Satellite Image Segmentation

Advanced Computer Image Processing - Final Project

Introduction

This project focuses on semantic segmentation of satellite images, which consists in assigning a semantic class to each region of a satellite image, such as *roads*, *buildings*, *fields*, or *wooded areas*. This technique is widely used across many application domains. In precision agriculture, it enables the estimation of cultivated areas and the assessment of crop health. In cartography, it is commonly employed to automatically detect infrastructure elements such as roads or buildings. It is also used in economic intelligence, where some financial institutions rely on the automatic detection of active industrial sites to anticipate the economic health of regions or entire countries.

Within this project, several image segmentation methods are developed, implemented, and compared, including histogram-based approaches and convolutional neural networks (CNNs). The work deliberately emphasizes methodological aspects rather than raw performance, with the goal of maintaining a strong pedagogical focus. In addition, constraints related to limited computational resources are taken into account, particularly for CNN training. As a result, the entire project codebase is optimized to minimize computational cost as much as possible.

The report begins with a presentation of the dataset, followed by data preprocessing and feature extraction stages. The different implemented segmentation methods are then described in detail. Special attention is given to the inference and post-processing phases, before concluding with an analysis of the results and performance of the various approaches.

I. Dataset Presentation

Description and Attribution

The dataset used in this project comes from LandCover.ai, created by Adrian Boguszewski, Dominik Batorski, Natalia Ziemba-Jankowska, Tomasz Dziedzic, and Anna Zambrzycka, and is distributed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International *(CC BY-NC-SA 4.0)* license.

The dataset contains approximately **10,000 images** of size 512 × 512 pixels, representing satellite imagery of Poland (Central Europe) annotated into five land-cover classes: Field, Woodland, Water, Building, and Road. The full dataset covers an area of approximately 216 km², with a spatial resolution ranging from 25 cm to 50 cm per pixel. The images are provided in JPEG (.jpg) format as three-channel RGB arrays of shape (512, 512, 3), while each image is paired with a pixel-wise labeled segmentation mask in PNG (.png) format. The masks are single-channel images of shape (512, 512), with integer values from 0 to 4 encoding the corresponding land-cover class. These ground truth masks are used both for training semantic segmentation models and for evaluating their performance within the scope of this project.

The dataset has also been published on Kaggle : LandCover_Kaggle.ai. In this version, the author provides an additional Python script allowing the dataset to be automatically split into training, validation, and test subsets with a 70 / 15 / 15 ratio.

Analysis and Preprocessing

This dataset provides a sufficient amount of data to carry out the analyses required for the successful execution of our project, while remaining relatively simple, with only five well-defined classes and no unnecessary or complex metadata. However, attention must be paid to the highly **imbalanced nature of the classes** in the dataset. Indeed, the Field and Woodland classes alone account for nearly 91% of the total area across all images, whereas Building and Road represent less than 1% and 2% of the area, respectively. This imbalance is logical, as the dataset images are primarily taken from a relatively rural environment where buildings and roads are minor features. Consequently, metrics such as overall accuracy are largely meaningless in this context, since it would be possible to achieve over 90% accuracy by labeling only the dominant classes, Field and Woodland.

Nevertheless, the minority classes, Building and Road, are still relatively well-represented across images: more than 16% of the images contain at least one pixel labeled as Building, while over 37% contain roads. This ensures that effective training is still possible. To further emphasize the presence of minority classes, **data augmentation** was applied, including rotation and zoom, specifically on images containing Roads or Buildings. In particular, images containing Buildings underwent more extensive augmentation, with zooming and blurring applied to regions containing Building pixels, effectively increasing their relative area.

Finally, many images labeled exclusively as Woodland or Field were removed from the dataset to retain a more informative sample. Only images containing at least some Buildings, Roads, or Water were selected for the remainder of the project, resulting in approximately **8,000 usable images** for our work.

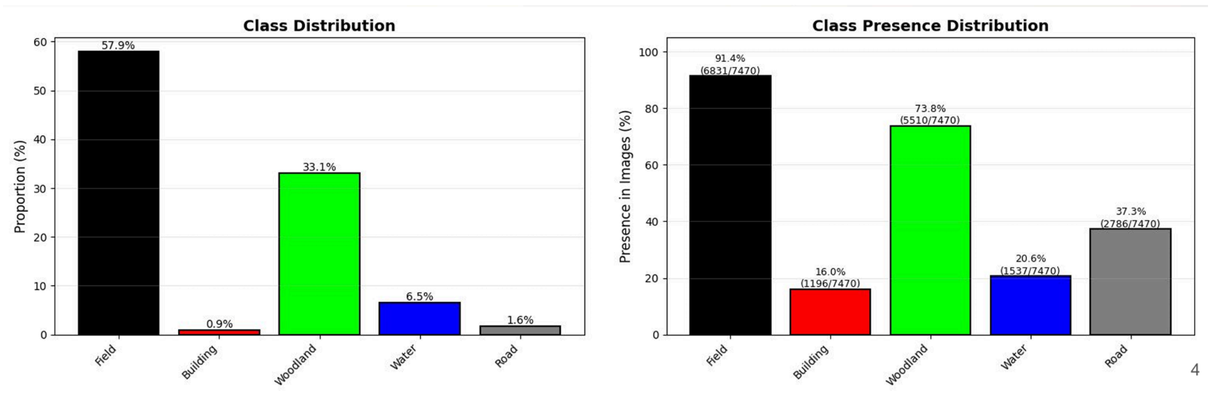

Figure 1: On the left, the class distribution by area across the entire dataset shows that the classes are highly imbalanced. On the right, the presence of classes per image (an image can contain multiple classes) is shown; the imbalance persists but is less pronounced.

II. Feature Choice and Extraction

Feature Choices

To successfully carry out the project, it was necessary to select and extract a set of features that would allow for discriminating between the different classes. For feature selection, the decision was made to extract as many features as possible in order to perform a brute-force exploration of the dataset. Indeed, the dataset contains a heterogeneous set of data and classes, making it difficult to know a priori which features would be the most discriminative. Therefore, the approach chosen to maximize the chances of obtaining good results was to extract a wide variety of features.

