Leipzig University

Deep learning for Master Students

Prof. Alexander Binder

EuroSAT RGB & Multispectral Classification Assignment

Report on mandatory task

January 1, 2026

Group Members

Asel Baibekova 3719704
Jiacheng Lang 3701369

Work Description

The assignment covers three main tasks:

- dataset splitting
- RGB-based scene classification
- multispectral classification with late fusion on the EuroSAT dataset

All experiments are controlled via a fixed random seed corresponding to the student's matriculation number (in use: 3719704), ensuring reproducibility across runs.

Project root and dataset roots

To satisfy the path-handling requirements of the assignment, the implementation distinguishes clearly between the *project root* and the *dataset roots*.
The **project root** is defined as the top-level directory that contains the source code, the datasets and all output subdirectories. In the submitted implementation, the project root is the directory that contains among others the following entries:

```
project_root/
 project/
    splits/
    checkpoints/
    submission/
    models/
    logs/
    predictions/
    python files
 EuroSAT_RGB/
 EuroSAT_MS/
```

A small configuration module `config.py`, located in `homework final/` computes the project root dynamically based on the location of the source file and derives from it the dataset roots and the output directory:

```
PROJECT_ROOT = personal project root sets by user
DATASETS_ROOT = personal datasets root sets by user
SPLITS_ROOT = PROJECT_ROOT / "splits"
RGB_DATASET_ROOT = DATASETS_ROOT / "EuroSAT_RGB"
MS_DATASET_ROOT = DATASETS_ROOT / "EuroSAT_MS"
```

By default, the RGB and multispectral datasets are expected in `project root/EuroSAT_RGB` and `project root/EuroSAT_MS` under the project root, and `SPLITS_ROOT` defines the project-level output directory outside `project`. This design follows the assignment requirement that all files written by the code must reside in subdirectories under a given `project root` and that no hard-coded absolute paths should appear in the implementation.

1 Execution Instructions

Please run the necessary python files in following order:

- Task 1:
  - Data Splits: `split_data.py`
- Task 2:
  - RGB-Training: `train.py`
  - RGB-Test: `test.py`
  - RGB-Reproduction(save logits): `reproduce.py --generate`
  - RGB-Reproduction(default): `reproduce.py`
- Task 3:
  - MS-Training: `train_ms.py`
  - MS-Test and Reproduction: `ms_test_reproduce.py`
  - `SAVE_LOGITS`: True -> save logits, False -> compare logits

2 Task 1

2.1 Dataset Splitting

For Task 1, a dedicated script performs a stratified split of the original EuroSAT RGB dataset into training, validation, and test subsets.

- The original EuroSAT RGB dataset contains **27,000** images across **10 land use/land cover classes**.
- For each class independently, the available samples are split into:
  - **16,200** training images
  - **5,400** validation images
  - **5,400** test images

The split is **stratified by class**, so that the class distribution is preserved in all three subsets. The script:

- scans the dataset once,
- writes three text files — `train.txt`, `val.txt`, `test.txt` — each containing:
  - a **relative path** (with respect to the dataset root), and
  - the corresponding class label,
- verifies that:
  - the three subsets are **pairwise disjoint**, and

– their union exactly covers the full set of 27,000 images.

These split files are reused in Tasks 2 and 3 for RGB and multispectral training and evaluation, such that the train/validation/test separation is fixed and consistent throughout the project.

3 Task 2

3.1 ResNet-18 modification

We utilized a ResNet-18 backbone initialized with ImageNet weights. However, standard ResNet architectures are designed for 224×224 inputs, which causes aggressive downsampling on smaller satellite images (e.g., 64×64), leading to the loss of fine-grained spatial features. To adapt the network for satellite imagery, the initial 7x7 convolution was replaced with a 3x3 convolution, and the first max pooling layer was removed to preserve spatial resolution. The final fully connected layer was modified to map features to the 10 EuroSAT classes, with all layers of the network being fine-tuned. Additionally, a fixed random seed was used to initialize the environment, ensuring the reproducibility of training and evaluation.

3.2 Data loading and augmentations

A custom `EuroSATRGBDataset` reads the split files:
`train.txt`, `val.txt`, and `test.txt`.

A custom dataset class reads the split files and employs both "mild" and "strong" augmentation schemes for training. Mild augmentation includes basic geometric flips and light color jitter, while strong augmentation introduces aggressive rotation, affine transformations, and RandomErasing regularization. The model is trained for 20 epochs under each configuration, with loss and accuracy logged via Tensorboard.

