



Apocalypse Now

Geospatial
Earth Observation **Big Data & Analytics**

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
MASTER OF DATA SCIENCE AND MACHINE LEARNING

EVANGELOS CHANIADAKIS

ECHANIADAKIS@GMAIL.COM

03400279



SUBMITTED AT
MAY 4, 2025



Contents

1	Introduction	2
2	Dataset Preparation	3
3	Model Architecture	10
4	Training and Evaluation	13
5	Inference and Prediction	17
6	Conclusion	19

Introduction

The proliferation of satellite imagery, has transformed the study of land cover dynamics, enabling precise monitoring of environmental, urban and agricultural changes. This essay delineates a methodological framework for employing deep learning, specifically the U-Net architecture, to conduct semantic segmentation on Sentinel-2 L1C multispectral satellite data. The focus is a practical pipeline for generating accurate land cover maps, eventually tested on the Peloponnesian region of Achaia. Although the data comes from the same year as the training set, the area itself is unseen by the model, providing us with a clear evaluation of its ability to generalize to new geographic contexts.

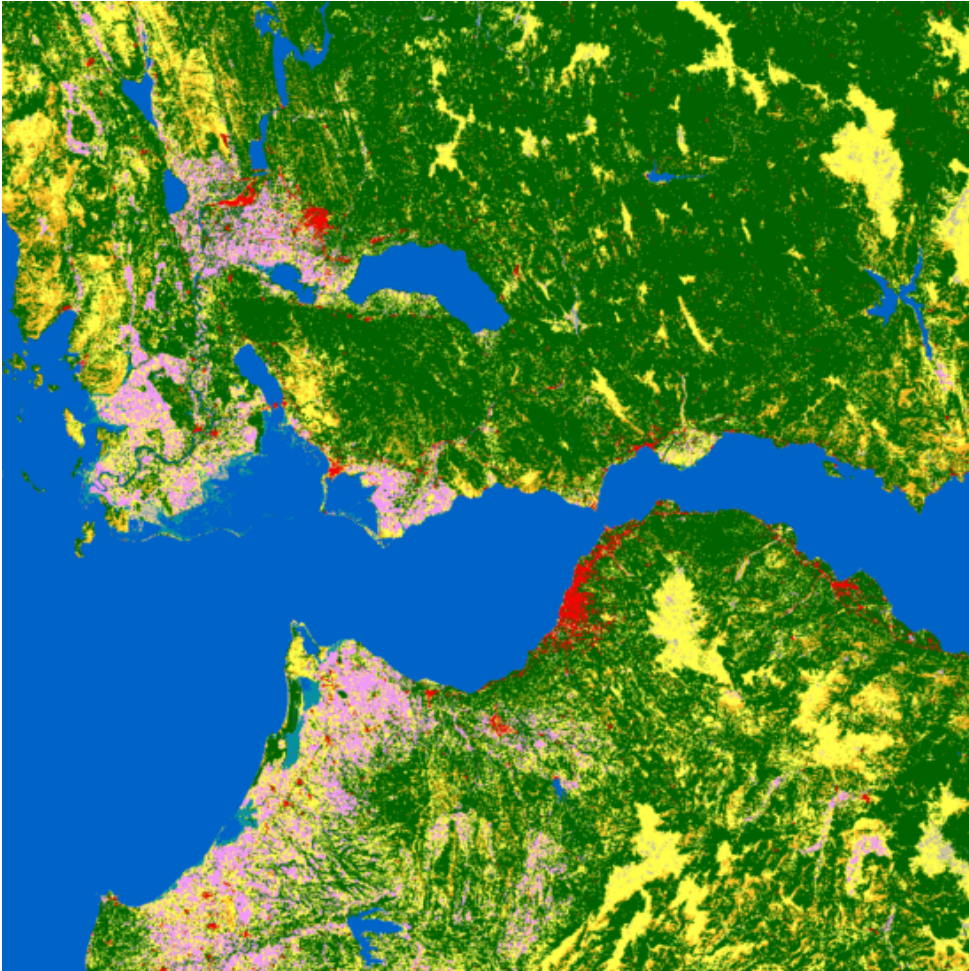


Figure 1.1: Ground truth land cover labels for the Achaia region in 2021

Chapter 2

Dataset Preparation

The dataset used for this study combines satellite observations from the [Copernicus Browser](#) and reference land cover labels provided by the course instructor through [NTUA OneDrive](#). Sentinel 2 Level 1C products were selected with strict consideration for cloud coverage, retaining only those with cloud coverage below 2%. The dataset includes four tiles, namely T34SEJ, T34SFH, T34TEK & T34TFK, each capturing surface conditions on the 25th of September 2021. The selected Sentinel-2 Level-1C tiles cover the greater Larissa area, within which the reference dataset is entirely nested.

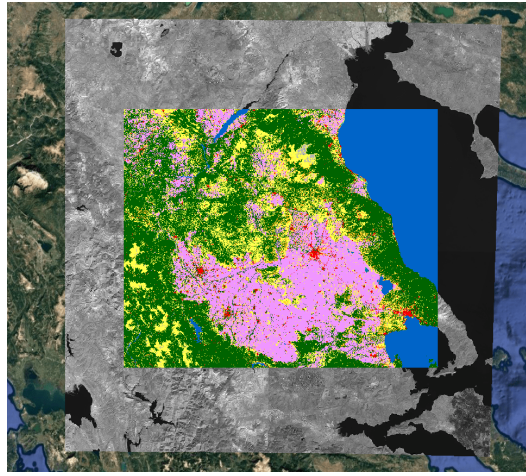


Figure 2.1: Overlay of Sentinel-2 L1C Band 8A imagery (grayscale) from the four selected tiles and the reference land cover labels (colored) on the QGIS software

The reference dataset is distributed as a single-band GeoTIFF file encoded in the EPSG:4326 coordinate reference system and contains per-pixel annotations based on the ESA WorldCover 2021 classification. Each pixel is assigned an integer label representing a distinct land cover class. The embedded metadata defines the meaning of each class, following the official ESA scheme which is detailed in the following table:

Class Value	Land Cover Category
10	Tree cover
20	Shrubland
30	Grassland
40	Cropland
50	Built-up
60	Bare / Sparse vegetation
70	Snow and Ice
80	Permanent water bodies
90	Herbaceous wetland
95	Mangroves
100	Moss and Lichen

Table 2.1: ESA WorldCover 2021 land cover classes.

Preprocessing

To ensure a consistent spatial resolution across all input channels, all thirteen spectral bands from the Sentinel-2 Level-1C products were resampled to a uniform 10-meter grid. Bands originally provided at this resolution (like B02, B03, B04, B08) were retained as-is, while the remaining bands were resampled using bilinear interpolation. Additionally, Brovey pansharpener was selectively applied to bands such as B8A, B11, and B12, enhancing their spatial detail by integrating higher-resolution data from the panchromatic band (B08). This procedure was carried out in `i_pansharpener.py`. Subsequently, the ground truth raster, originally provided in EPSG:4326 and with a spatial resolution of approximately 8.33×10^{-5} degrees per pixel, was reprojected into the Sentinel-2 coordinate reference system (EPSG:32634, UTM Zone 34N) and resampled to a uniform 10-meter resolution using nearest-neighbor interpolation to preserve categorical label integrity. This alignment process, implemented in `ii_crs_alignment.py`, ensures pixel-wise correspondence across inputs and targets, forming a coherent and spatially aligned dataset.