The features were extracted across five categories:

- Colors and spectrum: R, G, B, Hue, Saturation, Value
- Intensity and context: Grayscale and Gaussian blurs (e.g., σ = 2.5, σ = 5 …)
- Gradient / geometry: Gradient, Anisotropy
- Texture: Variance, Entropy, Local Binary Patterns (LBP)
- Indices: NDVI, Water (Specific to satellite segmentation)

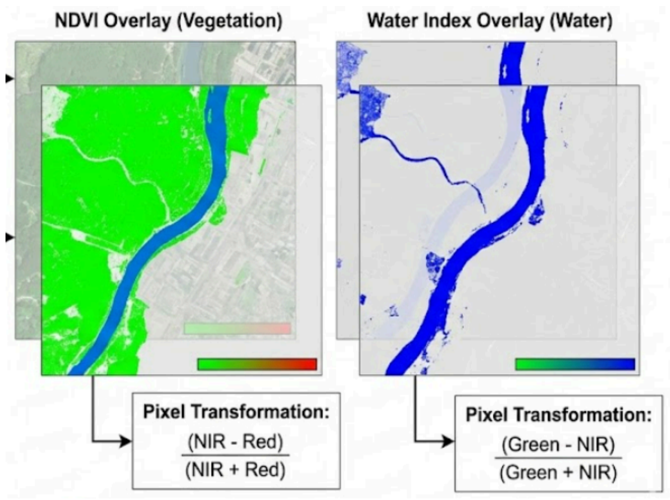
NDVI Overlay (Vegetation)
Water Index Overlay (Water)
Pixel Transformation:
(NIR - Red) / (NIR + Red)
Pixel Transformation:
(Green - NIR) / (Green + NIR)

Figure 2: The NDVI and Water features are calculated using the formulas presented above. These features are specifically designed for satellite image segmentation, in particular to detect vegetation and water, respectively, by comparing color channels and the near-infrared (NIR) channel.

Feature Extraction Pipeline

The data extraction pipeline provides a comprehensive framework for transforming raw satellite imagery into structured, machine-learning-ready datasets through a systematic multi-stage process. The pipeline starts with an input dataset of RGB satellite images (512, 512, 3) and their corresponding ground-truth masks, which classify pixels into multiple classes. The first stage performs pixel-wise feature extraction to capture diverse spatial and spectral characteristics. This includes color and spectral features that analyze RGB channel distributions and spectral signatures, texture features derived from local pixel patterns and spatial arrangements, gradient and geometric features that capture edges and shapes, and multi-scale contextual features that provide hierarchical spatial information around each pixel. After feature extraction, the pipeline applies feature normalization and dimensionality reduction. Min-Max normalization scales features to the [0-1] range, and Principal Component Analysis (PCA), learned exclusively on the training set, reduces dimensionality. The extracted features are then organized into feature tensors stored as NumPy (.npy) files with dimensions (512, 512, F), where F represents the total number of extracted features or the number of PCA components retained. The dataset structuring phase consolidates all components into a ready-to-use training dataset. This includes the feature tensor files, mask files in .png or .npy format, and a CSV metadata file that ensures complete traceability by listing unique identifiers for each sample along with file paths linking images to their corresponding feature tensors and ground-truth masks.

The pipeline is designed with modularity and flexibility in mind. New feature types can be added by simply including a new extraction function in the code. Users can select specific feature subsets for particular tasks, apply custom normalization schemes, or incorporate PCA transformations directly during extraction. This provides an adaptable framework suitable for diverse remote sensing and machine learning applications, while maintaining data integrity and reproducibility through comprehensive metadata tracking.

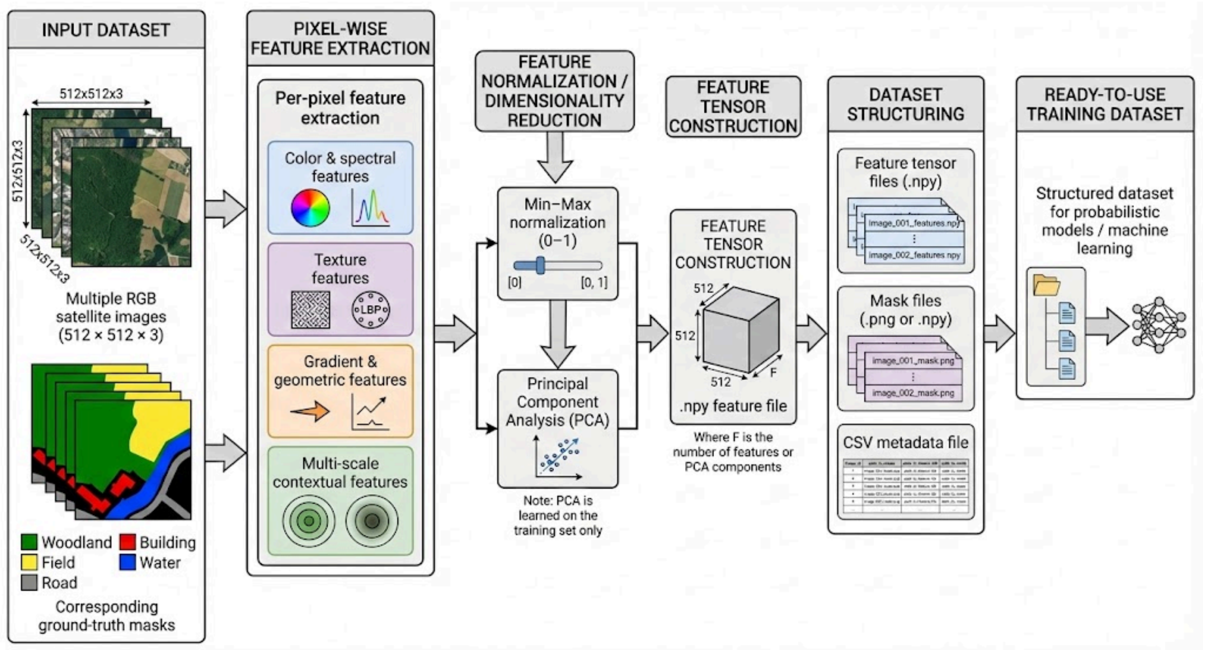
INPUT DATASET
512x512x3
Multiple RGB satellite images (512 × 512 × 3)
Woodland
Building
Field
Water
Road
Corresponding ground-truth masks
PIXEL-WISE FEATURE EXTRACTION
Per-pixel feature extraction
Color & spectral features
Texture features
LBP
Gradient & geometric features
Multi-scale contextual features
FEATURE NORMALIZATION / DIMENSIONALITY REDUCTION
Min–Max normalization (0–1)
[0]
[0, 1]
Principal Component Analysis (PCA)
Note: PCA is learned on the training set only
FEATURE TENSOR CONSTRUCTION
FEATURE TENSOR CONSTRUCTION
512
512
512
F
.npy feature file
Where F is the number of features or PCA components
DATASET STRUCTURING
Feature tensor files (.npy)
image_001_features.npy
image_002_features.npy
Mask files (.png or .npy)
image_001_mask.png
image_002_mask.png
CSV metadata file
READY-TO-USE TRAINING DATASET
Structured dataset for probabilistic models / machine learning

