Visual Recognition HW1

111550034 黃皓君

Git link: Link

1. Introduction

My implementation is to improve model robustness and prediction accuracy by employing an ensemble method based on bagging. Use multiple instances of the ResNeXt101_32x8d architecture, each trained on a bootstrapped subset of the training data. The final prediction is obtained by averaging the softmax probabilities across the ensemble. This method mitigates overfitting and leverages the variability introduced by random sampling to improve generalization.

2. Method

Data Augmentation:

- Random resized crop to 224×224
- Random horizontal flip (p=0.5)
- Random rotation up to 30 degrees
- Color jitter (brightness, contrast, saturation, hue)
- Normalization: Each image is normalized to mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225], consistent with ImageNet statistics.

Model Architecture and Hyperparameters

- **Backbone:** ResNeXt101_32x8d, pre-trained on ImageNet (via PyTorch's ResNeXt101_32X8D_Weights.IMAGENET1K_V2).
- **Optimizer**: Stochastic Gradient Descent (SGD) with:
  - **Learning rate** = 0.01
  - **Momentum** = 0.9
  - **Weight decay** = 1e-4
- **Scheduler**: StepLR with step_size = 10 and gamma = 0.1
- **Batch size**: 32
- **Number of epochs**: Up to 50 (with early stopping)

Training and Ensemble Strategy

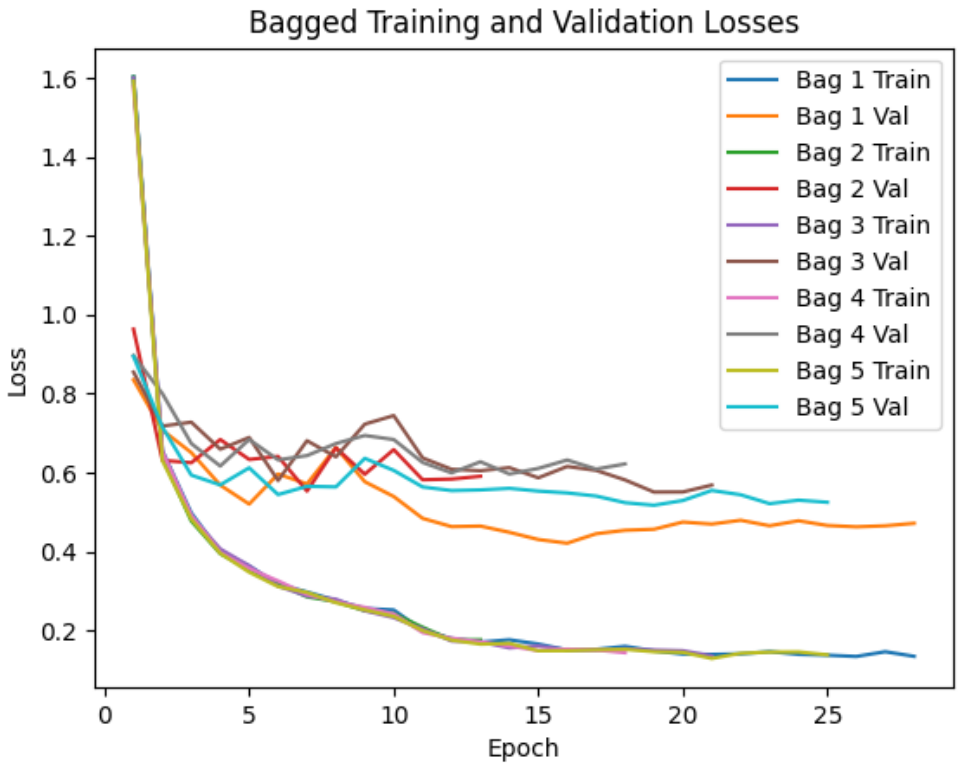
- **Bootstrap Sampling**: For each of the 5 models (bags), randomly sample (with replacement) from the training set to create a new training subset of the same size as the original dataset.
- **Training:** Each bag model is trained independently on its corresponding

bootstrapped dataset.

- **Early Stopping**: Monitor validation accuracy. If it does not improve for 6 consecutive epochs, we stop training that particular bag model.
- **Ensemble Inference**: Instead of relying on models from a single training run, I tried **multiple training sessions** and selected the five models with the highest validation accuracy. The final ensemble prediction is computed by averaging the softmax probabilities from these top-performing models, thereby leveraging their collective strength and reducing the variance of individual predictions..

3. Results

One of the training and validation loss curves show below.



After selecting the 5 models with best accuracy, their individual validation accuracies ranged from **90% to 92%**. The ensemble accuracy on the final test set reached **96%**.

4. Additional Experiments

Modified Architecture:

Hypothesis: Adding dropout before the final classification layer will help

reduce overfitting.

How This May Work:

Dropout forces the network to learn more robust features by mitigating co-adaptation, which can lead to better generalization on unseen data.

Experiment: Tested various dropout probabilities (e.g., 0.0 vs. 0.5) and observed that moderate dropout (around 0.5) yielded better generalization on the validation set.

Bagging:

Hypothesis:

Averaging predictions from multiple models reduces variance and smooths out idiosyncratic errors of individual models.

How This May Work:

By training each model on a different bootstrapped subset, the ensemble benefits from diverse perspectives and errors are averaged out, leading to improved overall performance.

Experiment:

- Individual bag accuracies ranged from 90–92%.
- The ensemble accuracy reached 96%, confirming that the combined model outperforms any single bag model.

Implications: This demonstrates that bagging with a strong architecture (ResNeXt101_32x8d) is an effective strategy for boosting classification performance without major changes to the underlying network.

Test-Time Augmentation

Hypothesis:

For each test image, apply several augmentations (e.g., standard resize & crop, horizontal flip, fixed rotation) and average the resulting softmax probabilities for each model.

How This May Work:

TA can average out the noise from any single augmented view, leading to more stable and accurate predictions.

Experiment:

The ensemble with TTA achieved a test accuracy from 94% -> 95%.

5. References

torch.optim — PyTorch 2.6 documentation

resnext101_32x8d — Torchvision main documentation

torchvision — Torchvision 0.21 documentation