Mosaicking

Following preprocessing, the 4 individual tiles were combined into unified band mosaics inside `iii_merge.py` script. For each of the thirteen spectral bands, the corresponding 10-meter resampled tiles were merged into a single raster that spans the full spatial extent of the study area. To ensure that only relevant regions are retained, a validity mask is derived from the reference label raster by identifying its non-nodata area. This spatial mask is used to constrain the output mosaics to the bounds of annotated land cover data. The result is a set of multi-channel images precisely aligned with the labeled ground truth, suitable for direct patch extraction.

Patch Generation

To prepare the dataset for supervised training, the aligned imagery and labels are partitioned into smaller, fixed-size patches using `iv_build_images.py`. The script applies a sliding window across the full spatial extent, extracting square patches of configurable size and stride. Each patch contains thirteen spectral channels and a corresponding label map, preserving strict pixel-level alignment.

We retain only patches containing at least 10% valid reference pixels, effectively discarding regions dominated by nodata values. In addition, binary validity masks are generated to explicitly denote the usable areas within each patch. This process yields a structured dataset of uniformly sized image-label pairs, ready to be consumed by the model during training and evaluation phases.

Dataset Implementation

To manage our Sentinel-2 patch-based data, we employ the `Sentinel2Dataset` class, which handles 512×512 pixel patches extracted with a stride of 512, which means no spatial overlap. Each patch contains 13 spectral bands along with its corresponding ground truth label, both stored as GeoTIFF files under a structured directory path (`data/patch_dataset-<patch_size>_<stride>`). A binary mask is also included for each sample, delineating valid pixels and excluding no-data regions from loss computation. The `Sentinel2Dataset` class dynamically loads each patch, extracting multi-spectral imagery, semantic labels and validity masks from GeoTIFF files. To ensure radiometric consistency, pixel values are scaled by $\frac{1}{10,000}$, converting raw digital numbers (DNs) to Top-of-Atmosphere (TOA) reflectance values within the physically interpretable range $[0, 1]$, in accordance with Sentinel-2 Level-1C specifications.

Semantic labels are remapped to integer class indices using the `label_to_cls` dictionary, which translates ESA WorldCover codes (e.g., 10, 30, ..., 100) to model-compatible class IDs (e.g., 1, 2, ..., 8). Any unmapped region is assigned to class index 0, representing no-data regions. The dataset includes 8 relevant land cover categories, deliberately omitting classes such as **Snow and Ice**, **Mangroves**, and **Moss and Lichen**, which are absent in our study area of Larissa. Patches are downsampled to 256×256 pixels during loading using OpenCV's `cv2.resize`, with bilinear interpolation for images and nearest-neighbor interpolation for labels and masks to preserve categorical integrity. This down-sampling balances computational efficiency with spatial detail, as full-resolution patches significantly increase memory demands without proportional performance gains. The resulting data format is a tuple of tensors: image ($[13, 256, 256]$), label ($[256, 256]$) & mask ($[256, 256]$).

Splitting

The dataset is partitioned using the `create_split_files` function, which allocates patch indices into training (70%), validation (20%), and test (10%) subsets according to pre-defined split ratios. A user-defined random seed ensures deterministic and reproducible shuffling. To avoid spatial overlap and ensure fair evaluation, the splits are constructed to correspond to geographically distinct regions, thereby mitigating the risk of data leakage. The resulting patch identifiers are saved in plain text files (`train.txt`, `val.txt` & `test.txt`) within the `splits` subdirectory.

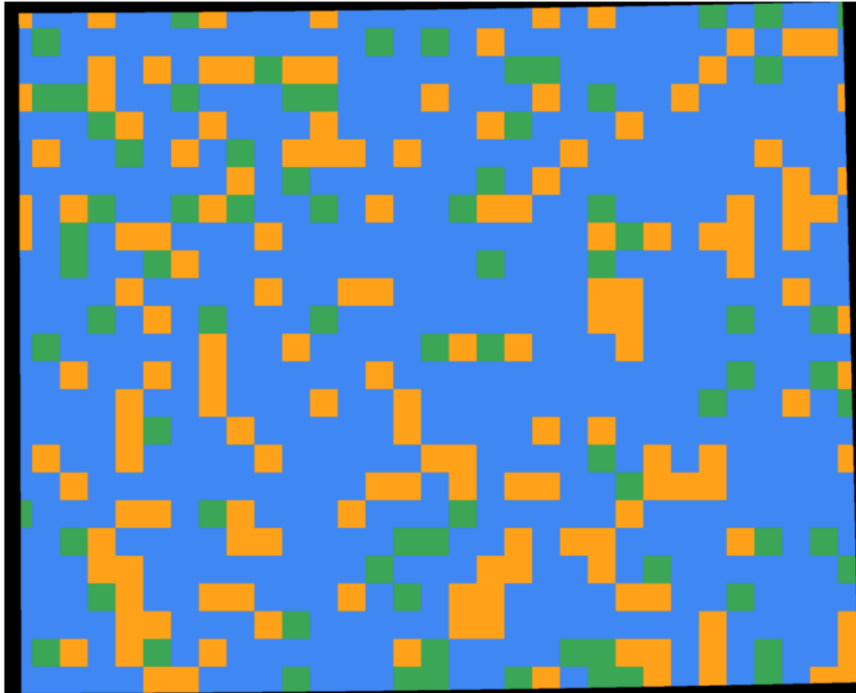


Figure 2.2: Spatial layout of **training**, **validation** & **test** patches split from our dataset.

After completing the data resampling and alignment steps, a slight angular offset becomes visually noticeable between the reference label raster and the underlying Sentinel-2 imagery grid, as shown in Figure 2.2. This subtle rotation arises from the fact that the axes in the geographic coordinate system (latitude and longitude) are not strictly parallel to the axes in the projected UTM coordinate system (X and Y). As a result, a minor misalignment can appear after reprojection, although it does not compromise pixel-level correspondence in practice or the usability of the dataset for the purpose we need it.

Augmentation

To enhance model robustness, the training set undergoes augmentation via `get_training_augmentations` method, implemented using the *Albumentations* library. The augmentation pipeline includes both geometric and radiometric transformations, applied on-the-fly to increase dataset variability while preserving semantic consistency. The transformations, seeded with a configurable seed to ensure reproducibility, are:

- **Geometric Augmentations:**
 - Horizontal Flip (50% probability): Mirrors the image and label horizontally.
 - Vertical Flip (30% probability): Mirrors the image and label vertically.
 - Random 90-Degree Rotation (30% probability): Rotates the image and label by 0, 90, 180, or 270 degrees.
 - Affine Transformation (40% probability): Applies scaling (0.8–1.2), translation (10% of image size), rotation ($\pm 15^\circ$) and shear ($\pm 10^\circ$) with constant border filling (0 for images and masks).
- **Radiometric Augmentations:**
 - Random Brightness and Contrast (50% probability): Adjusts brightness and contrast with limits of ± 0.3 .
 - Gaussian Noise (30% probability): Adds per-channel noise with standard deviation in $[0.03, 0.1]$.
 - Multiplicative Noise (30% probability): Applies per-channel, element-wise noise with a multiplier in $[0.9, 1.1]$.
- **Normalization:** Normalizes each spectral band to zero mean and unit variance using predefined means and standard deviations computed over the whole training set per band, to ensure consistent input distributions.

