Report: Analysis of CUB-200 Classification Based on ResNet101

1. Project Background

This project aims to perform image classification on the CUB-200 dataset using deep convolutional neural networks. The CUB-200 dataset contains images of 200 bird species, with a large number of images and subtle differences between classes, thus imposing high requirements on the model's feature extraction capability. The code is implemented based on the PyTorch framework, adopting the ResNet101 pre-trained model for transfer learning and incorporating the Dropout mechanism to prevent overfitting.

---

2. Data Processing and Augmentation

- Data Augmentation

  To improve the model's generalization ability, multiple augmentation methods are applied to the training data:

  - Resize to (256, 256) and RandomResizedCrop to (224, 224)
  - RandomHorizontalFlip, RandomRotation(15°)
  - ColorJitter (brightness, contrast, saturation, hue)
- Normalization

  Normalization is performed using the mean and standard deviation of ImageNet.
- Data Splitting
  - The dataset is split into training set and validation set at an 8:2 ratio.
  - DataLoader is used for batch loading, with shuffle enabled for the training set and disabled for the validation set.

---

3. Model Design

- Backbone Network

  The ResNet101 model is used, with convolutional layers loaded with ImageNet pre-trained weights (excluding the fc layer).
- Classification Head Modification
  - The original fully connected layer is replaced with `Dropout + Linear`, where the number of output nodes equals the number of classes.
  - The Dropout probability is set to 0.5 to prevent overfitting.
- Transfer Learning
  - When loading pre-trained weights, the final fc layer is excluded to retrain the classifier.

- The model is moved to GPU (if available) to accelerate training.

---

4. Loss Function and Optimizer

- **Loss Function**: Cross-entropy loss `nn.CrossEntropyLoss()`, suitable for multi-classification tasks.
- Optimizer

  : Stochastic Gradient Descent (SGD)
  - Learning rate: 1e-4
  - Momentum: 0.9
  - Weight decay: 1e-4

---

5. Training Strategy

- Training Loop
  - After each training epoch, evaluate loss and accuracy on the validation set.
  - Calculate training/validation loss and accuracy for each epoch for curve plotting.
- Early Stopping Mechanism
  - The early stopping patience is set to 7 epochs. Training stops early if the validation loss does not decrease for 7 consecutive epochs.
- Best Model Saving
  - After the 30th epoch, save the model whenever the validation loss achieves a new minimum.
- Real-time Plotting
  - Call `save_metrics_curve` after each epoch to save training/validation loss and accuracy curves.

---

6. Implementation Features and Highlights

1. **Combination of Transfer Learning and Fine-tuning**: ResNet101 is used to enhance feature extraction capability, while the fc layer is retrained to adapt to the new task.
2. **Comprehensive Data Augmentation**: Multiple augmentation methods are combined to improve the model's robustness to changes in lighting, rotation, and color.
3. **Training Monitoring and Visualization**: Automatically save training and validation curves to facilitate real-time observation of training dynamics.
4. **Early Stopping Mechanism**: Avoid overfitting and improve training efficiency.
5. **GPU Acceleration**: Automatically detect the device and use CUDA to speed up model training.

---

7. Usage and Outputs

- Training Outputs

:

  - Print training/validation loss and accuracy for each epoch in the console.
  - Save the best model as `best_model.pth`.
  - Save training curves as `train_val_metrics.png`.
- Applicable Scenarios

  :

  - Image classification tasks
  - Transfer learning experiments
  - Fine-tuning of large-scale CNN models

---

8. Analysis of Loss and Accuracy Curves & Comprehensive Conclusions

8.1 Loss Curve Analysis (Left Figure)

- **Trend**: Both training loss and validation loss decrease with the number of training epochs and tend to stabilize around the 90th epoch.
- Convergence

