Road Segmentation

Antonin Hudry, Leonardo Tredici, Joana Pires
EPFL Lausanne, Switzerland

Abstract—**Manually labeling satellite maps is time-consuming, especially for large-scale datasets. This paper explores the use of machine learning to automate the process of road segmentation in satellite images. We used a dataset of 150 RGB satellite images containing various roads, split into a training set (100 images) and a test set (50 images). To improve model performance and diversity, we applied data augmentation techniques, such as rotation and flipping, resulting in a final training set of 1600 images. We tested several models, including logistic regression, a convolutional neural network (CNN), and a specialized U-Net architecture for image segmentation. Results indicate that the U-Net model outperformed other approaches in terms of segmentation quality, demonstrating the effectiveness of deep learning in automating the road segmentation process in satellite images. This work highlights the potential of using machine learning for large-scale urban planning applications and sets the foundation for future improvements in automated geographic feature recognition.**

I. Introduction

Satellite images are frequently used by urban planners for traffic management, city and suburban development, and map generation systems for GPS navigation. In these applications, roads, buildings, and green spaces are critical elements that must be accurately distinguished from each other. However, manually labeling satellite maps is a highly time-consuming process, particularly for large-scale images. One way to automate this process is through the use of machine learning in segmentation tasks. For this project, we will demonstrate the potential of this approach with a concrete example: a road segmentation task.

II. Models and methods

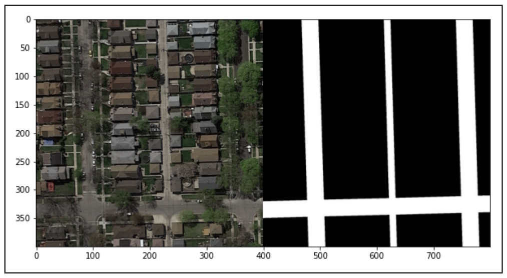


Fig. 1: Example of a satellite image (on the left) with its groundtruth (on the right) from training set

For this task, 150 satellite images in RGB colors with various road archetypes were used, split into a training set of 100 images and a test set of 50 images. The training images are associated with a labelled ground truth. We tried different approaches such as logistic regression, a convolutional neural network and finally, a UNet which is an architecture of convolutional neural network specialized for image segmentation.

A. Preprocessing

By applying two data augmentation functions sequentially, we significantly expanded the training dataset, thereby providing greater diversity for the machine learning model to learn from. Starting with 100 images, the final augmented dataset size reached 1600 images, and the same transformations were applied to the ground truth labels to maintain consistency.

The first augmentation function, rotates each image three times (90°, 180°, and 270°) using the NumPy function rot90, increasing the dataset size to 400 images.

The second function, further augments the dataset by flipping each image horizontally and vertically using the NumPy functions fliplr (horizontal flip) and flipud (vertical flip). This step expanded the dataset size to 1600 images.

B. Models

In total, we tested a baseline model and two other models.

To establish the baseline, we conducted a quick comparison of various basic models, including Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines, using a simple grid search to optimize parameters such as regularization strength, solver type or number of tree estimators. Based on this comparison, we selected a regularized logistic regression model as the baseline for the project. For this baseline model, each image is segmented into small batches of size 16×16, and features representing the colors and brightness are extracted. The model predicted a single probabilistic output for each batch, a close value to 1 indicates that the entire batch is classified as road, while a value of 0 indicates that it is classified as non-road. Batches are then concatenated to form the whole image probability map. We use the regularised logistic regression with regularisation parameter at 10, class weight is balanced, the solver is lbfgs and we used 100 maximum interation in the gridsearch. The number of iterations was selected by compromising the training loss and the computational time.

The first model tested is a Convolutional Neural Network (CNN) built in PyTorch. The architecture consists of four convolutional layers with 8, 16, 32, and 64 channels, respectively, and kernels of sizes 15, 7, 5, and 3, each with corresponding padding of (7, 3, 2, 1). Between each convolutional layer, batch normalization and ReLU activation are applied. A final convolutional output layer, followed by a sigmoid activation function, produces the output. The primary purpose of this model is to serve as a comparison between linear CNN architectures and more complex CNN architectures, such as the second model tested in this project, which is based on the U-Net design. The layers of this first model are set up to capture both global (large kernel, few channels) and local (small kernels, many channels) information in the images. The exact parameters were chosen based from a few empirical tests.