The validation set is loaded without any augmentation to maintain evaluation consistency. Augmented data is not saved to disk, ensuring efficient storage while leveraging on-the-fly transformations during training.

Data Loading

The dataset is batched and shuffled using PyTorch’s `DataLoader`. The batch size is carefully selected to balance computational efficiency, gradient stability and GPU memory constraints. While larger batches can accelerate training and provide smoother gradient estimates, other than exceeding VRAM limits they may lead to overfitting due to reduced stochasticity. Conversely, very small batch sizes can introduce noisy gradients and unstable convergence. In our implementation, the batch size was empirically determined and fixed to 12, as a compromise that satisfies GPU memory constraints while adhering to principles of stable optimization. Empirical tuning was essential to ensure that the model fits comfortably within available VRAM during training and evaluation, while maintaining a balance between convergence stability and gradient stochasticity. Four worker threads accelerate data fetching and a configurable seed is applied to the data generator to ensure consistent batch ordering across runs. The training loader shuffles patches to promote stochastic gradient descent, while the validation loader maintains a fixed order for reproducible metrics.

Inspection

An inspection of the dataset is critical for understanding its characteristics and potential challenges before we develop our semantic segmentation model for land cover classification on Sentinel-2 imagery and our training & evaluation strategy.

Class Distribution Analysis

Class imbalance is a well-documented challenge in land cover classification, where natural landscapes often exhibit skewed distributions due to geographic and ecological factors. To quantify this, we compute the pixel-wise frequency of each land cover class across the entire reference dataset, as illustrated in Figure 2.3. The dataset comprises 8 classes, mapped from raw labels to indices via the `label_to_cls` dictionary (e.g., Tree cover: 10 \rightarrow 1, Permanent water bodies: 80 \rightarrow 7), with unmapped classes (e.g., Snow and Ice, Mangroves, Moss and Lichen) absent due to the study area’s geographic context. The distribution reveals a pronounced imbalance: Tree cover dominates at 39.5%, followed by Cropland (22.1%), Permanent water bodies (17.1%), Grassland (15.5%) & Shrubland (2.6%). In contrast, Bare/Sparse vegetation (1.9%), Built-up (0.4%), and Herbaceous wetland (\approx 0%) are severely underrepresented. This skewness aligns with ecological expectations for the region, where forested areas and agricultural lands are prevalent, while urban and wetland areas are sparse.

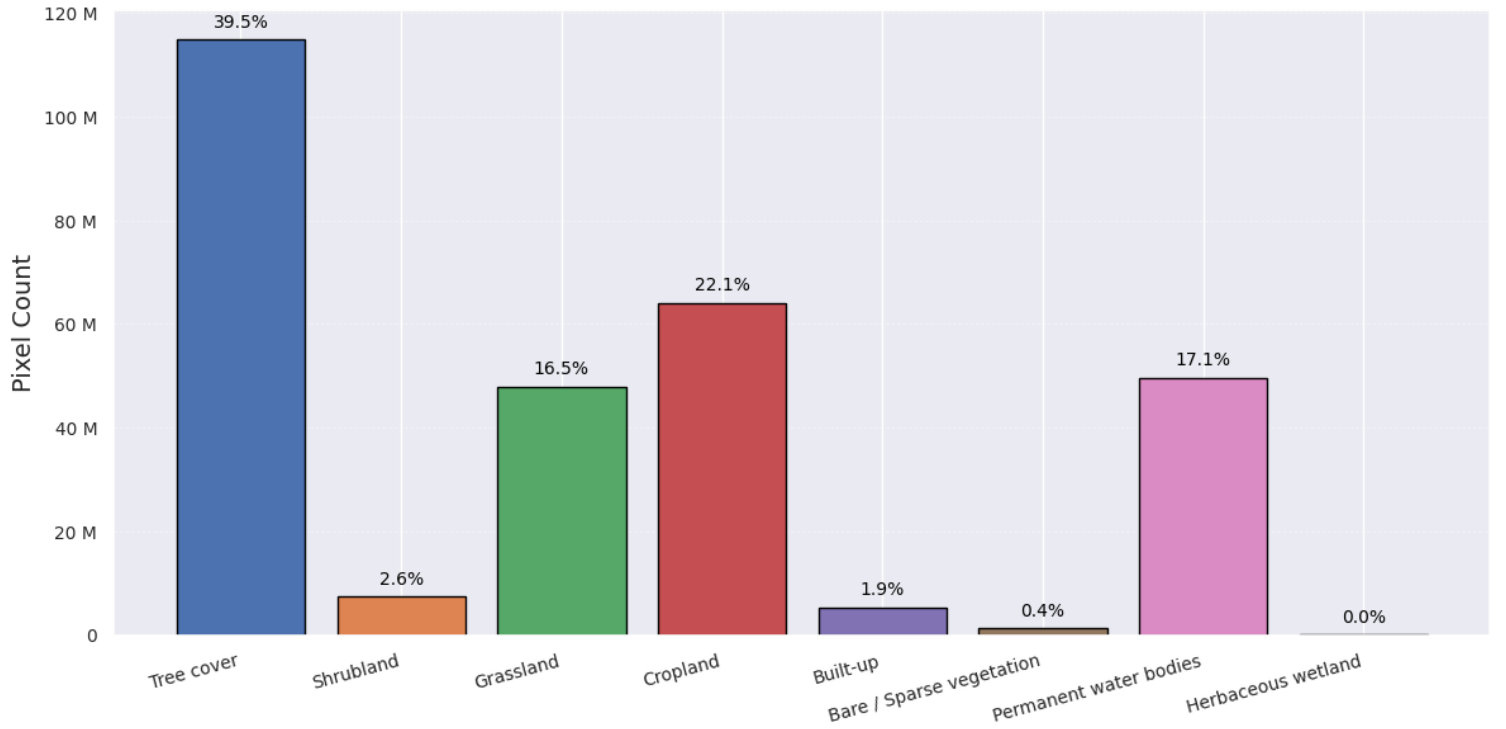


Figure 2.3: Label frequency in the reference raster, highlighting significant class imbalance across the dataset.

Theoretically, such imbalance poses significant challenges for deep learning models. As noted by Johnson and Khoshgoftaar (2019), class imbalance in segmentation tasks can lead to biased models that over-predict majority classes, resulting in high precision but low recall for minority classes. The absence of classes like **Snow and Ice** further constrains the model’s generalizability to diverse geographic regions, a limitation we acknowledge in this study and one that may concern us even more later, when we predict on the unseen region of Achaia.

In our effort to mitigate these issues, we adopt several strategies informed by theory:

- **Class Weighting:** The `AdaptiveLoss` (see Chapter 3 Section 3) incorporates class weights in its Cross-Entropy component ($w_c = 1/(f_c + 10^{-6})$), an approach inspired by Lin et al. (2017), to prioritize rare classes during training.
- **Metric Selection:** We prioritize weighted F1 score and IoU, which handle class imbalance by emphasizing recall and overlap, monitoring per-class F1 scores to ensure rare classes are addressed.
- **Data Sampling Strategies:** Although ultimately not used in our pipeline due to lack of performance improvement, the `compute_pixel_based_patch_weights` method offers a potential strategy for weighting patches based on their class composition. Such methods, as discussed by Buda, Maki, and Mazurowski (2018), are used to oversample patches containing rare classes, further balancing the training distribution.

