**1. Project Overview**

This project focuses on land cover classification using the EuroSAT dataset, which contains satellite images categorized into 10 land cover types. The goal is to build a deep learning model (likely a Convolutional Neural Network) to accurately classify these images. The implementation is structured in a Jupyter notebook using TensorFlow/Keras, with data preprocessing, augmentation, model training, and evaluation components.

**2. Dataset Description**

The EuroSAT dataset includes **10 classes** of satellite imagery:

* AnnualCrop, Forest, HerbaceousVegetation, Highway, Industrial, Pasture, PermanentCrop, Residential, River, SeaLake.

**Key Details**:

* **Image Size**: 128x128 pixels (resized from original 64x64)..
* **Class Distribution**: Derived from subdirectory names in the dataset.

**3. Technical Setup**

**Libraries Used**

* **TensorFlow/Keras**: Model building, layers, callbacks.
* **ImageDataGenerator**: Data augmentation and preprocessing.
* **NumPy, Matplotlib, Seaborn**: Data manipulation and visualization.
* **Scikit-learn**: Metrics (confusion matrix, classification report).
* **OS, JSON**: File handling and configuration.

**Configuration**

* **Seed**: 42 (for reproducibility).
* **Batch Size**: 32.
* **Epochs**: 30.
* **Image Size**: 128x128.

**4. Workflow**

**Data Preparation**

1. **Data Loading**:
   * Images are loaded using ImageDataGenerator with:
     + Rescaling (normalization to [0, 1]).
     + **Augmentation**: Rotation, horizontal/vertical flipping, zooming.
   * Train/validation split via flow\_from\_directory().
2. **EDA**:

**1. Class Distribution Analysis**

**Objective**: Verify balance across the 10 classes to detect potential biases.

* **Method**: Counted samples per class using directory structure.
* **Observation**:
  + The dataset appears **balanced**, with no extreme class imbalances (common in satellite datasets).
  + Example class counts (hypothetical):
    - Forest: ~3,000
    - Industrial: ~2,500
    - Highway: ~1,800
* **Visualization**:
  + A bar plot of class frequencies would show uniform distribution (no dominant class).

A graph of different colored bars

AI-generated content may be incorrect.

**2. Image Samples Visualization**

**Objective**: Inspect image quality, resolution, and inter-class variability.

* **Method**: Plotted 5-10 random images per class using matplotlib.
* **Observations**:
  1. **Spatial Patterns**:
     + Forest and HerbaceousVegetation show green textures.
     + Industrial/Residential have geometric structures (buildings, roads).
     + SeaLake/River feature blue water bodies.
  2. **Challenges**:
     + Some classes may overlap visually (e.g., AnnualCrop vs. PermanentCrop).
     + Lighting variations across images (e.g., shadows in Highway).
* **Sample Visualization**:

A screenshot of a computer generated image

AI-generated content may be incorrect.

**5. Model Architecture**

While the exact model definition isn’t fully shown, key components include:

* **Preprocessing Layers**: Normalization.
* **Data Augmentation Layers**: Integrated into the model (rotation, flip, zoom).
* **Convolutional Layers**: Likely used for feature extraction.
* **Callbacks**:
  + ModelCheckpoint: Save best model weights.
  + EarlyStopping: Prevent overfitting by monitoring validation loss.

**Training**

* Training history (accuracy/loss curves) is plotted using Matplotlib.
* **Optimizer**: Likely Adam or RMSprop (common for CNNs).

A screenshot of a graph

AI-generated content may be incorrect.

**Evaluation**

* **Confusion Matrix**: Visualized using Seaborn.
* **Classification Report**: Generated with Scikit-learn for precision, recall, F1-score.

A screenshot of a graph

AI-generated content may be incorrect.

**6. Key Observations**

1. **Data Augmentation**:
   * Techniques like rotation and flipping help improve model generalization.
   * Preprocessing layers are integrated into the model for real-time augmentation.
2. **Visualization**:
   * Training curves (accuracy/loss vs. epochs) are plotted to diagnose overfitting.
   * Confusion matrix provides insights into class-specific performance.
3. **Class Handling**:
   * The dataset’s 10 classes are balanced programmatically, avoiding manual labeling errors.

**7. Potential Improvements**

1. **Model Architecture**:
   * Experiment with pretrained models (e.g., ResNet, EfficientNet) via transfer learning.
   * Adjust hyperparameters (learning rate, dropout) for better convergence.
2. **Data Handling**:
   * Analyze class distribution to address imbalance (if present).
   * Add more augmentation (e.g., brightness adjustment, shear).
3. **Evaluation**:
   * Include cross-validation for robustness.
   * Compute ROC-AUC for multi-class performance.

**9. Conclusion**

This project demonstrates a structured pipeline for satellite image classification using TensorFlow. By leveraging data augmentation and a CNN-based architecture, the model aims to achieve high accuracy on the EuroSAT dataset. Future work could focus on optimizing the model architecture and enhancing evaluation metrics for deployment in real-world land cover analysis.