Improving Road Segmentation on Small Datasets with Post-Processing and Data Augmentation

Mounir Raki, Théo Ducrey, Marchon Xavier
Group: Highway Hackers
Computational Intelligence Lab, ETH Zürich, Switzerland

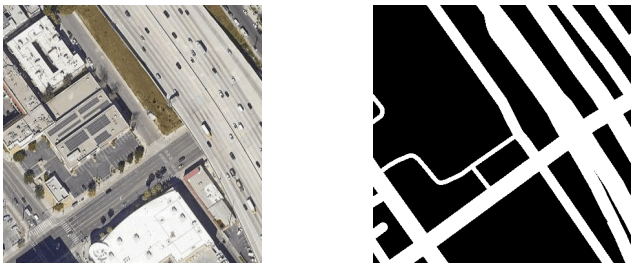
***Abstract*—We present an approach to automating the extraction of mapping information from aerial images using a transformer-based model. We address the challenges posed by the limited dataset and the underlying data imbalances. We extensively explored post-processing methods to refine model's outputs, focusing on enhancing road connectivity. Our results demonstrate the effectiveness of transformer-based models in road segmentation tasks and highlight the potential of automatic post-processing while demonstrating the difficulties posed by manual post-processing.**

I. Introduction

With millions of people around the world relying on map services for daily navigation, the demand for precise maps that enable accurate itinerary predictions is exceedingly high. Our rapidly evolving world presents significant challenges to the mapping process, making the transition from manual to automated mapping essential. Additionally, climate change has increased the frequency of natural disasters, which can drastically alter landscapes and thus affect navigable paths. Over the past decade, there has been a swift development of efficient models for image segmentation tasks. In this paper, we build upon existing methods to segment aerial images into road and non-road patches. Our model aims to fully automate the extraction of mapping information from aerial images, minimizing human intervention and achieving the lowest possible prediction error. This project was conducted as part of the Computational Intelligence Laboratory course at ETH Zurich.

A. Task and dataset

The training dataset comprises 144 RGB images, each with dimensions of 400 x 400 pixels. Each image is paired with a corresponding binary mask where a pixel value of 1 indicates a road pixel, and a pixel value of 0 indicates a non-road pixel. The evaluation task involves classifying 16x16 pixel patches within these images. A patch is classified as a road patch if more than 25% of its pixels are road pixels. Figure 1 illustrates an example from the training dataset. The evaluation metric available on Kaggle is the accuracy of our model on a larger test set measured by the F1 score. This metric is necessary because the training dataset exhibits an intrinsic imbalance, with approximately 75% of the pixels being background (non-road) pixels.



(a) Training image (b) Training groundtruth

Figure 1: Training dataset example

II. Models and Methods

A. Data augmentation

Given the data-intensive nature of transformers, it is evident that our dataset alone will not be sufficient for effective training. Moreover, powerful models are more prone to overfitting when trained on a limited amount of data. To address these issues, we decided to augment our dataset. Intuitively, roads in aerial images can appear in any orientation, so our first augmentation strategy was to apply random rotations along with horizontal and vertical flips. Additionally, to enhance road connectivity, we incorporated random erasing, following work in [1] on random erasing for data augmentation. We also considered color alterations but were uncertain whether modifying colors would eliminate crucial features, as roads typically exhibit a specific subset of the color spectrum. Furthermore, we experimented with random cropping but ultimately decided against it, as most rotations inherently provide some form of cropping. In the following sections, we will detail each of the augmentation techniques employed.

Rotational augmentation was achieved by first rotating the image and then cropping and resizing it to the largest possible size without introducing black corners, as illustrated in Figure 2. Additionally, we enhanced our rotation module by including the ability to perform horizontal and vertical flips for each image, thereby covering the full spectrum of rotations.

Figure 2: Rotations with cropping to avoid black corners

Random erasing augmentation was performed by randomly cropping small rectangles from the training images and replacing them with random values. This technique aims to encourage the model to learn how to use the surrounding road context to infer road continuity through the masked parts of the image. Figure 3 presents an example of random erasing.

