EuroSAT Land Cover Classification: Approach Write-up

Problem Statement

Satellite imagery plays a crucial role in environmental monitoring, urban planning, and disaster response. The EuroSAT dataset contains multispectral Sentinel-2 satellite image tiles of size 64×64 pixels across 10 land-use classes. The task is to develop a complete machine learning pipeline to classify each tile into its correct land-cover type, handling raw, noisy, and inconsistent data.

1. Environment Setup and Library Imports

All necessary libraries, including TensorFlow, OpenCV, PIL, scikit-learn, and matplotlib, are imported. This ensures the environment is ready for data processing, model building, and evaluation.

2. Configuration and Data Path Setup

Key parameters such as `DATA_ROOT`, `IMG_SIZE`, `BATCH_SIZE`, `EPOCHS`, and `RANDOM_SEED` are defined. This allows for easy configuration and reproducibility of the experiments.

3. Image Reading and Corruption Detection Utilities

Robust functions are implemented to read images in various formats (.jpg, .png, .tif) and detect corrupted files. Corruption is identified based on black pixel thresholds and variance analysis, ensuring only valid images are processed.

4. Data Cleaning and Preprocessing Pipeline

The dataset is processed by reading images, detecting and moving corrupted files to a separate folder. A Gaussian denoising filter is applied to clean the images, and the clean data is organized for subsequent training steps.

5. Feature Extraction and Visualization

Functions are developed to extract color histograms and edge histograms from images. Visualization functions are also included to display sample images alongside their feature distributions, aiding in understanding the data characteristics across different classes.

6. Label Encoding and Data Splitting

Class names are converted into numerical indices. The dataset is then split into stratified training and validation sets. Data generators are set up with augmentation techniques to enhance the training process and improve model generalization.

7. Baseline CNN Model Architecture

A Convolutional Neural Network (CNN) is built from scratch. It consists of three convolutional blocks, each incorporating `Conv2D`, `BatchNormalization`, `ReLU` activation, `MaxPooling`, and `Dropout` layers. These are followed by dense layers for classification.

8. Training Helpers and Visualization Functions

Utility functions are provided for plotting the training history (loss and accuracy curves) and visualizing intermediate layer activations of the model. These tools are crucial for monitoring training progress and understanding model behavior.

9. Load and Preprocess Dataset

This section executes the defined data loading and preprocessing pipeline. Images are loaded, cleaned, and normalized to a [0,1] range. The class distribution is also checked to ensure data balance.

10. Data Splitting and Preparation for Training

The preprocessed data is split into training and validation sets using `train_test_split` with stratification. Data augmentation is configured using `ImageDataGenerator` to create `train_generator` and `val_generator`, preparing the data for model training.

11. Baseline Model Training

The baseline CNN model is compiled with an Adam optimizer, categorical cross-entropy loss, and accuracy metrics. Callbacks for model checkpointing, early stopping, and learning rate reduction are configured to optimize the training process and save the best performing model.