The second model tested is a U-Net-like neural network, again built in Pytorch. U-Net is a well-known network architecture characterized by a series of convolutional layers. The network initially descends through alternating convolutional layers and max-pooling operations, which reduce the spatial dimensions while capturing high-level features. It then ascends through alternating up-sampling and convolutional layers, progressively restoring the spatial dimensions. The architecture concludes with a final convolution layer to produce the final output. The ascending layers are symmetric to the descending layers, arranged in inverse order, and each symmetric pair is connected via a skip connection. These skip connections ensure that the model captures both local information from the descending layers and global information from the ascending layers (see Fig. 2).

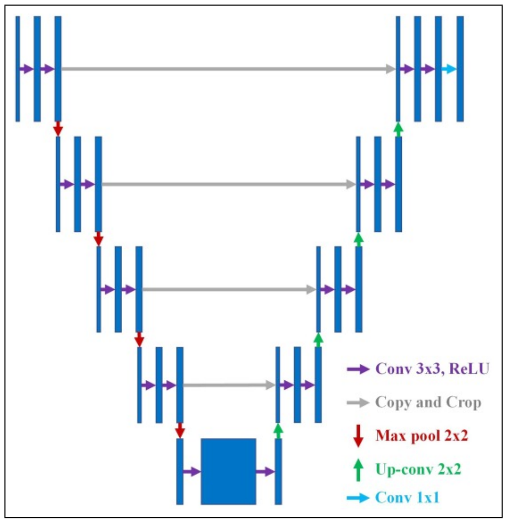


Fig. 2: Example of a U-Net, with descending layers on the left and ascending layers on the right (source: https://doi.org/10.1016/j.conbuildmat.2021.126265)

In our case, we defined the U-Net model with two sets of convolutional layers at each stage, using a 3×3 kernel with a padding of 1. SiLU was chosen as the activation function due to the depth of the network, to avoid the appearance of dead neurons, frequent with simple ReLUs. Max pooling with a 2×2 kernel was used in the descending path, while 2×2 upsampling was employed in the ascending path to maintain symmetry. The final layer consists of a 1×1 convolution followed by a sigmoid activation function to generate the output. During the descending path, spatial dimensions are reduced by a factor of 4 at each max-pooling step, while the number of feature maps is doubled after each double convolution, following the classical U-Net architecture.

C. Pipeline

Both models are trained using the Binary Cross Entropy loss criterion, which is a standard choice for binary classification tasks involving 1D and 2D data. The Adam optimizer is employed due to its robustness and effectiveness across diverse applications, making it a reliable choice given the project's time constraints. Similarly, StepLR is used for learning rate scheduling, offering a straightforward yet effective approach to adjusting the learning rate during training. The specific training parameters include 40 epochs, an initial learning rate of 0.002 for Adam, a step size of 5 epochs, and a decay factor of 0.8 for the scheduler. All other parameters are set to their default values as provided in the PyTorch implementation. These training settings were chosen based on extensive empirical testing and iterative trial and error. By monitoring the training loss, we ensured that the parameters allowed proper convergence for both models.

After completing the model training, we conduct a simple grid search on the threshold applied to the probability map output by the model. This threshold determines whether each pixel is classified as 1 (road) or 0 (non-road), allowing us to produce the final prediction. Since we do not have a specific preference for recall over precision in this application, we used the F1 score to score the threshold and to check the overall performance of our models.

D. UNet layers and base number of output channels

For U-Net, the number of layers and the base number of output channels need to be set prior to training the model. Increasing the number of layers reduced the loss, and similarly, increasing the base number of channels had the same effect (see Fig. 3 and 4). However, the number of channels and layers that can be added is constrained by the computational power and RAM of our computers. Based on testing, 16 base channels with 7 layers appeared to be the optimal limit for our setup. Adding more layers would result in overload of the memory. We can expect better results if we increase those parameters but we would need to use servers with more GPU power.