Figure 3: Overview of the entire data extraction pipeline. The images and masks are provided as input and converted into NumPy arrays of size (512, 512, F) and (512, 512), respectively, and stored in specified folders. After this step, the data is ready for training. □

III. Image segmentation techniques

Probabilistic histogram-based segmentation

The first segmentation approach implemented relies on a probabilistic method based on modeling class-specific feature distributions using multidimensional histograms. Each pixel is represented by a feature vector $x^F \in R^F$, where F denotes the total number of extracted features. For each class c, an empirical estimate of the conditional probability (P(x \mid c)) is learned from the training data by discretizing the feature space into a set of bins. The probability density is approximated by the relative frequency of feature vectors falling into each bin, according to:

$$P(x|c) = N_{c,\,bin(x)}/N_c$$

where $N_{c,\,bin(x)}$ is the number of samples of class c falling into the bin associated with vector x, and N_c is the total number of samples belonging to class c. The segmentation of a new image is then performed on a pixel-wise basis using a maximum likelihood Bayesian decision rule, assigning each pixel to the class that maximizes this probability:

$$c_{pred} = arg_c\, max\, P(x|c)$$

This formulation frames the segmentation task as a probabilistic classification problem based on non-parametric density estimation.

This approach is well suited to the problem at hand, as it enables the joint exploitation of a large and heterogeneous set of features without assuming any parametric form for the class distributions. In contrast to simple thresholding methods, which rely on deterministic rules applied to one or a few isolated features, histogram-based probabilistic modeling allows multiple feature dimensions to be combined while explicitly accounting for uncertainty. Thresholding techniques are typically highly sensitive to illumination changes, noise, and threshold selection, and they do not provide any measure of confidence. By contrast, the probabilistic framework provides explicit likelihood estimates for each class, enabling the identification of ambiguous or poorly separated regions. However, this method also suffers from notable limitations, including the curse of dimensionality, which makes reliable histogram estimation difficult in high-dimensional feature spaces, as well as a strong dependence on the choice of bin number and size, potentially leading to information loss or noisy estimates if the bias–variance trade-off is not properly managed.

Histogram-based segmentation pipeline

The histogram extraction pipeline was implemented as follows. In a first step, the program is provided with the set of features selected for histogram learning. Feature subsets can be selected by the user using feature identifiers (indices), and the program automatically performs slicing along the third dimension of the feature tensors, corresponding to the feature axis of size FFF. This operation directly selects the desired features. The corresponding segmentation mask is also provided as input.

All operations are fully vectorized to ensure efficient processing and learning. Histogram models are learned independently for each class, resulting in a total of N×FN \times FN×F histograms, where NNN denotes the number of classes and FFF the number of selected features. Each histogram corresponds to a specific class–feature pair and is discretized into BBB bins, where BBB is a user-defined parameter. After construction, the histograms are smoothed to obtain a more continuous and accurate representation, which improves robustness during the inference stage.

All learned histograms, which constitute the probabilistic model used for inference, are stored in a single compressed NumPy file in `.npz` format. During inference, this file is loaded to compute the maximum likelihood estimates for each pixel, allowing the system to directly produce an output segmentation mask encoded with the predicted classes.

