EuroSAT ResNet50 Training Report

Method and data preprocessing

This project applies transfer learning using a ResNet50 model to classify satellite images from the EuroSAT dataset. The workflow covers dataset preparation, model construction, training with scheduled learning rates, metric logging, visualization, and evaluation. In addition to describing the training process, this report explains which output files contain which information so that results can be inspected directly.

The dataset is loaded from the Kaggle environment. The files train.csv, validation.csv, and test.csv define the data splits and provide file paths and class labels. A custom dataset class processes image files and applies appropriate transforms. For the training split, random horizontal flips, vertical flips, and random rotations are used to increase robustness. Validation and test splits use only normalization based on ImageNet statistics.

Model Training and Fine-tuning

The project uses a PyTorch Lightning DataModule to organize dataloaders and separate data handling from model logic. A pretrained ResNet50 model is loaded from a file named resnet50-11ad3fa6.pth, which is provided in the Kaggle input directory. The final fully connected layer is replaced with a new layer producing ten output classes.

Training is executed for a maximum of fifteen epochs using AdamW optimization and OneCycleLR scheduling. The EarlyStopping callback monitors validation loss to avoid unnecessary epochs. The RichProgressBar and Timer callbacks display run-time information. PyTorch Lightning logs all metrics through CSVLogger, which stores its outputs in a directory named logs.

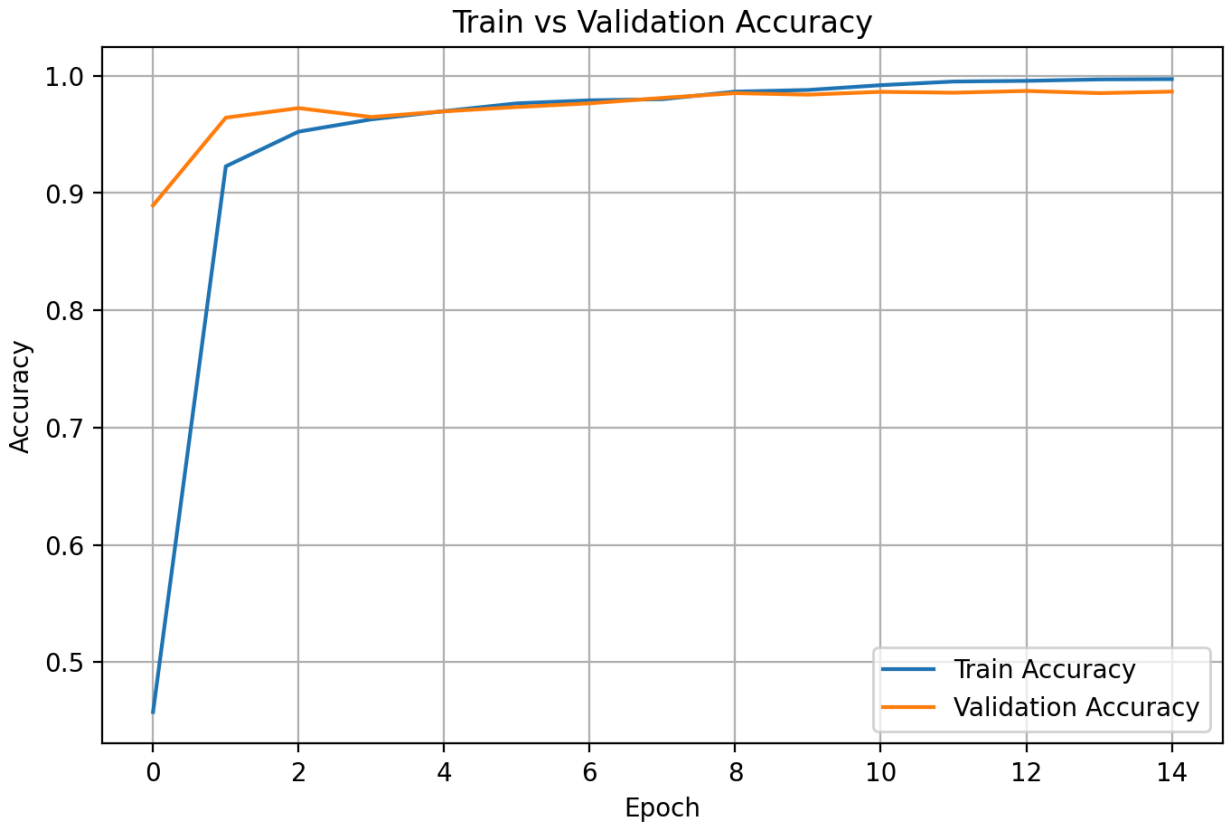
Result

All recorded metrics during training and validation are written into logs/version_x/metrics.csv. This file contains values such as train_loss, train_acc, val_loss, and val_acc recorded at the epoch level. After completion, metrics.csv can be used to examine the learning dynamics or create plots.

A second generated file named lr_history.csv stores the learning rate at every optimization step. This file allows the OneCycleLR curve to be inspected. The line lr inside this CSV shows how the learning rate increased and then decreased. This helps verify that the scheduler behaved correctly.

Two additional JSON files summarize training results. The file epoch_metrics.json contains epoch grouped metrics. Each epoch entry includes final values for train_loss, train_acc, val_loss, and val_acc. This file is useful for checking whether the early epochs behaved as expected and whether the model converged steadily.

The file final_metrics.json contains only the metrics from the final epoch. It includes values such as final validation accuracy and validation loss. This file provides a quick reference for the final performance without scanning all epochs.



[fig.1 learning_curve_acc]

A plot named *learning_curve_acc* is saved to the same logs directory. This figure displays training accuracy and validation accuracy across epochs. In the produced curve, both lines rise quickly during the early epochs and stabilize near 0.98 to 1.00. The closeness of both curves indicates minimal overfitting.

Test metric	DataLoader 0
test_acc	0.9877777695655823
test_loss	0.048327233642339706

[Table 1. Test metric]

After completing training, the model is evaluated using the test split. The resulting test accuracy is approximately 0.9878 and the test loss is about 0.0483 *[Table 1]*. These values show that the model generalizes well and can classify unseen satellite images with high precision. These metrics are displayed in the console and also stored in metrics.csv and final_metrics.json.

The entire trained model is saved as resnet50_eurosat_best.pth in the working directory. This file contains the full state dictionary for the fine tuned ResNet50. It can be loaded later for inference or cloud deployment. Because the model was trained using PyTorch and saved with standard state_dict format, the checkpoint is portable and can be loaded on different operating systems or environments, including CPU or GPU based inference setups.

Summary

In summary, this workflow fine tunes a ResNet50 model using a structured pipeline based on PyTorch Lightning. The combination of data augmentation, pretrained initialization, and OneCycleLR allows the model to reach nearly perfect generalization on EuroSAT. The logged files, especially metrics.csv, lr_history.csv, epoch_metrics.json, learning_curve_acc.png, and resnet50_eurosat_best.pth, enable complete transparency of the training process and make the experiment fully reproducible.