**PROJECT REPORT**

**1) Project Aim**

The objective of this project is to develop a binary image classifier to distinguish between *soil* and *non-soil* images using a deep learning-based approach. Leveraging EfficientNet-B0, hybrid datasets, data augmentation, and a boosted ensemble, the goal is to build a highly accurate, generalizable model to aid automated soil detection.

**2) Dataset**

The training data is sourced from two separate soil image datasets:

* Dataset 1 (Binary):  
  Path → /kaggle/input/soil-classification-2/soil\_competition-2025/  
  Contains labeled images for binary classification.
* Dataset 2 (Multi-class):  
  Path → /kaggle/input/soil-classification/soil\_classification-2025/  
  Originally multi-class, but re-labeled to "1" (soil) to enrich the positive class.
* Test Set:  
  Provided separately under the test folder, along with test\_ids.csv.

Each sample is referenced by an image\_id, and the associated labels are stored in CSV files.

**3) Cleaning and Preprocessing**

* Smart Cropping:
  + A custom function crops the image by removing low-contrast background pixels using thresholding and contour detection.
* Transformations:
  + Training Data Augmentations:
    - Resize to 256×256
    - Random affine transformations
    - Horizontal/vertical flips
    - Color jitter
    - Random erasing
    - Normalization (ImageNet stats)
  + Validation/Test Data:
    - Only resize and normalization applied.
* Fallback Handling:
  + If image reading or transformation fails, a zeroed 256×256 fallback image is used to avoid data loss.
* Class Weights:
  + Computed using sklearn.utils.class\_weight to address class imbalance during training.

**4) Model Used**

* Backbone:  
  EfficientNet-B0 (from torchvision.models) pre-trained on ImageNet.
* Modifications:
  + The final classification layer is replaced with a Linear layer outputting 2 classes (soil vs. non-soil).
  + Model is deployed to GPU (if available).

**5) How We Trained the Model**

* Boosted Ensemble:
  + 3 rounds of training with different random seeds (42, 43, 44) using stratified splits to ensure balanced validation sets.
  + Each round:
    - Uses train\_test\_split() on the combined dataset.
    - Trains for 15 epochs using:
      * CrossEntropyLoss
      * Adam Optimizer (learning rate: 1e-4)
      * Mixed Precision Training via torch.amp.GradScaler for GPU efficiency.
    - The best model per round (based on F1 score on validation set) is saved.

**6) Making Predictions**

* Inference Pipeline:
  + Loads all 3 best models.
  + Runs test images through the same preprocessing pipeline as validation.
  + Softmax probabilities from all models are averaged (simple ensembling).
  + Final prediction = argmax of the averaged probabilities.

**7) Submission File**

* The predicted labels are merged with test image IDs and saved as:
* /kaggle/working/submission.csv
* Format:

| image\_id | label |
| --- | --- |
| example1.jpg | 0 |
| example2.jpg | 1 |

This file is ready for submission in the competition environment.