Patch-Level Analysis

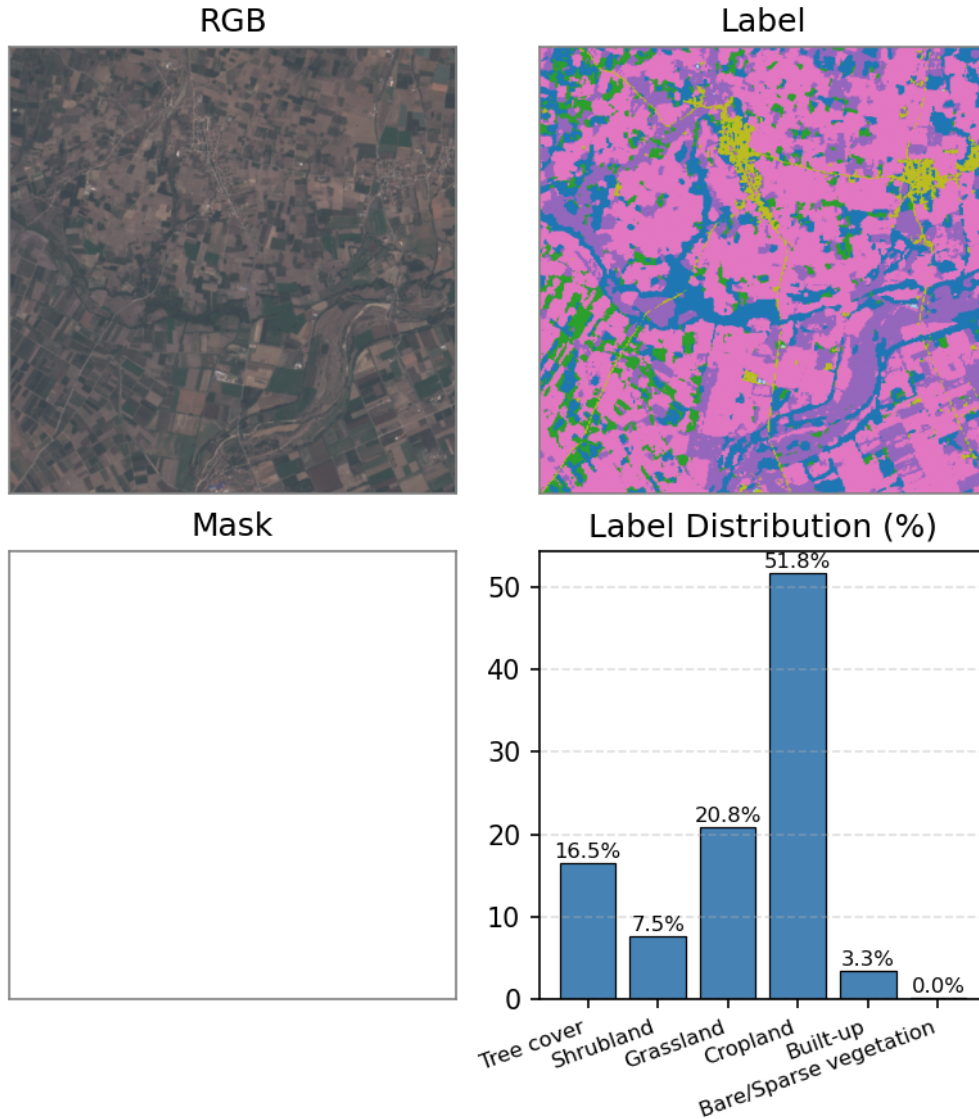



Figure 2.4: Sample patch from the dataset, including RGB visualization (bands B04, B03, B02), ground truth labels, validity mask, and label distribution histogram, illustrating spatial composition and local class imbalance.

We also inspect individual patches to understand their spatial composition, label quality and class distribution at a finer scale, but more importantly to ensure the data region aligns perfectly with the label area. This is achieved through the visualization of randomly selected patches, as shown in Figure 2.4, which includes four components: the RGB rendering of the Sentinel-2 imagery (using bands B04, B03, B02), the semantic label map, the binary validity mask and a histogram of the patch’s class distribution.

The alignment between the RGB imagery and the label map confirms the integrity of the patch extraction process. The binary validity mask, fully white in this example, indicates that all pixels have valid labels, thus no `nodata` regions in this specific sample. The histogram quantifies the class distribution within the patch, revealing a local imbalance similar to the global distribution, with rare classes like Bare/Sparse Vegetation ($\approx 0.0\%$) almost absent in this sample.

Our inspection highlights important theoretical insights that inform both model architecture and evaluation. Transitions between spectrally similar classes, such as Tree cover and Grassland, pose challenges for accurate boundary prediction. Our observation motivates the inclusion of attention mechanisms in the decoder, as they enhance feature discrimination and improve boundary precision. Ultimately, the insights we derive from our inspection ground our design choices and support the development of a model that balances performance across all classes, handles spatial heterogeneity, and generalizes effectively to unseen regions.

Model Architecture

he semantic segmentation model presented in this study is a sophisticated extension of the U-Net architecture, specifically engineered for processing multi-spectral satellite imagery, such as that provided by Sentinel-2. This model is designed to perform land cover classification by integrating deep learning components, including a ResNet backbone, Atrous Spatial Pyramid Pooling, Convolutional Block Attention Modules & an adaptive hybrid loss function. These components collectively address challenges such as class imbalance, boundary sensitivity, and the need for multi-scale contextual understanding in Earth observation tasks. Below, we provide a detailed exposition of each component, elucidating their roles and implementation details.

Encoder: ResNet Backbone

The encoder leverages a Residual Network (ResNet), a highly influential architecture introduced by K. He et al. (2016), which is foundational in computer vision due to its ability to train very deep networks efficiently. By introducing identity-based residual connections, ResNets effectively mitigate the vanishing gradient problem, enabling deeper models such as ResNet-101 to be trained successfully. These architectures have been extensively validated on large-scale benchmarks, most notably ImageNet (Deng et al. (2009)), where they consistently demonstrate superior feature extraction capabilities. In our context, the ResNet encoder is tasked with learning hierarchical representations from 13-channel Sentinel-2 imagery. Its depth can be configured (ResNet-18 to ResNet-152) depending on performance and resource constraints. In our experiments, running on a consumer GPU, we concluded that ResNet-101 has the best results.

Our encoder leverages pretrained ImageNet weights to initialize all layers except the first convolutional layer, which is modified to accommodate 13 spectral bands rather than the default 3 RGB channels. This modification retains the original weight structure for RGB by averaging and replicating the learned kernels across the extra input channels, ensuring our network benefits from transfer learning despite the input mismatch. To preserve spatial resolution in deep layers, the stride in the last residual block is reduced to 1, effectively halving the downsampling compared to standard configurations. The ResNet’s fully connected layer is replaced with an identity operation to retain spatial feature maps. Feature maps from intermediate layers are stored for skip connections to the decoder, in alignment with the U-Net design introduced by Ronneberger, Fischer, and Brox (2015), which promotes fine-grained spatial recovery. Finally, our model expects input tensors of shape $[N, 13, H, W]$ where N is the batch size, 13 corresponds to Sentinel-2’s spectral bands (B1 to B12 & B8A) and H, W are the spatial dimensions, in our case 256×256 pixels, because we feed our model with downsampled patches (of initial size 512×512 pixels).