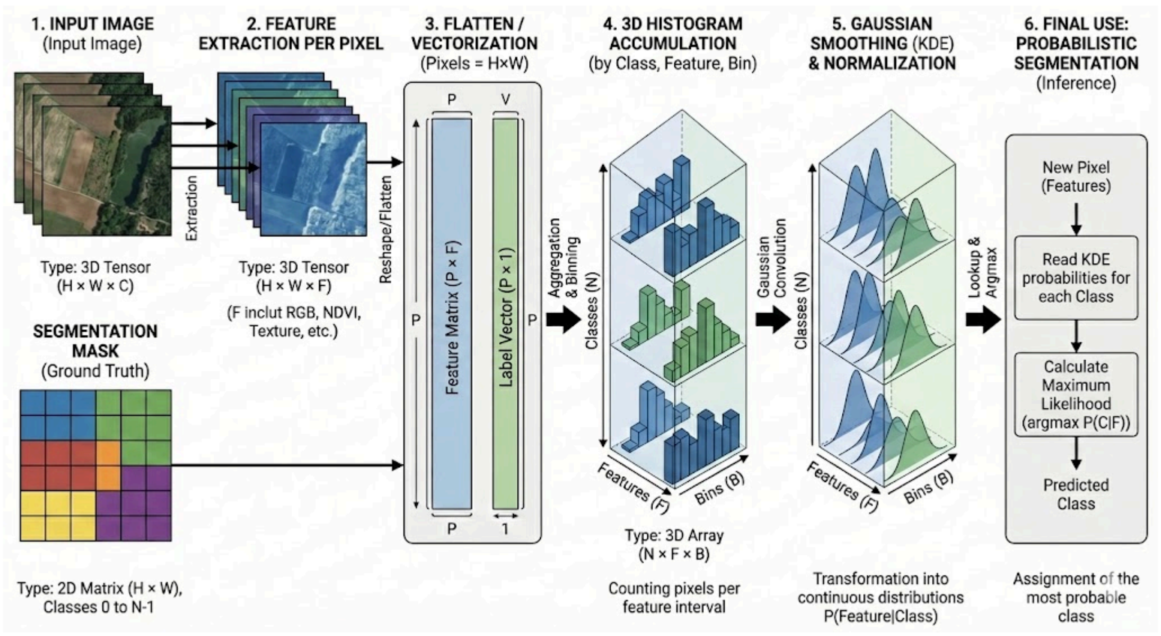
1. INPUT IMAGE
(Input Image)
Type: 3D Tensor
(H × W × C)
Extraction
SEGMENTATION MASK
(Ground Truth)
Type: 2D Matrix (H × W), Classes 0 to N-1
2. FEATURE EXTRACTION PER PIXEL
Type: 3D Tensor
(H × W × F)
(F inclut RGB, NDVI, Texture, etc.)
Reshape/Flatten
3. FLATTEN / VECTORIZATION
(Pixels = H×W)
P
V
Feature Matrix (P × F)
Label Vector (P × 1)
1
Aggregation & Binning
4. 3D HISTOGRAM ACCUMULATION
(by Class, Feature, Bin)
Classes (N)
Features (F)
Bins (B)
Type: 3D Array
(N × F × B)
Counting pixels per feature interval
Gaussian Convolution
5. GAUSSIAN SMOOTHING (KDE) & NORMALIZATION
Classes (N)
Features (F)
Bins (B)
Transformation into continuous distributions P(Feature|Class)
Lookup & Argmax
6. FINAL USE: PROBABILISTIC SEGMENTATION
(Inference)
New Pixel
(Features)
Read KDE probabilities for each Class
Calculate Maximum Likelihood
(argmax P(C|F))
Predicted Class
Assignment of the most probable class

Figure 4: Overview of the histogram learning pipeline up to the inference stage. It starts with the ingestion of the training data (feature matrix and segmentation mask) and ends with pixel-wise prediction using maximum likelihood formulas. □

Alternative approach: CNN-based segmentation using U-Net

As a second segmentation approach, a convolutional neural network based on the U-Net architecture was implemented. U-Net is a widely used CNN model for image segmentation, particularly well suited for pixel-wise prediction tasks. The model directly takes RGB images with three channels as input, represented as matrices of size (H,W,3)(H, W, 3)(H,W,3), and produces as output a one-hot encoded segmentation mask corresponding to the target classes. In the context of this project, the output is therefore a tensor of size (H,W,5)(H, W, 5)(H,W,5), reflecting the five land-cover classes. The one-hot encoded output can subsequently be converted into a single-class segmentation mask with class indices ranging from 0 to 4 by applying a pixel-wise argmax operation.

The U-Net architecture follows an encoder–decoder structure. The encoder progressively downsamples the input image through convolution and pooling operations, allowing the network to capture high-level semantic information. The decoder then upsamples the feature maps to recover spatial resolution and generate a dense segmentation map. Crucially, U-Net uses skip connections between corresponding encoder and decoder layers, enabling the model to combine fine-grained spatial details with high-level contextual information, which significantly improves segmentation accuracy. To reduce computational cost and training time, input images are sometimes downscaled to resolutions such as 256 × 256 or 128 × 128 pixels, at the expense of some spatial detail.

During training, the best-performing model checkpoints are continuously saved in order to ensure traceability and robustness, particularly in the event of premature interruption of the training process. The file *evaluation.json* stores the final evaluation metrics of the selected model, while *training_history* records the evolution of training and validation losses as well as accuracy values for each epoch. Finally, the file *config.json* contains all training parameters associated with the model. This configuration file is especially important during the inference phase, as it ensures that the same preprocessing steps applied during training are consistently reused when processing new input data.

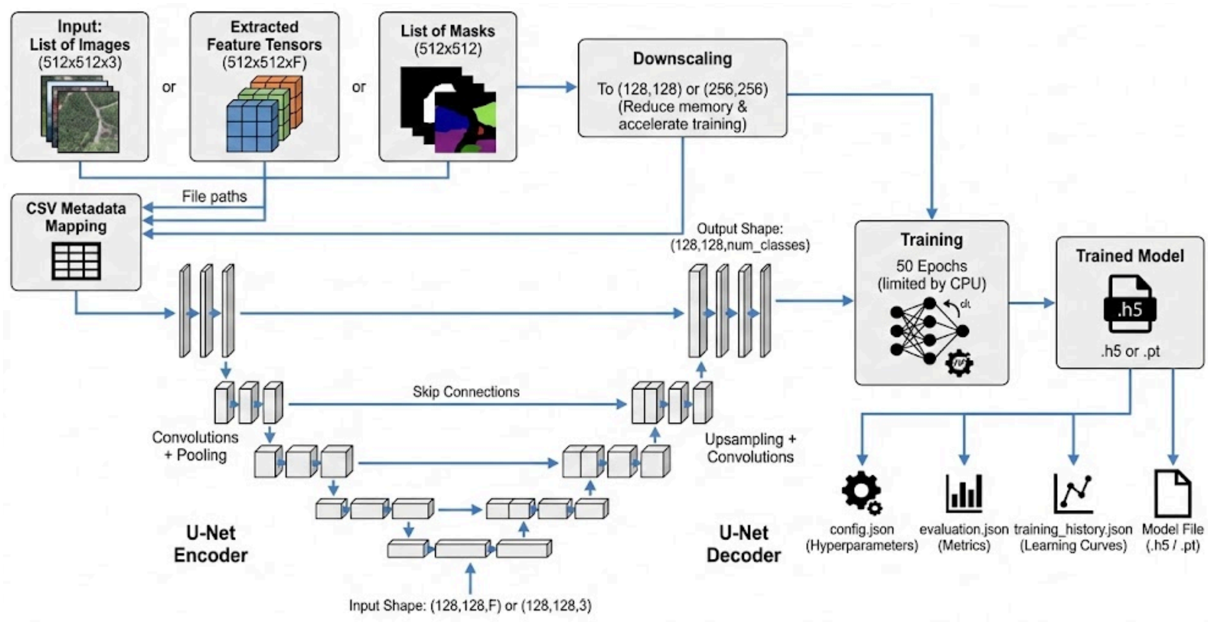

Figure 5: This figure shows the entire training pipeline of the U-Net model. In the top left, the data ingestion is shown (mask and feature and/or 3-channel RGB image). The images are then downscaled according to the user's choice. Once the training is complete, the model is saved and informational .json files are generated automatically. □