3.3 Model selection and performance

- For each augmentation configuration, the checkpoint with the **highest validation accuracy** is stored as:
  - `mild_run/best_model.pth`
  - `strong_run/best_model.pth`
- After training with both augmentation schemes, we compare the performance of two models: mild as 98.43% versus strong as 97.91%. The best overall model (highest validation accuracy across both) is `mild_run/best_model.pth` (best model).
- The best validation accuracy for the RGB model reaches `98.43%` (mild augmentation).
- The corresponding test accuracy of the final RGB model is `98.17%`.

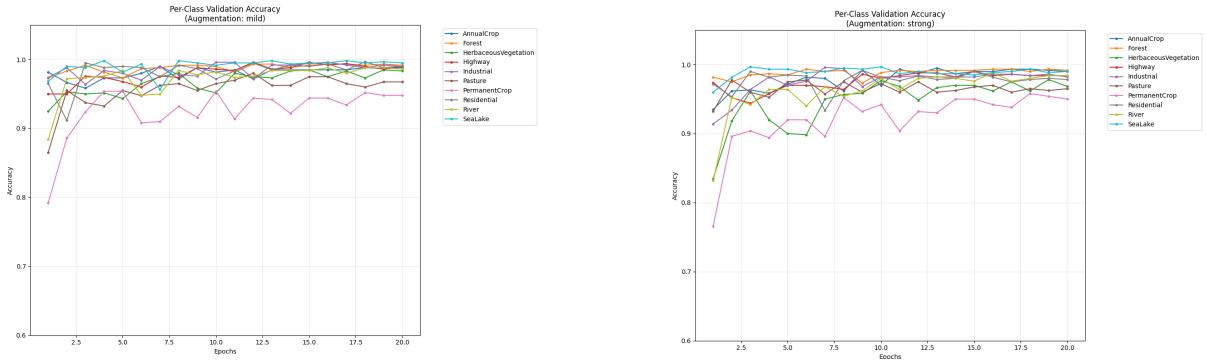
- The True Positive Rate per class on the test set is high for almost all classes (>95%), the highest is 99.50% in class Forest and the lowest is 95.60% in class PermanentCrop.

3.4 Test logits and reproducibility

The test.py script evaluates the best model, calculates metrics, and saves the raw logits as a binary file. You can also save logits by generate mode of reproduce.py. The reproduce.py script verifies consistency by re-running inference and comparing the new logits against the saved ones. Experiments showed a maximum absolute difference of 0.0 between runs, confirming the full numerical reproducibility of the pipeline.

3.5 Task 2 Results

3.5.1 Validation Performance Graphs



(a) mild augmentation (saved model) (b) strong augmentation

Figure 1: Validation Performance Graphs Per Class

3.5.2 Test Accuracy

Class Name	TPR (Recall/Acc)
AnnualCrop	98.83%
Forest	99.50%
HerbaceousVegetation	97.00%
Highway	97.80%
Industrial	99.00%
Pasture	96.00%
PermanentCrop	95.60%
Residential	99.33%
River	98.20%
SeaLake	99.33%
Overall Accuracy	98.17%

Figure 2: final performance number on test data using best model

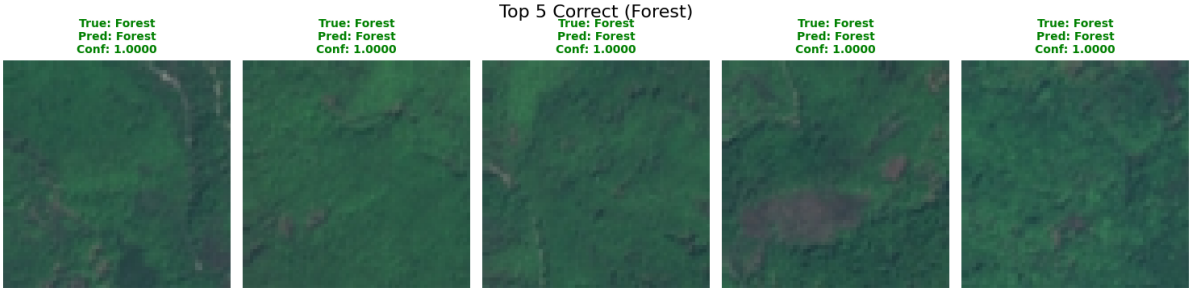
3.5.3 top-5 and bottom-5

Using visualization.py top-5 and bottom-5 test examples are extracted for several classes (e.g. *Highway*, *Forest*, *River*).