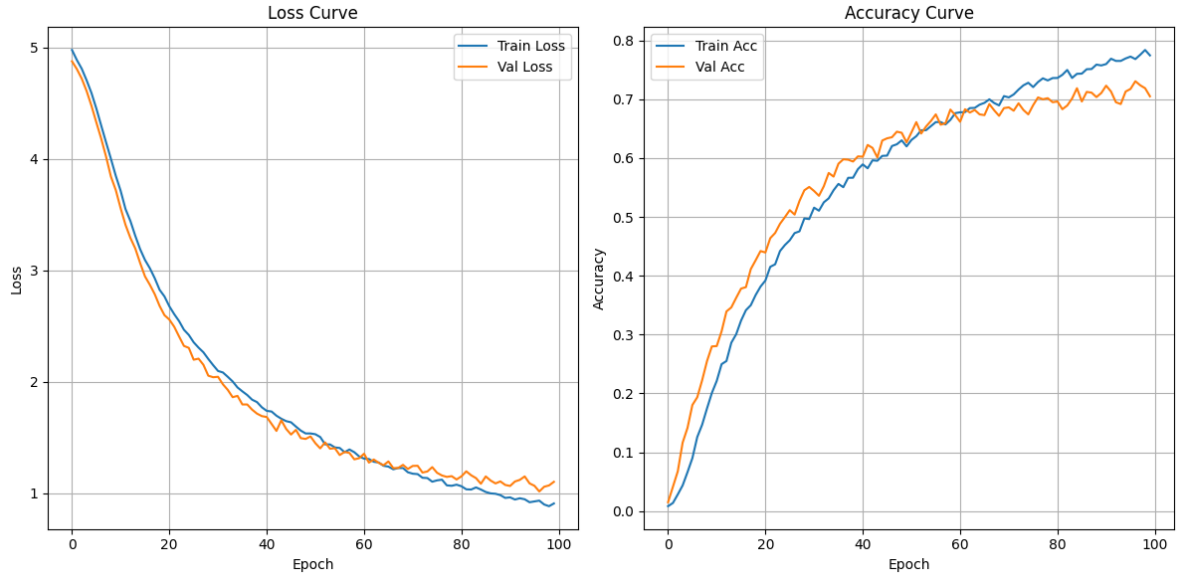
  :

  - The loss drops rapidly in the early stage (0–40 epochs), indicating that the model quickly learns effective features.
  - The decline slows down in the mid-to-late stage, showing the model enters a fine-tuning phase with gradually converging loss.
- Difference Between Training and Validation

  :

  - The validation loss curve is very close to the training loss curve, indicating no significant overfitting.
  - Occasional cases where validation loss is higher than training loss are normal fluctuations.

8.2 Accuracy Curve Analysis (Right Figure)



- **Trend**: Both training accuracy and validation accuracy steadily improve throughout the training process.
- Convergence
  - The validation accuracy stabilizes at around 60–70%, while the final training accuracy is slightly higher (approximately 77%), which is consistent with common training patterns.
- Overfitting Situation
  - Although the training accuracy is slightly higher than the validation accuracy, the gap between the two curves is small (≈7–8%), demonstrating the model has good generalization ability.
- Training Dynamics
  - The accuracy increases significantly in the first 30–40 epochs, corresponding to the stage of rapid loss reduction.
  - The accuracy growth slows down in the later stage as the model enters convergence.

8.3 Comprehensive Conclusions

The model training process is stable, with smooth loss and accuracy curves. It exhibits rapid learning in the early stage and effective fine-tuning convergence in the later stage. The validation set performance is close to that of the training set, which confirms the effectiveness of the transfer learning strategy and data augmentation methods. The final model achieves approximately 77% training accuracy and 70% validation accuracy, making it suitable for subsequent fine-tuning or deployment.

To further improve the accuracy, the following optimizations can be attempted:

- Increase the intensity of data augmentation or expand the sample size.
- Adjust the learning rate or adopt learning rate decay strategies.
- Introduce more advanced regularization techniques (e.g., Label Smoothing or CutMix).