Atrous Spatial Pyramid Pooling

To capture multi-scale context effectively, we integrate Atrous Spatial Pyramid Pooling (ASPP), a technique first introduced by Chen et al. (2017). ASPP employs dilated convolutions with various dilation rates to enlarge the receptive field without increasing the number of parameters or reducing spatial resolution. This mechanism is particularly

fit for our scenario of satellite imagery, where object scales vary significantly and global context is essential for disambiguating land cover types.

Our ASPP module consists of five parallel branches:

- Four convolutions with dilation rates $\{1, 6, 12, 18\}$, enabling multi-scale feature extraction.
- One global average pooling branch followed by a 1×1 convolution and bilinear upsampling to reintroduce global context.

All outputs are concatenated and passed through a 1×1 convolution to unify channel dimensions. This design ensures a balanced fusion of local and global features, critical for the diverse spatial structures present in Earth observation data.

Decoder: Attention-Augmented Upsampling

The decoder mirrors the classic U-Net structure, progressively upsampling and merging encoder features to recover spatial resolution. To improve the selectivity and robustness of feature fusion, we integrate Convolutional Block Attention Modules (CBAM), proposed by Woo et al. (2018). CBAM applies both channel-wise and spatial attention to refine feature maps, helping the network focus on semantically important regions while suppressing irrelevant or noisy activations. This is a vital improvement when dealing with spectral variability in satellite imagery. It could also help us dial down the degrading effect of cloud cover and shadows in the performance but we carefully chose our dataset to not deal with this kind of problem. To prevent overfitting and enhance generalization, dropout with probability $p = 0.2$ is applied after the first convolution in each decoder block and before the final 1×1 convolution.

Each decoder block performs the following operations:

- Applies CBAM to the incoming feature map, computing both channel and spatial attention maps.
- Concatenates the upsampled decoder feature with the corresponding encoder skip feature.
- Refines the fused features via two convolutional layers with batch normalization and ReLU.
- Uses bilinear interpolation for spatial upsampling.
- Adds a residual shortcut with 1×1 convolution if needed to match dimensions.

The decoder terminates with a 1×1 convolution that projects features into class logits, ready for softmax activation during training.

Design of the Loss Objective

Optimizing segmentation performance in the presence of severe class imbalance and thin structures (e.g., rivers, roads) is by no means a trivial task and it remains a big challenge for us. In our effort to address it, we are using a weighted combination of three loss functions: **Cross-Entropy**, **Dice** & **Focal loss**, each of which is designed to counteract specific failure modes. The Dice Loss incorporates a mask of shape $[N, H, W]$ to exclude invalid pixels (nodata regions), aligning with the `ignore_index=0` in Cross-Entropy Loss, in order to ensure robust handling of incomplete or preprocessed satellite imagery. The Focal Loss uses $\gamma = 2.0$ and $\alpha = 0.25$, while the Dice Loss includes a smoothing factor of 1.0 to stabilize division. The adaptive hybrid loss initializes weights as $w_1 = 0.2$ for Cross-Entropy, $w_2 = 0.5$ for Dice and $w_3 = 0.3$ for Focal, which are then updated during training and normalized via `softplus`. Our formulation draws inspiration from

multi-task loss weighting strategies based on uncertainty, as proposed by Kendall, Gal, and Cipolla (2018).

We define a composite loss:

$$\mathcal{L}_{\text{total}} = w_1 \mathcal{L}_{\text{CE}} + w_2 \mathcal{L}_{\text{Dice}} + w_3 \mathcal{L}_{\text{Focal}},$$

where w_1, w_2, w_3 are softplus-normalized learnable weights updated via gradient descent. Each component serves a role:

1. **Cross-Entropy:** Provides stable pixel-level supervision and benefits from class weighting to address imbalance.
2. **Dice Loss:** Targets overlap accuracy, especially useful for underrepresented or small-area classes.
3. **Focal Loss:** Reduces dominance of easy examples and focuses on hard-to-classify pixels via the $(1 - p_t)^\gamma$ modulation.

The dynamic adjustment of loss weights allows the model to adapt its focus among classification accuracy, spatial overlap and sample difficulty throughout training. This flexibility enhances the model’s ability to generalize, especially across heterogeneous geographic regions and varied land cover types.

Training and Evaluation

The training and evaluation pipeline, implemented in the `train_model` function, orchestrates the optimization of the `UNetResNet` model over a specified number of epochs. The model is trained on a GPU if available, falling back to CPU otherwise. After heavy experimentation, we deemed appropriate to initialize our model with a ResNet-101 backbone pretrained on ImageNet, which yields the best performance. The model is optimized using the `AdaptiveLoss`, which combines Cross-Entropy, Dice and Focal losses with learnable weights normalized via softplus. Optimization is performed using the Adam optimizer with an initial learning rate of 5×10^{-5} . To ensure convergence, a `ReduceLROnPlateau` scheduler reduces the learning rate by a factor of 0.5 if the validation *F1score* fails to improve for 2 consecutive epochs. The training loop includes gradient clipping implicitly through PyTorch’s automatic differentiation and GPU memory is efficiently managed by clearing the cache at the end of each epoch. To avoid overfitting, early stopping is employed with a patience of three epochs. If the validation F1 score does not improve within this window, training is halted and the model checkpoint with the highest validation F1 score is saved, along with associated metadata (epoch, F1 score, IoU, accuracy & hyperparameters).

Each epoch consists of a training and validation phase:

- **Training Phase:** The model is set to training mode, enabling dropout and batch normalization updates. For each batch, images, labels and masks are moved to the device and the model computes predictions. `AdaptiveLoss` is calculated, back-propagated and the optimizer updates the model parameters. Training loss and predictions are accumulated for metric computation.
- **Validation Phase:** The model is switched to evaluation mode, disabling dropout and batch normalization updates. Predictions and losses are computed without gradient tracking to reduce memory usage. Validation metrics are calculated over the entire validation set.

In the end of each epoch we log detailed metrics, including training and validation loss, F1 score, accuracy & IoU as well as the adaptive loss trainable weights and per-class F1 scores to detect classes with poor performance.

The model’s performance is assessed using a suite of evaluation metrics implemented via the `torchmetrics` library, ensuring a comprehensive analysis of its effectiveness in our given task. These metrics are computed exclusively on valid pixels, as defined by the binary mask, to exclude `nodata` regions, ensuring that the evaluation reflects the model’s performance on meaningful regions only. We select three primary metrics, each addressing distinct aspects of segmentation quality.

1. **Weighted F1 Score:** The F1 score, defined as the harmonic mean of precision and recall, is computed per class and aggregated with weights proportional to class frequencies, ignoring the `nodata` class (index 0). For a given class c , they are defined as:

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}, \quad \text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c},$$

where TP_c , FP_c , and FN_c represent the true positives, false positives and false negatives for class c .