U-Net training evaluation

The evolution of the loss indicates effective learning with rapid convergence during the first 25 epochs (loss decreasing from 0.95 to 0.36), followed by a fine optimization phase down to approximately 0.18. The stepwise learning rate schedule (1e-3, 5e-4, 2.5e-4, then 1.25e-4) is well calibrated, allowing for an initial fast descent followed by a gradual refinement of the weights. The increasing gap between training loss and validation loss from epoch 25 onward indicates moderate overfitting, with the model progressively memorizing the training data at the expense of generalization. The validation loss, after decreasing to around 0.30, stagnates or slightly increases toward the end of training, confirming that the final epochs do not provide further meaningful improvement. The training process would have benefited from early stopping around epochs 35–40 or from additional regularization techniques (such as dropout or data augmentation) to better balance learning capacity and generalization. Despite this late-stage overfitting, the overall training quality remains satisfactory, with an acceptable final validation loss (~0.33) and stable convergence without divergence.

To further improve model training, it would have been beneficial to provide not only RGB channels as input, but also one or two highly specific and discriminative features for satellite image segmentation, such as NDVI or a water-related index. However, this choice comes at the cost of increased training time and was not feasible within the scope of this project due to limited computational resources, as training was performed exclusively on

CPU. The estimated training time is approximately 8 hours for 50 epochs, corresponding to roughly one full night of computation. Adding additional features would have increased the dimensionality of the input data and consequently extended the processing time beyond reasonable limits. Finally, incorporating an early stopping strategy during training could have helped mitigate some overfitting issues. Nevertheless, training for 50 epochs appears to be a reasonable compromise, given that the observed overfitting remains relatively moderate.

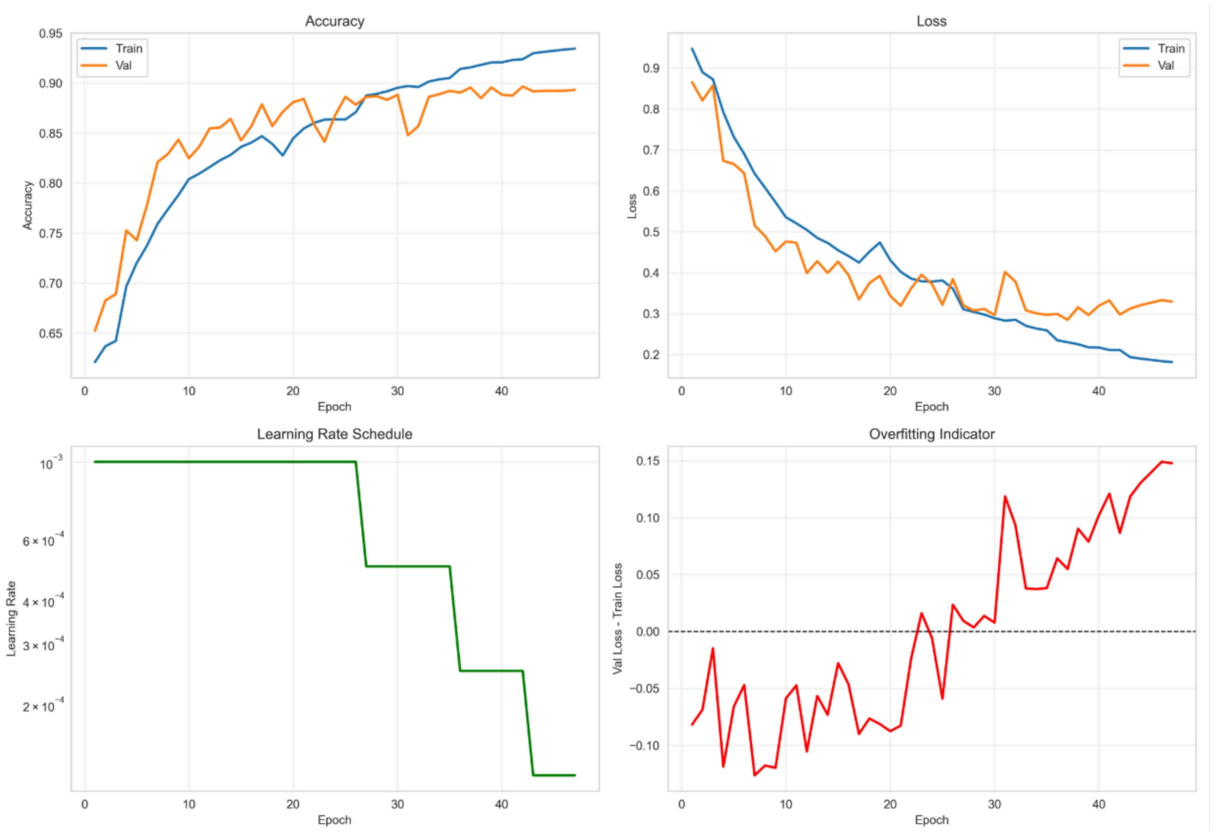

Figure 6: Overview of the training results: top left shows accuracy, top right shows loss for both training and validation. Bottom left displays the evolution of the learning rate per batch over the epochs. On the right, an overfitting indicator is shown (ratio of validation loss to training loss).