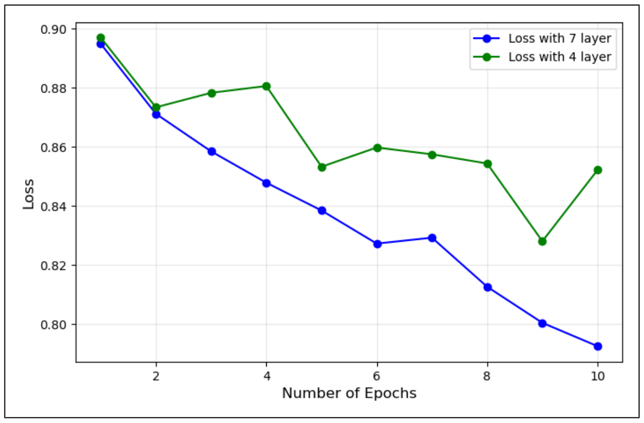


Fig. 3: Training loss over each epoch for 7 or 4 layers

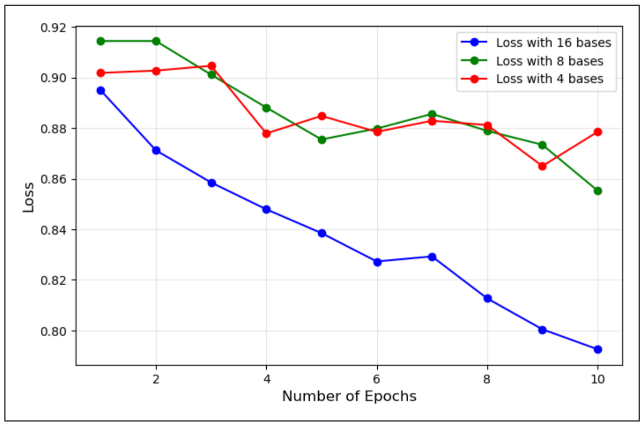


Fig. 4: Training loss over each epoch for different bases

E. Postprocessing

We attempted to improve our models further by incorporating a postprocessing step that refines the road segmentation output from the U-Net model. The postprocessing aims to address common issues we found in the output of both CNN and U-Net models, such as disconnected road segments, false positives, and small artifacts that may appear in the raw output. To achieve this, we applied several techniques designed to clean up and refine the segmentation map, leveraging useful functions found in the skimage, networkx, and scipy libraries.

Very small objects, which often correspond to irrelevant structures like noise or tiny outlier pixels, are removed using a function that eliminates components smaller than a specified size, using label function from skimage morphology. Additionally, small holes in the roads are filled, and the shape of the roads is smoothed and straightened through morphological closing operations from skimage, using square and diamond filters to help preserve the road's continuity.

We also address false positives, such as small, non-road regions like houses, by analyzing the shapes of the detected objects. Each object is examined based on its length vs. width ratio, which is the ratio of the major axis length to the minor axis length (computed with skimage). As we expect roads to be long and thin, objects with a low ratio are removed. However, to avoid removing complex road networks, such as orthogonal intersections, the removal of these objects is restricted by an area threshold. This ensures that only small, non-road components are filtered out while preserving larger, more intricate road structures.

To further refine the results, we added functionality to connect disconnected road segments. This step begins by skeletonizing each disconnected road region, via skimage, which reduces them to a thin, one-pixel-wide representation. A global graph is then built, via networkx, by treating each skeletonized pixel as a node and connecting nodes that are neighbors. For each region, we identify edge pixels (those with only one neighboring pixel) and check if they are close enough to other road segments. If they are, the segments are connected using a line creation algorithm from skimage, restoring continuity between nearby but disconnected roads. Once connected, the road regions are dilated to match the estimated width of the road they came from, ensuring smooth transitions between the connected segments.

After the training of our models was completed, we refined the postprocessing steps by fine-tuning the correction parameters using a grid search on the outputs of the trained models. These parameters included the size threshold for outliers to be removed, the size of the shape filters used to fill holes and gaps (as larger filters can overextend the corners of roads), and the maximum allowable distance between disconnected road segments for them to be connected. Other parameters, such as the length-to-width ratio and the maximum area for false positives, were not included in the grid search due to the already extensive parameter space being tuned. Instead, these were set to fixed values of 2.5 and 400, respectively, based on what seemed reasonable. Only the best-performing U-Net model output underwent postprocessing, as fine-tuning the hyperparameters of this process is time-intensive.