The F1 score for class c is then:

$$F1_c = 2 \cdot \frac{\text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}.$$

The weighted F1 score aggregates these per-class scores, weighted by the number of true instances N_c for each class:

$$\text{Weighted F1} = \frac{\sum_{c=1}^C N_c \cdot F1_c}{\sum_{c=1}^C N_c},$$

where C is the number of classes, which is 8 in our scenario. This metric, which balances precision and recall, is particularly suited for imbalanced datasets, as it emphasizes performance on rare classes through weighting, a strategy emphasized by H. He and E. A. Garcia (2009) as particularly effective in addressing class imbalance. It serves as the primary criterion for early stopping and model selection, ensuring that the model prioritizes balanced performance across all classes.

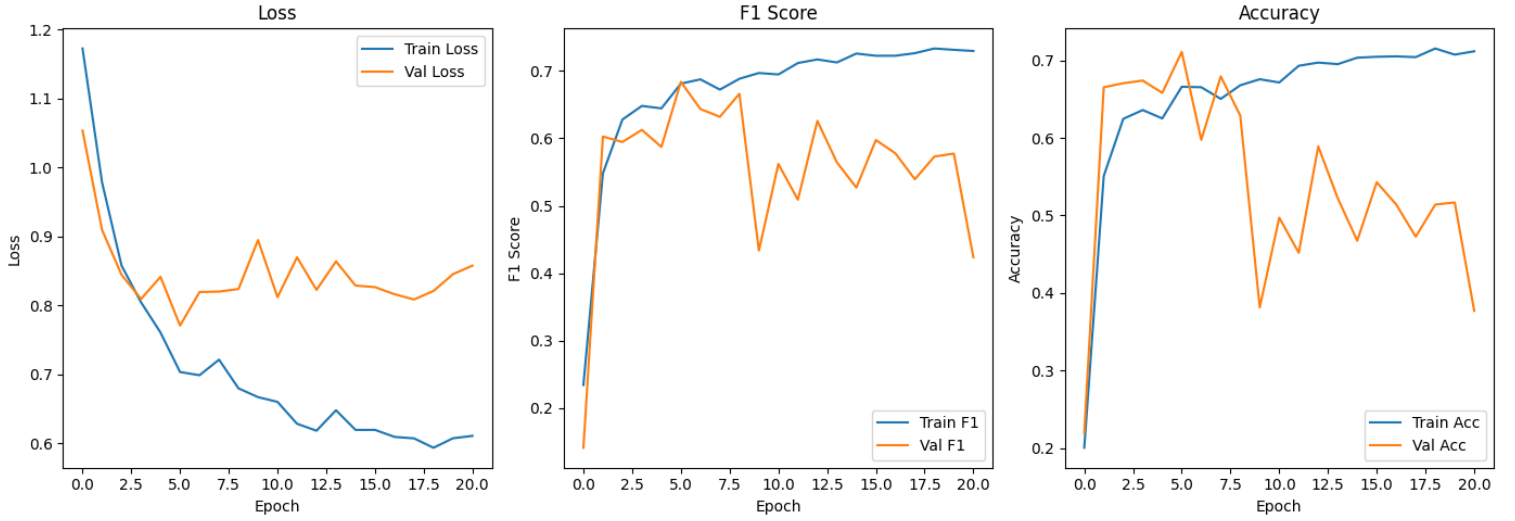


Figure 4.1: U-Net Training Metrics Visualization

2. **Weighted Intersection over Union (IoU):** Also known as the Jaccard Index, IoU quantifies the spatial overlap between predicted and ground truth segmentation maps, making it a critical measure for assessing pixel-wise accuracy in segmentation tasks. For class c , IoU is defined as:

$$\text{IoU}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c + \text{FN}_c},$$

where the denominator represents the union of predicted and ground truth pixels for class c . The weighted IoU is computed similarly to the weighted F1 score:

$$\text{Weighted IoU} = \frac{\sum_{c=1}^C N_c \cdot \text{IoU}_c}{\sum_{c=1}^C N_c}.$$

This metric is essential in remote sensing applications, as it directly evaluates the model's ability to delineate class boundaries accurately, which is crucial for land cover mapping, as discussed in the evaluation framework of S. Garcia et al. (2010). The weighting ensures that rare classes, such as Built-up or Herbaceous wetland, contribute proportionally to their prevalence, mitigating the dominance of majority classes like Tree cover.

3. **Micro Accuracy:** This metric computes the overall proportion of correctly classified pixels across all non-background classes, providing a holistic measure of classification correctness. It is defined as:

$$\text{Micro Accuracy} = \frac{\sum_{c=1}^C \text{TP}_c}{\sum_{c=1}^C (\text{TP}_c + \text{FN}_c)},$$

where the numerator aggregates true positives across all classes, and the denominator represents the total number of true instances. Unlike macro-averaged metrics, micro accuracy treats all pixels equally, offering a complementary perspective to the weighted F1 score and IoU by focusing on overall pixel-level correctness, aligning with the systematic comparison presented by Sokolova and Lapalme (2009). While less sensitive to class imbalance, it provides a baseline understanding of the model’s performance across the entire dataset.

Following model training, we render model predictions on randomly selected validation patches with `visualize_prediction` method to get some qualitative insight on our model accuracy, computing it as the proportion of correctly classified valid pixels. This helped us during development phase to identify systematic failures, such as misclassification of rare classes. Reproducibility is enforced via a global configurable seed, applied consistently across data shuffling, augmentation and model initialization.

The training pipeline is implemented in `vii_train_unet.py`, with dependencies on `v_prepare_training_data.py` for dataset handling and `vi_sentinel2_unet.py` for the model architecture and loss definitions.

Table 4.1: Training metrics per epoch

Epoch	Train Loss	Val Loss	Train F1	Val F1	Train Acc	Val Acc
1	1.1729	1.0537	0.2344	0.1414	0.2008	0.2201
2	0.9786	0.9095	0.5481	0.6026	0.5510	0.6653
3	0.8578	0.8444	0.6278	0.5945	0.6245	0.6704
4	0.8044	0.8092	0.6480	0.6125	0.6359	0.6738
5	0.7604	0.8414	0.6444	0.5873	0.6251	0.6580
6	0.7032	0.7706	0.6810	0.6836	0.6659	0.7108
7	0.6985	0.8193	0.6874	0.6435	0.6653	0.5974
8	0.7211	0.8199	0.6722	0.6319	0.6503	0.6793
9	0.6794	0.8237	0.6883	0.6660	0.6678	0.6288
10	0.6667	0.8948	0.6968	0.4337	0.6756	0.3816
11	0.6598	0.8119	0.6947	0.5620	0.6714	0.4972
12	0.6281	0.8697	0.7116	0.5090	0.6929	0.4520
13	0.6179	0.8224	0.7170	0.6259	0.6970	0.5890
14	0.6476	0.8637	0.7124	0.5644	0.6950	0.5223
15	0.6191	0.8287	0.7257	0.5270	0.7033	0.4675
16	0.6191	0.8264	0.7224	0.5975	0.7045	0.5428
17	0.6090	0.8161	0.7224	0.5778	0.7051	0.5142
18	0.6068	0.8085	0.7262	0.5393	0.7041	0.4727
19	0.5932	0.8208	0.7331	0.5730	0.7152	0.5141
20	0.6070	0.8452	0.7312	0.5774	0.7073	0.5168

Validation metrics initially seems to outperform some training metrics due to data augmentations applied only to the training set, increasing its difficulty. As training progresses, the model improves steadily, reaching peak validation performance at epoch 6. Beyond this point, training metrics continue to rise while validation performance fluctuates, suggesting overfitting.