Post-Processing Strategy

Once the inference pipelines are developed, model performance can be further improved using a post-processing pipeline. This step aims to ensure that predictions are consistent with certain class-invariant properties. For example, roads are continuous and straight, so predicting a road that does not exhibit these characteristics is impossible. In our current model, however, the initial outputs often include fragmented road segments, interrupted either by occlusions such as clouds or shadows, or by intrinsic model errors. To address these issues, the pipeline applies an aggressive reconstruction strategy designed to restore continuity and preserve the linearity of road axes. At the core of this strategy is skeletonization, a morphological operation that iteratively erodes object contours until only a one-pixel-wide central axis remains. This process simplifies the complex geometry of roads into a topological graph, which serves as the basis for connecting nearby segments using extrapolation algorithms. The methodology prioritizes recall over precision, deliberately accepting the risk of creating artificial connections to ensure that no relevant routes are missed (increasing recall at the expense of precision).

Buildings, on the other hand, are subject to strict geometric constraints, unlike natural elements with irregular shapes. The post-processing pipeline applies regularization procedures to correct “organic” or imprecise contours generated by the neural network. Techniques such as oriented bounding box adjustments are used to restore the characteristic rectangular shape of buildings and reinforce their sharp corners. Beyond geometric correction, an expansion strategy slightly enlarges the detection area around each building. This approach also aims to maximize recall, ensuring that the full footprint of each structure is captured, even if it slightly overestimates bordering pixels. This decision is motivated by the importance of capturing complete building footprints rather than avoiding minor false positives along edges, given the potential impact on downstream analyses such as urban planning.

Finally, segmentation masks for natural classes—such as fields, water bodies, or forests—often exhibit speckled noise, known as “salt and pepper” noise. To improve spatial consistency, the pipeline applies basic morphological operations. Opening, which consists of erosion followed by dilation, removes isolated pixels and small bright artifacts. Closing, which combines dilation and erosion, fills small holes and cracks within homogeneous areas. By enforcing the dominance of the majority class within local neighborhoods, these operations effectively eliminate anomalous predictions that would break the geographical logic of continuous natural areas, producing a cleaner and more interpretable representation of landscapes.

After class-specific processing, residual conflicts inevitably arise where different classes overlap. The pipeline resolves these conflicts using a carefully designed class-priority hierarchy. For instance, road pixels take precedence over fields, while buildings override forested areas. This hierarchy ensures that the semantic meaning of overlapping elements is preserved according to logical and practical considerations. To finalize the segmentation map, a consistency rule is applied to handle unclassified pixels or those belonging to abnormal categories. These pixels are reassigned to the default majority class, typically fields, eliminating visual discontinuities or undefined regions. This final boundary smoothing ensures that the resulting map represents a complete and coherent partition of the territory.

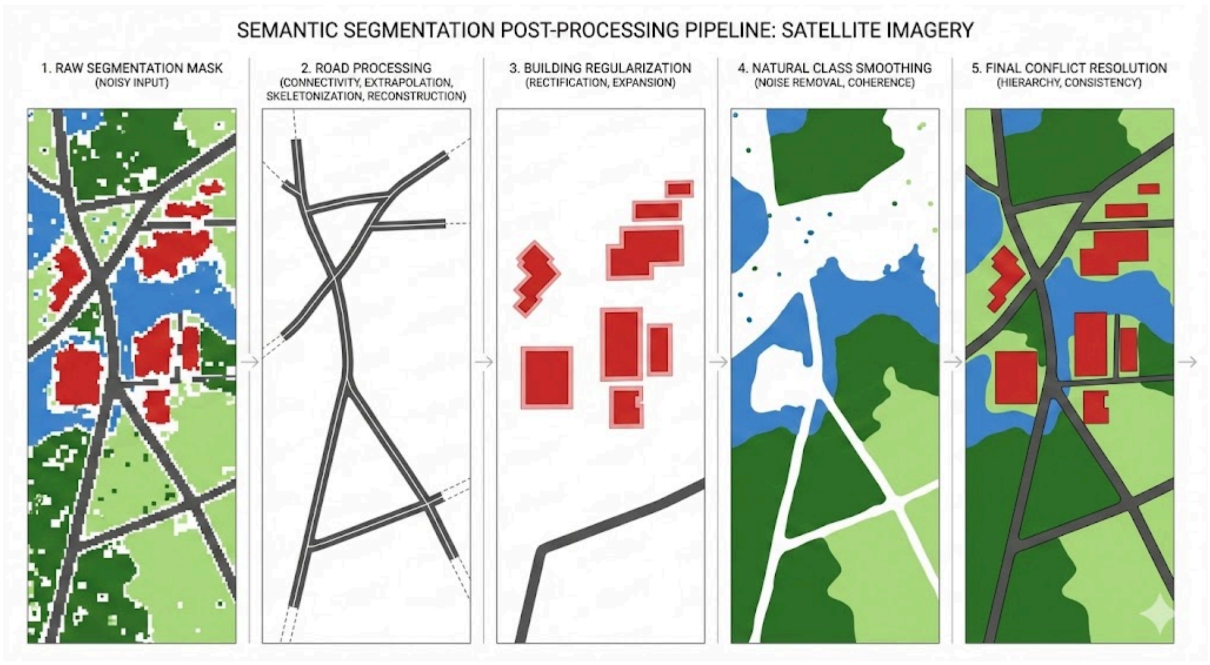
SEMANTIC SEGMENTATION POST-PROCESSING PIPELINE: SATELLITE IMAGERY
1. RAW SEGMENTATION MASK
(NOISY INPUT)
2. ROAD PROCESSING
(CONNECTIVITY, EXTRAPOLATION, SKELETONIZATION, RECONSTRUCTION)
3. BUILDING REGULARIZATION
(RECTIFICATION, EXPANSION)
4. NATURAL CLASS SMOOTHING
(NOISE REMOVAL, COHERENCE)
5. FINAL CONFLICT RESOLUTION
(HIERARCHY, CONSISTENCY)