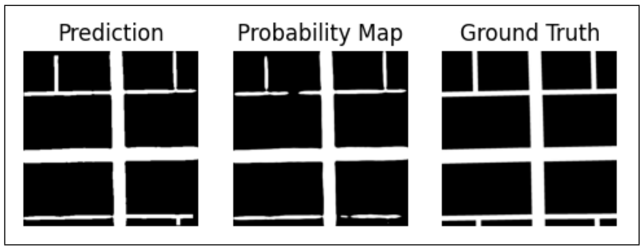


Fig. 5: Example of train image postprocess after U-Net output

III. Results

Our baseline model, a simple regularized logistic regression, achieved an F1 score of 0.357 and an accuracy of **0.512** using a threshold of **0.5**.

The CNN demonstrated significant improvement, achieving an F1 score of **0.525** and an accuracy of **0.815** with a threshold of **0.25**.

The U-Net models performed markedly better, achieving F1 scores above **0.7** (see Table I), even with a low number of layers and when trained on only 100 images instead of 1600. The highest scores were an F1 of **0.874** and an accuracy of **0.938**. Due to the U-Net's well-segregated output, predictions remained consistent for thresholds between 0.1 and 0.9 for best model; we adopted a threshold of **0.1**.

Finally, after applying postprocessing to the output of the best-performing U-Net, the model achieved an F1 score of **0.878** and maintained an accuracy of **0.938**. The optimized hyperparameters for the postprocessing were: outlier size of **1** (no outlier removal), a shape filter size of **1** (pratically no filtering), and a maximum connection distance of **60 pixels**.

IV. Discussion

The results indicate that the baseline model struggles significantly with the complexities of urban image segmentation. This approach often fails to differentiate between roads and rooftops, particularly when both feature similar colors or

Models	F1 Train	F1 Test	Accuracy Test
1. Regularized Logistic regression, lambda=10, t=0.5	0.380	0.357	0.512
2. CNN 1600 imgs, t=0.25	0.541	0.525	0.815
3. UNet 100 imgs, 16 base channels, 7 layers, t=0.45	0.744	0.710	0.859
4. UNet 1600 imgs, 16 base channels, 4 layers, t=0.45	0.823	0.810	0.909
5. UNet 1600 imgs, 16 base channels, 7 layers, t=0.1	0.884	0.874	0.931
6. UNet 1600 imgs, 16 base channels, 7 layers + road correction post processing	0.889	0.878	0.938

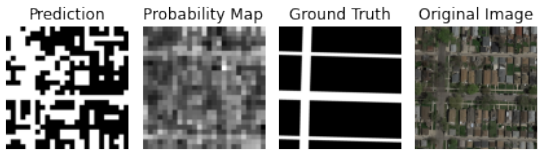
TABLE I: Best F1 and Accuracy scores of the different Models settings

textures, such as gray surfaces. The model's inability to distinguish these features results in a high number of false positives, leading to a low F1 score of **0.357**. Additionally, this model assigns a binary label (1 or 0) to each 16×16 patch, further exacerbates the issue by producing imprecise segmentations of roads. This lack of granularity makes it ill-suited for tasks requiring fine boundary detection and pixel-level accuracy (see Fig. 6a)(see Table 1 line 1).

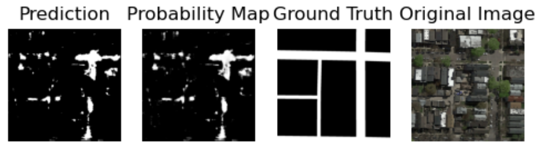
The simple CNN demonstrates an improvement over the baseline by better differentiating roads from other urban features, using convolutional layers. However the model still exhibits several limitations. One significant issue is its tendency to produce discontinuities in road predictions due to interference from small objects such as cars and trees. These occlusions disrupt the model's ability to maintain consistent road connectivity. Additionally, the CNN struggles with identifying small or narrow roads, often missing them entirely. (see Fig. 6b) The CNN uses a series of convolutional layers with increasing channels (8, 16, 32, 64) and progressively smaller kernel sizes (15, 7, 5, 3), the relatively large initial kernel sizes (e.g., 15×15) prioritize capturing global features over fine-grained details. As a result, the model is less effective at detecting small-scale structures. (see Table 1 line 2)