(a) Top 5 Correct Predictions

(b) Bottom 5 Failures

Figure 3: Analysis of **Highway** Clas



(a) Top 5 Correct Predictions

(b) Bottom 5 Failures

Figure 4: Analysis of **Forest** Class (only 3 pictures misclassified)

(a) Top 5 Correct Predictions

(b) Bottom 5 Failures

Figure 5: Analysis of **River** Class

4 Task 3

Multispectral Classification with Late Fusion.
In use: the **EuroSAT_MS** dataset with 13 spectral bands (Sentinel-2).

4.1 Data loading and preprocessing

- A custom `EuroSATMSDataset` reads the same split files `train.txt`, `val.txt`, and `test.txt` used in Task 2.
- Image paths are converted from `.jpg` to `.tif`; corresponding files are loaded from the EuroSAT_MS root directory.
- Multispectral images are loaded using `skimage.io.imread`:
  - raw pixel values are converted to `float32`,
  - values are normalized to the range **[0, 1]** (division by 65535 if necessary),
  - the data layout is rearranged from H×W×C to C×H×W.
- From the 13 available bands, **6 channels** are selected (e.g. bands B04, B03, B02, B05, B06, B12). These 6 channels are splitted into **two groups of three channels** each.
- Custom augmentation transforms (resize, random flips, random rotations by 90°) are applied directly to the multi-channel tensors without relying on `ToTensor`.

4.2 Late fusion architecture

To reuse pretrained RGB weights while processing 6-channel input, a **late fusion** architecture is designed following way:

- A `ResNet18Features` module wraps ResNet18 and returns the feature vector after the adaptive average pooling layer (i.e., before the final classification head).
- The 6 selected bands are separated into two 3-channel groups, each of which is treated as a pseudo-RGB image.
- Both groups are passed through the **same** `ResNet18Features` instance (shared weights), resulting in two feature vectors.
- These two vectors are then **concatenated** and fed into a single linear layer (`nn.Linear`) that produces the 10-class logits.

This design corresponds to a **single neural network with one final linear classifier**, as required by the assignment (important: not as an ensemble of two independent models).

4.3 Training and performance

- The training protocol mirrors the RGB case:
  - two augmentation schemes ("light" and "strong"),
  - 10 epochs per configuration,
  - logging of training/validation loss and accuracy into `ms_metrics.csv`.
- For each augmentation setting, the best validation checkpoint is stored as:
  - `ms_best_light.pt`,
  - `ms_best_strong.pt`.
- The overall best model across both augmentations is saved as `ms_final.pt`.

In the reported experiments:

- The best validation accuracy for the multispectral model reaches **91.0%** (strong augmentation).
- The final multispectral model achieves **90.8%** test accuracy.
- Per-class TPRs are mostly above 0.8 with lower values for certain vegetation-related classes such as *HerbaceousVegetation* and *Pasture*.

4.4 Logits, visualization and reproducibility

- Likewise test logits are saved to `splits/predictions/ms_test_logits.pt` and the corresponding filenames in turn to `splits/predictions/ms_test_filenames.txt`.
- A dedicated evaluation script for the MS model uses the same logic:
  - with `SAVE_LOGITS = True`, test logits and filenames are computed and stored;
  - with `SAVE_LOGITS = False`, newly computed logits are compared to the stored ones using `torch.allclose`.

As we mentioned before, the maximum absolute difference between *saved* and *recomputed* logits is `0.0`, confirming that the multispectral test evaluation is also fully reproducible.

An additional inspection script visualizes RGB and MS test patches side by side for the same geographic locations and allows interactive exploration of **top-5 / bottom-5** examples per class.
Qualitatively "easy" classes such as *Industrial*, *River*, *SeaLake* and *Residential* appear very similar in RGB and in a pseudo-RGB projection of multispectral bands (B04,B03,B02). Therefore both models assign high confidence to these scenes.
Ambiguous vegetation patches (such as mixtures of fields, tracks or small forest fragments) are the most challenging and are more frequently confused between *AnnualCrop*, *HerbaceousVegetation* and *Pasture*, especially in the multispectral model.

5 Discussion

5.1 Results

A direct comparison of the best models represents following results:

- **RGB model (ResNet18, full fine-tuning):**
  - Validation accuracy: **98.43%**
  - Test accuracy: **98.17%**
- **MS model (late fusion, ResNet18 feature extractor):**
  - Validation accuracy: **91.0%**
  - Test accuracy: **90.8%**

We see, that RGB model shows clearly higher overall accuracy which is expected given that ResNet18 was originally trained on RGB natural images and its filters are well adapted to RGB textures. Nevertheless, the multispectral model provides competitive performance using only 6 selected spectral bands and yields meaningful classifications across all 10 classes. The quantitative metrics and the qualitative inspection of top-5/bottom-5 examples both support these conclusions (plotting.ipynb , validation.ipynb).