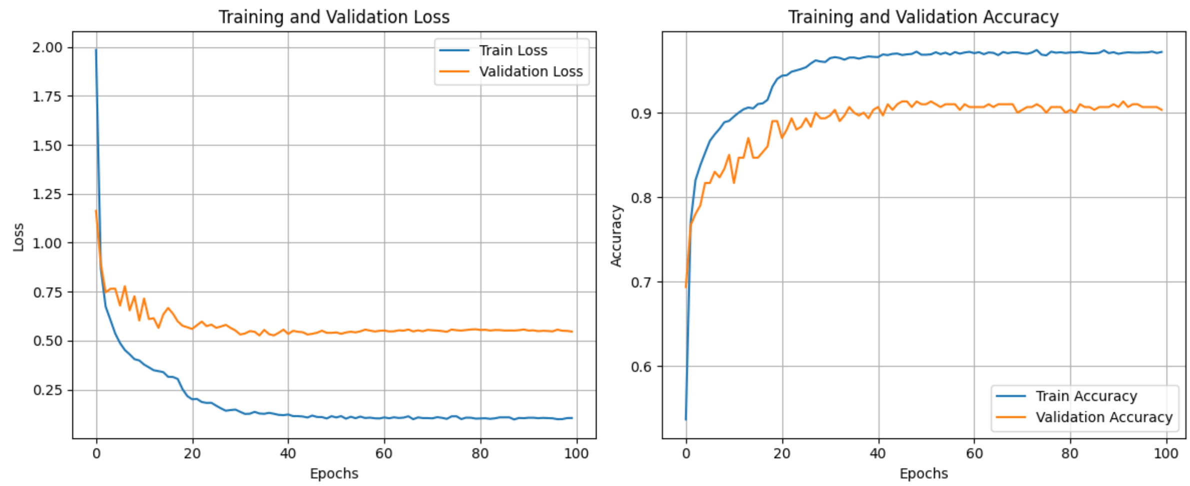
I employed **mixed precision training** using `torch.amp` to improve efficiency. Training followed these steps:

1. Forward pass with `autocast` for reduced memory usage.
2. Compute loss and perform gradient accumulation.
3. Update model parameters every 2 mini-batches.
4. Validate the model after each epoch and save the best checkpoint based on validation accuracy.

3 Results

3.1 Model Performance

The best validation accuracy achieved was **91.33%** using **CrossEntropyLoss**. Below are the training and validation loss curves and accuracies:



Public Leaderboard:

23	110550067	1	2025-03-20 10:39	249082	110550067	0.95
24	313553024	1	2025-03-15 02:21	246075	313553024	0.95
25	sudokudummy	1	2025-03-20 15:51	249287	313554001	0.94
26	111550089	1	2025-03-22 11:16	250316	111550089	0.94
27	110550108	1	2025-03-22 19:59	250469	110550108	0.94
28	313553023	1	2025-03-20 02:50	248881	313553023	0.94
29	113550901	1	2025-03-13 11:28	244813	113550901	0.94
30	strong-baseline	1	2025-03-02 01:28	239205	n/a	0.94
31	111550084	1	2025-03-20 14:23	249213	111550084	0.94

3.2 Key Findings

- The **learning rate scheduler** dynamically adjusted learning rates, improving the final performance.

4 Additional Experiments: Loss Function Comparisons

4.1 Hypothesis

Different loss functions influence how a model learns and generalizes. The hypothesis for each loss function:

- **CrossEntropyLoss:** Standard loss function, but may be sensitive to class imbalance.
- **Focal Loss:** Helps the model focus on *harder-to-classify samples*, reducing bias toward frequent classes.
- **Label Smoothing Loss:** Prevents overconfidence in predictions, improving *generalization*.

4.2 Expected Outcomes

- **Focal Loss** should improve the results for underrepresented classes, but it might require careful tuning.
- **Label Smoothing Loss** should prevent overfitting but may slow down learning in early epochs.

4.3 Experiment Results and Implications

Loss Function	Final Validation Accuracy	Observations
CrossEntropyLoss	91.33%	Baseline performance
Focal Loss	90.67%	More stable training, but slightly lower accuracy
Label Smoothing Loss	90.33%	Better generalization, but convergence was slower

- **Focal Loss** helped handle class imbalance but was sensitive to hyperparameters.
- **Label Smoothing Loss** resulted in better generalization, but the model required more epochs to converge.
- **CrossEntropyLoss** remained the best performing option, balancing both accuracy and training speed.

5 References

- **ResNet Architecture:** He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770-778. [Link]
- **PyTorch ResNet Implementation:** Official PyTorch Model Zoo [Link]