Table 4.2: Per-class F1 scores

Epoch	Tree cover	Shrubland	Grassland	Cropland	Built-up	Bare land	Water	Wetland
1	0.2151	0.0000	0.0022	0.0401	0.0000	0.0004	0.3088	0.0000
2	0.7355	0.0372	0.0518	0.7092	0.0047	0.0020	0.8833	0.0000
3	0.7178	0.0003	0.0089	0.7001	0.0503	0.0068	0.9316	0.0966
4	0.7380	0.0518	0.0172	0.7185	0.1922	0.0003	0.9380	0.0272
5	0.7117	0.0048	0.0004	0.6857	0.1285	0.0187	0.9225	0.0302
6	0.7780	0.0569	0.3091	0.7352	0.0482	0.2146	0.9544	0.0810
7	0.6879	0.1094	0.3253	0.6860	0.1748	0.0534	0.9551	0.0413
8	0.7445	0.0491	0.1299	0.7084	0.1841	0.0593	0.9366	0.0260
9	0.7194	0.1183	0.3619	0.6957	0.1960	0.0173	0.9661	0.0938
10	0.2247	0.0672	0.1179	0.6761	0.1697	0.0018	0.9645	0.0767
11	0.4977	0.0822	0.1957	0.7189	0.2418	0.1267	0.9696	0.0515
12	0.4700	0.0749	0.0562	0.6488	0.1791	0.0732	0.9701	0.0925
13	0.6920	0.1037	0.1538	0.7138	0.2644	0.0290	0.9699	0.0668
14	0.5735	0.0876	0.0425	0.7212	0.2677	0.0376	0.9708	0.0572
15	0.4339	0.0786	0.1261	0.7221	0.2479	0.0714	0.9710	0.0727
16	0.6481	0.0927	0.1044	0.6979	0.2755	0.0582	0.9711	0.0636
17	0.5965	0.0869	0.1014	0.6943	0.2984	0.1052	0.9728	0.0751
18	0.4544	0.0814	0.1530	0.7121	0.2928	0.2561	0.9741	0.0932
19	0.5940	0.0865	0.0814	0.6868	0.2902	0.2911	0.9749	0.1006
20	0.6072	0.0849	0.0719	0.6983	0.2467	0.1348	0.9758	0.0985

Per-class F1 scores reveal strong performance in dominant classes like *Water* and *Tree cover*. Less represented categories such as *Shrubland* and *Grassland* show weaker and more variable scores. The overall trend indicates that well-represented classes benefit earlier and more consistently from training, while rare classes improve later, after overfitting has already occurred.

Inference and Prediction

The inference pipeline was developed to perform semantic segmentation over previously unseen Sentinel-2 data. The target area, corresponding to the Sentinel-2 tile T34SEH covering Achaia, underwent the same preprocessing workflow as used during training. This includes pansharpener to ensure 10m spatial resolution for all 13 bands and alignment of the reference raster to the tile's coordinate system and resolution. The processed tile was then partitioned into non-overlapping fixed-size patches of 512×512 pixels, as in the training period. They were saved to disk to mitigate memory pressure during inference and each patch individually got loaded, normalized and forwarded through our pretrained UNetResNet model. Predictions were performed using the class with highest softmax score (via `argmax`) and then assembled into a full-resolution prediction map. To evaluate how well the model performed in this task, we used a ground truth map of this region provided by the course instructor through [NTUA OneDrive](#). Since the reference data originally was in a different resolution and projection, we reprojected to our preferred CRS (EPSG:32634, UTM Zone 34N) and re-sampled it to 10m spatial resolution, so that it lines up perfectly with our data, allowing for accurate comparison with our predictions. The code for this section is implemented in `viii_predict_unseen_region.py`