Figure 7: Overview of the post-processing steps. *The initial image is noisy, with fragmented roads and shapeless buildings. Next, the roads are reconstructed and the buildings reshaped. Finally, noise in the natural classes is removed, and the boundaries between classes are smoothed.* □

IV. Results and Performances

Segmentation Technique Performance

The Dice coefficient (or F1-score) is the primary evaluation metric for this semantic segmentation task due to the extreme class imbalance in satellite images. Over 90% of the pixels belong to the dominant classes "Field" and "Woodland," while less than 10% are distributed among "Road," "Building," and "Water" (with buildings representing less than 1% of the data). Using overall accuracy would therefore be misleading and ineffective. A naïve model that classifies all pixels as "Field" or "Woodland" could easily achieve over 90% accuracy while completely failing to detect critical urban infrastructure. The Dice coefficient, computed as the harmonic mean of precision and recall, heavily penalizes false negatives and false positives, providing a balanced assessment of performance for each class individually, regardless of its representation in the dataset. This metric thus captures the model's true ability to accurately segment essential minority classes, such as buildings and roads, which are crucial for urban mapping and land-use planning applications.

The histogram-based and probabilistic distribution approach using 18 pixel-wise features shows dramatically insufficient performance, with catastrophic Dice scores for the minority classes: Building (0.064), Road (0.088), and Field (0.197), highlighting the inadequacy of this method for satellite image segmentation. Several factors explain this failure. First, pixel-wise analysis completely ignores spatial context and object geometry, whereas satellite image interpretation relies on spatial patterns, textures, and pixel neighborhood relationships. Second, classes like "Building" and "Road" exhibit high intra-class spectral variability (e.g., roofs of different colors, varied materials, shadows) while sharing similar spectral signatures with other classes (e.g., confusion between asphalt roads and dark roofs), making separation based on simple statistical feature distributions impossible, even with a large number of features. Third, the extreme class imbalance biases probabilistic distribution estimates toward the dominant classes, further marginalizing the underrepresented classes.

The U-Net architecture demonstrates substantial improvement, achieving Dice scores of 0.912 for Field, 0.915 for Woodland, 0.852 for Water, 0.694 for Building, and 0.528 for Road, outperforming the histogram-based approach by large margins (×10.8 for Building, ×6 for Road). This exceptional performance is explained by the inherent capabilities of deep convolutional networks: hierarchical feature extraction allows simultaneous capture of low-level characteristics (textures, edges) and high-level features (shapes, spatial structures), which are essential for discriminating complex objects. U-Net's encoder-decoder architecture, with skip connections between symmetrical layers, preserves fine spatial information while benefiting from a wide receptive field, enabling precise segmentation of building boundaries while integrating the surrounding urban context. Successive convolutions learn to recognize class-specific patterns (road alignment, regular building geometry, field homogeneity), while pooling and upsampling operations effectively handle scale variations. Although Road (0.528) and Building (0.694) remain more challenging to segment than natural classes, reflecting their geometric complexity and diversity in the dataset, U-Net nonetheless achieves practically usable performance, whereas the histogram-based approach fails completely.

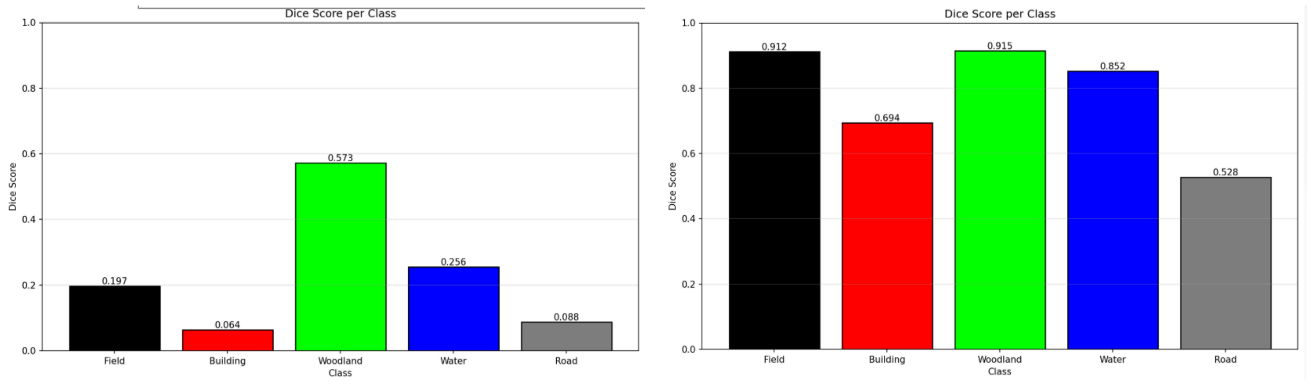

***Figure 8:** On the left, Dice scores by class for the histogram-based classification method. On the right, Dice scores by class for deep-learning classification using a U-Net architecture.*

Impact of Post-Processing on Performance

Post-processing has the expected effect on model performance, significantly improving recall for the Road and Building classes. Although precision decreases noticeably for these classes, the pixels relabeled as Building or Road are generally located near true pixels of these classes, indicating meaningful corrections. The normalized confusion matrices confirm this improvement: the main diagonal strengthens (Building increases from 0.64 to 0.78), reflecting more accurate classifications, while inter-class confusions decrease significantly. Post-processing notably reduces confusion between "Building" and "Field" (from 0.31 to 0.16) and between "Road" and "Field" (from 0.38 to 0.29). Even though post-processing primarily targets the Road and Building classes, the model's performance for Field, Water, and Woodland remains largely unchanged. These classes maintain high performance, with precision above 80% and recall above 85%, demonstrating the model's robustness for homogeneous classes.