The U-Net model outperformed all others, demonstrating exceptional ability to capture small structures and fine details. When trained with only 100 images, the model produced poorly defined roads due to limited generalization (see Fig. 6c). However, increasing the training set to 1600 images significantly improved segmentation quality, resulting in sharper probability maps and more accurate predictions (see Table 1, lines 3 and 5).

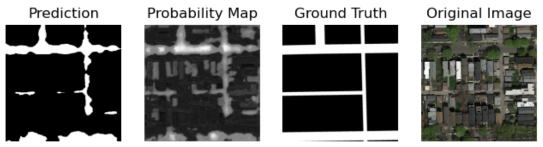
The postprocessing step further enhanced the results by connecting fragmented roads and ensuring continuity in the final output (see Fig. 6d). The optimized postprocessing hyperparameters highlight the U-Net's strong performance, as no outlier removal or closing operation was necessary. Despite this, the model still missed some road connections, which were partially corrected by the postprocessing step. This demonstrates the effectiveness of the U-Net, particularly when combined with sufficient training data and postprocessing (see Table 1, line 6).



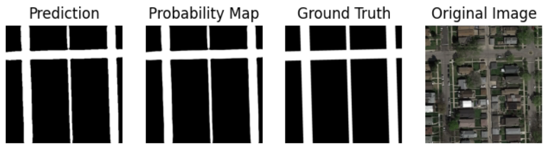
(a) Logistic regression trained on 1600



(b) CNN trained on 1600 images.



(c) Small UNet model trained on 100 images.



(d) Large UNet model trained on 1600 images.

Fig. 6: A combined figure with two subfigures stacked vertically and enclosed in a box.

V. Summary

The UNet model emerged as the best-performing approach for the road segmentation task, excelling at capturing small structures and intricate details. Its architecture, designed to extract features at multiple scales while preserving spatial resolution, makes it particularly well-suited for this type of segmentation problem.

Our results show that the size of the training set plays a critical role in the model's performance. When trained on only 100 images, the UNet struggles to delineate roads clearly, leading to less accurate probability maps and poorly defined road segments (see Fig. 6c).

In contrast, training the UNet on a substantially larger dataset of 1600 images results in significant improvements. Furthermore, the postprocessing step further enhances the model's output by improving road connectivity and correcting fragmented segments (see Fig. 6d).

Overall, the UNet model's ability to learn from augmented datasets and its architectural design make it an effective and usable approach for accurate and detailed road segmentation.

VI. Ethical risks

In the context of a project focused on identifying roads from satellite images, one of the key ethical risks to consider is bias in the model due to unrepresentative training data, which could result in inaccurate representation of roads, particularly in underrepresented areas. The bias risk occurs when the satellite images and their groundtruth labels used to train the model are not representative of all geographic regions. For example, if the satellite images primarily come from well-developed urban areas, the model may become more proficient at identifying roads in those areas while performing poorly in rural, remote, or less-developed regions. This can lead to significant disparities in the accuracy of road extraction across different regions.

The primary stakeholders impacted by this risk are government officials because inaccurate road extraction can hinder infrastructure planning particularly in underdeveloped areas. In addition, marginalized communities in underrepresented regions may face negative consequences if road data is misclassified, leading to delays for example in infrastructure improvement. Inaccurate road mapping can hinder the delivery of essential services like emergency responses as well as leading to decisions that are based on misleading data, potentially worsening inequalities between urban and rural regions.

The likelihood of this risk for our model is also significant, as the datasets provided seem skewed toward urban regions. We also reviewed potential sources of data bias in satellite imagery, particularly concerning geographic and socioeconomic factors (https://doi.org/10.1038/s41598-024-74150-9). We did not reduce the bias. This is because high-quality satellite imagery is often proprietary, and obtaining a balanced dataset that includes underrepresented regions proved challenging.