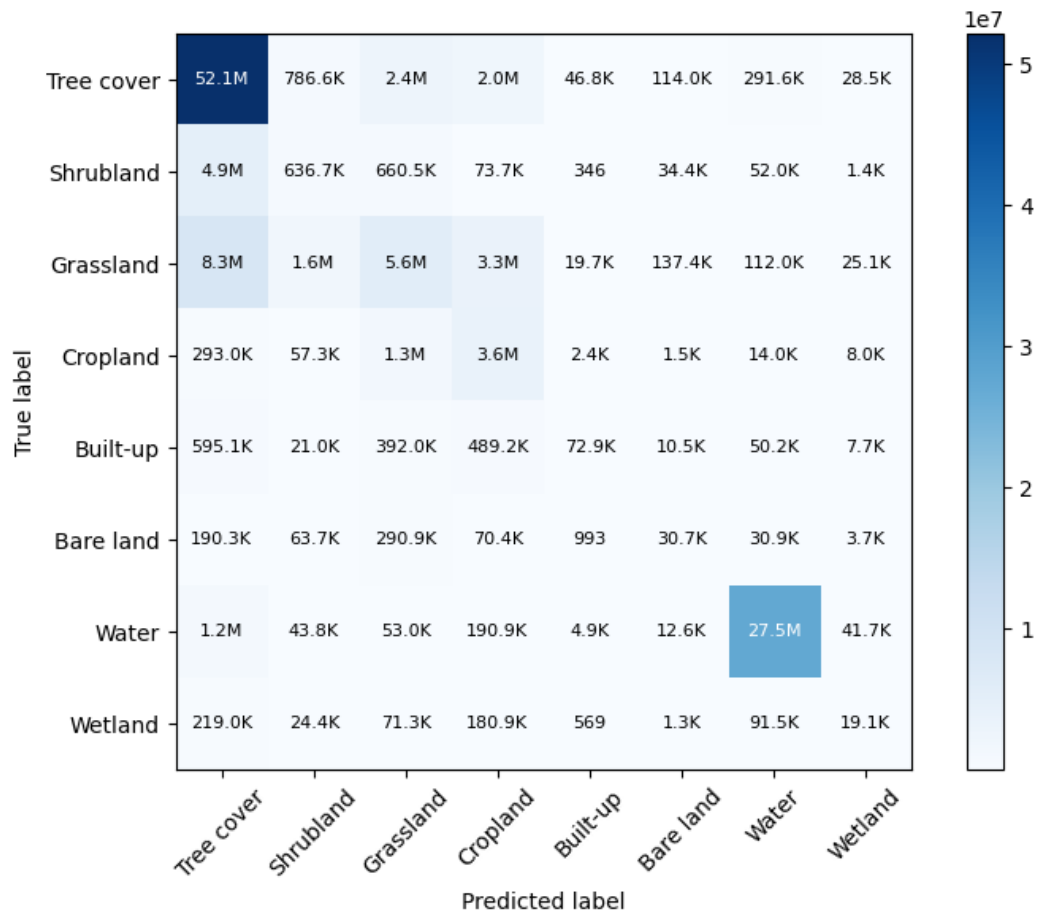


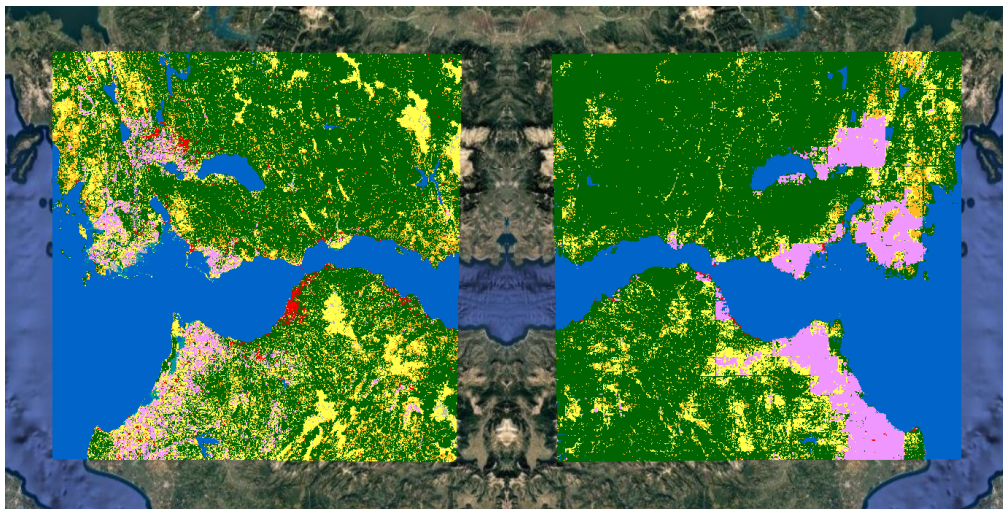
Figure 5.1: Confusion matrix for predicted vs. ground truth classes in the Achaia region.

To assess the model’s performance over the unseen region, we conducted a pixel-level comparison between the predicted and reference labels. The table below presents the precision, recall, and F1-score for each land cover class.

Table 5.1: Classification report for the Achaia test tile, colored using the class palette from the reference GeoTIFF.

Class	Precision	Recall	F1-score	Support
Tree cover	0.77	0.90	0.83	57,859,511
Shrubland	0.19	0.10	0.13	6,338,750
Grassland	0.52	0.29	0.37	19,098,128
Cropland	0.36	0.68	0.47	5,261,833
Built-up	0.49	0.04	0.08	1,638,684
Bare land	0.09	0.05	0.06	681,529
Water	0.98	0.95	0.96	29,029,912
Wetland	0.14	0.03	0.05	608,136
Accuracy			0.74	120,516,483
Macro Avg	0.44	0.38	0.37	120,516,483
Weighted Avg	0.72	0.74	0.72	120,516,483


The model demonstrates strong predictive performance on dominant land cover classes such as **Tree cover**, **Water** & **Cropland**, achieving high precision and recall across these categories. In contrast, underrepresented and spatially fragmented classes, including **Wetland**, **Bare land** & **Built-up**, exhibit considerably lower recall, suggesting that the model struggles to consistently detect smaller or less distinctive regions. As previously discussed, classes like **Snow and Ice**, **Mangroves** & **Moss and Lichen** were entirely absent from the training set and are therefore excluded from the evaluation phase. The macro-averaged metrics highlight this class imbalance, while the overall accuracy remains high due to the model’s strong performance on the dominant classes. These results indicate that the model generalizes reasonably well to unseen areas, particularly for prevalent land cover types. However, performance on minority classes may benefit from additional strategies, such as more aggressive and focused data augmentation. The final prediction map was exported as a GeoTIFF, preserving georeferencing metadata so that it can directly be viewed in GIS software, along with the ground truth.



(a) Achaia Ground Truth Land Cover

(b) Mirrored Achaia Predicted Land Cover

Conclusion

n this work we successfully developed and validated an automated pipeline for large-scale land cover classification using Sentinel-2 Level-1C imagery, leveraging a deep learning-based semantic segmentation approach. The proposed system, centered on a modified U-Net architecture with a ResNet-101 encoder, Atrous Spatial Pyramid Pooling (ASPP) and attention-augmented decoding, achieved robust performance on the unseen Achaia region, with a weighted F1 score of 0.72 and an overall accuracy of 0.74. The model excelled in classifying dominant land cover types such as **Tree cover** (F1: 0.83) and **Water** (F1: 0.96), demonstrating its ability to generalize to new geographic contexts. However, performance on minority classes like **Wetland** (F1: 0.05) and **Built-up** (F1: 0.08) revealed limitations in handling severe class imbalance and spatially fragmented regions, consistent with challenges noted by Johnson and Khoshgoftaar (2019).

Our pipeline adopts a modular design encompassing preprocessing, dataset construction, model training and inference, enabling systematic experimentation and optimization. To address class imbalance and enhance feature representation, we experimented with techniques such as ASPP, CBAM and adaptive hybrid loss, among others. However, most of our experimentations were ultimately excluded from the final pipeline as they did not yield any improvement in performance. While the selected enhancements provided slight benefits, class imbalance remains a persistent challenge in segmentation performance.

Bibliography

- [1] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. “A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks”. In: *Neural Networks* 106 (2018), pp. 249–259. DOI: [10.1016/j.neunet.2018.07.011](https://doi.org/10.1016/j.neunet.2018.07.011). URL: <https://doi.org/10.1016/j.neunet.2018.07.011>.
- [2] Li-Chiang Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2017), pp. 834–848.
- [3] Jia Deng et al. “ImageNet: A Large-Scale Hierarchical Image Database”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 248–255.
- [4] Salvador Garcia et al. *Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power*. Vol. 180. 10. Elsevier, 2010, pp. 2044–2064. DOI: [10.1016/j.ins.2009.12.010](https://doi.org/10.1016/j.ins.2009.12.010). URL: <https://doi.org/10.1016/j.ins.2009.12.010>.
- [5] Haibo He and Edwardo A. Garcia. “Learning from Imbalanced Data”. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: [10.1109/TKDE.2008.239](https://doi.org/10.1109/TKDE.2008.239). URL: <https://doi.org/10.1109/TKDE.2008.239>.
- [6] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.
- [7] Justin M. Johnson and Taghi M. Khoshgoftaar. “Survey on Deep Learning with Class Imbalance”. In: *Journal of Big Data* 6.1 (2019), p. 27. DOI: [10.1186/s40537-019-0192-5](https://doi.org/10.1186/s40537-019-0192-5). URL: <https://doi.org/10.1186/s40537-019-0192-5>.
- [8] Alex Kendall, Yarin Gal, and Roberto Cipolla. “Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7482–7491.
- [9] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2017, pp. 2980–2988. DOI: [10.1109/ICCV.2017.324](https://doi.org/10.1109/ICCV.2017.324). URL: <https://doi.org/10.1109/ICCV.2017.324>.
- [10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, pp. 234–241.
- [11] Marina Sokolova and Guy Lapalme. “A Systematic Analysis of Performance Measures for Classification Tasks”. In: *Information Processing & Management* 45.4 (2009), pp. 427–437. DOI: [10.1016/j.ipm.2009.03.002](https://doi.org/10.1016/j.ipm.2009.03.002). URL: <https://doi.org/10.1016/j.ipm.2009.03.002>.
- [12] Sanghyun Woo et al. “CBAM: Convolutional Block Attention Module”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 3–19.