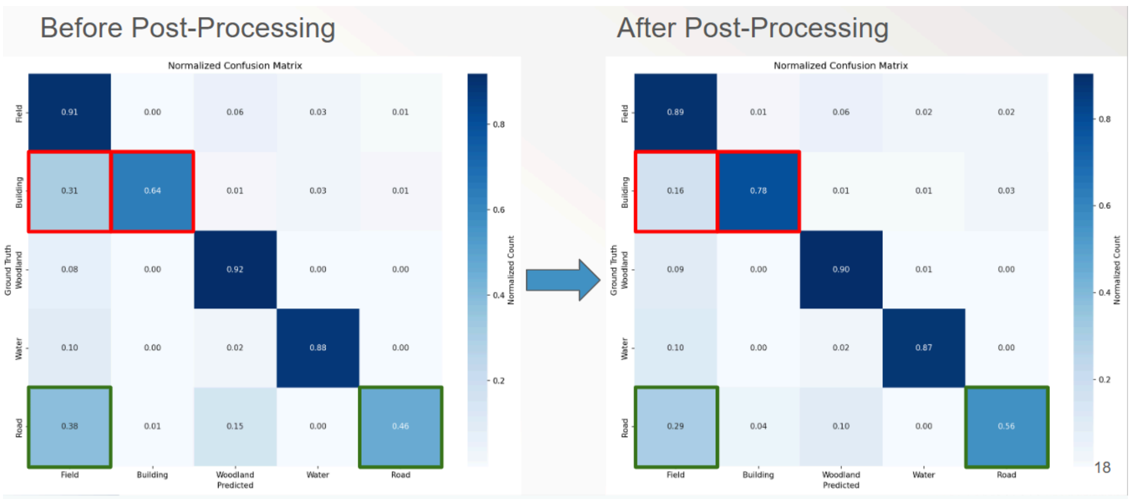

***Figure 9:** Normalized confusion matrices showing the reduction of inter-class classification errors after post-processing for the U-Net model.*

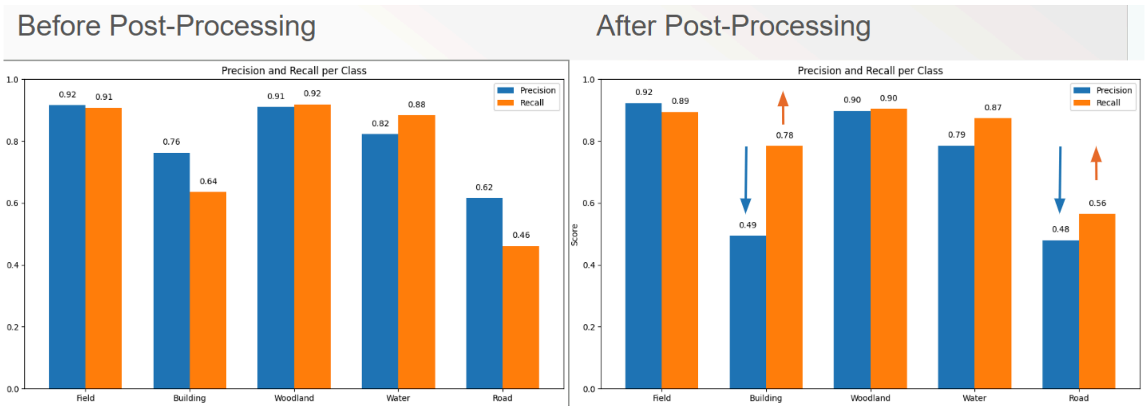

***Figure 10:** Precision and recall metrics by class before and after post-processing (notable improvement of recall for Buildings and Roads).*

Conclusion and Perspectives

This project enabled a systematic exploration of various approaches to semantic segmentation of satellite images, ranging from classical probabilistic methods to modern deep learning architectures. The comparative analysis of results clearly demonstrates the superiority of convolutional neural networks over traditional pixel-wise approaches, particularly for handling highly imbalanced classes and capturing the spatial context essential for interpreting images in our dataset. The U-Net architecture, combined with a specialized post-processing pipeline, achieved practically usable performance for automatic mapping applications, with Dice scores exceeding 0.90 for natural classes and remaining acceptable for urban infrastructures despite their strong under-representation. Beyond raw performance, this work highlights the importance of a rigorous methodological approach, integrating data preprocessing, relevant feature extraction, evaluation metrics adapted to class imbalance, and post-processing that leverages geometric and spatial constraints specific to each object category. The complete pipeline developed, from data ingestion to the generation of final segmentation masks, constitutes a comprehensive and modular processing chain, optimized to operate with limited computational resources.

Future improvements can be pursued along three main directions. First, the U-Net model's capabilities could be further optimized by integrating additional highly discriminative spectral features, such as NDVI, as extra input channels, exploring more recent architectures, and implementing more efficient training techniques for the dataset (e.g., early stopping) to reduce the overfitting observed at the end of training. Second, the post-processing pipeline could be enhanced with more sophisticated methods. Improving the robustness and computational efficiency of the code, as well as exploring semi-supervised or weakly supervised learning approaches, would also help reduce reliance on costly manual annotations and extend the system's applicability to diverse remote sensing and environmental monitoring contexts.

References

Dataset citation
Boguszewski, A., Batorski, D., Ziemba-Jankowska, N., Dziedzic, T., & Zambrzycka, A. (2021). *LandCover.ai: Dataset for Automatic Mapping of Buildings, Woodlands, Water and Roads From Aerial Imagery*. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (pp. 1102–1110).
This paper introduces the LandCover.ai dataset and provides dataset details and benchmark results.

Dataset official website
LandCover.ai. *Land Cover from Aerial Imagery* dataset.(Accessed [27-01-2026])
Available at: https://landcover.ai.linuxpolska.com/
Official dataset homepage with dataset description, download links, and license information.

Dataset distribution on Kaggle
LandCover.ai – Semantic Segmentation Dataset (Kaggle).
This version includes dataset files and an additional script for automatic train / validation / test splitting. (Accessed [27-01-2026]).
Information about the Kaggle distribution of the dataset.

Project repository
KolinMTG. (2026). IMG_Segmentation [GitHub repository].
Available at: https://github.com/KolinMTG/IMG_Segmentation
Contains the code and scripts developed for the semantic segmentation project using the LandCover.ai dataset.

□ *All images marked with this symbol in the figure captions were generated using highly detailed prompts with Nano Banana Pro, the latest version of the Nano Banana image generation model powered by Gemini 3 Pro Image.*