All our augmentations were implemented in a stochastic manner, meaning each was assigned a probability of being applied when pulling an image from the dataset. This approach allowed us to perform data augmentation by simply making several passes across our training dataset. We chose to make N=5 and N=10 passes across our dataset, which resulted in approximately 700 and 1400 augmented images, respectively. We did not increase the number of passes further to avoid the risk of overfitting

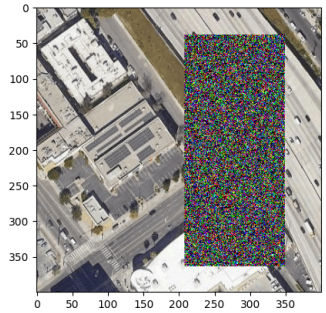


Figure 3: Random erasing of parts of the image

and to manage training time duration. We limited ourselves to these transformations for the same reasons, but many others could be implemented like noise perturbations, shearing and brightness changes.

B. Models

We conducted our experiments using SegFormer[2], a transformer-based model built on an encoder-decoder architecture. This choice allowed us to work with much larger receptive fields than CNN-based architectures such as U-Net for road segmentation. The encoder is a transformer that divides images into patches for its self-attention mechanism, while the decoder is a simple Multi-Layer Perceptron (MLP) that merges the transformer's outputs into a final output mask. Following the results reported in the same paper, we opted to use the parameters of the best-performing model, MiT-B5, for our experiments. Due to the substantial time, computational resources, and data requirements, we could not fully train the transformer on a custom dataset. Instead, we fine-tuned only the decoder part while utilizing the pretrained weights of the encoder provided by the paper's authors. For the training setup, we largely followed the procedure described in [2] and adjusted some hyperparameters. We employed the AdamW optimizer, which is essentially the Adam algorithm with weight decay regularization. We experimented with both a fixed learning rate and a "polynomial" learning rate scheduler, setting its base value to 0.00006.

We trained the model using three different data splits

- An 80/20 train/validation split, having 115 images to train on and 29 images to perform validation
- A 75/25 train/validation split, having 108 images to train on and 36 images to perform validation
- A 67/33 train/validation split, having 96 images to train on and 48 images to perform validation

Additionally, we implemented an early-stopping mechanism as a form of regularization, with a delay of 10 epochs before stopping and a hard limit of 100 epochs.

C. Data imbalance

Loss functions are an important part of machine learning models and are used to measure how well a model is performing. In the context of image segmentation, the two commonly used loss functions are the binary cross entropy loss and the soft dice loss, which are the ones we experimented with.
The binary cross-entropy is defined as follows:

$$l(x, y) = -\sum_{k=1}^{n} y_k * \log(\sigma(x_k)) + (1 - y_k) \log(1 - \sigma(x_k)) \tag{1}$$

where n is the number of samples in a batch, k is the sample index in a batch, x_k is the logits output related to input sample k, y_k is the ground truth mask for input sample k, and $\sigma(\cdot)$ denotes the sigmoid function mapping logits to the set $[0; 1]$.
The soft dice loss is defined as follows:

$$l(x, y) = 1 - \frac{1}{n} \sum_{k=1}^{n} 2 \frac{|\sigma(x_k) \cap y_k|}{|\sigma(x_k)| + |y_k|} \tag{2}$$

where x_k is the logits output of the model for input sample k, y_k is the mask ground truth for sample k, $\sigma(\cdot)$ is the sigmoid function mapping logits to the set $[0; 1]$ and $|\cdot|$ denotes the number of pixels belonging to the road class. The intersection operation is used here to count the number of correctly classified road pixels in the output of the model, using the ground truth mask as reference. The soft dice loss by definition mitigates the problem of class imbalance between the road class and the background, since it focuses on the class of interest (in our case here the road class).

D. K-fold cross validation

To enhance our model's performance on our very small dataset, we experimented with K-fold cross-validation. We tested values of $k \in \{3, 4, 5\}$, which resulted in the same training/validation splits as described in the Models section. We chose not to increase the number of folds beyond 5 to ensure enough samples for validation, nor decrease it below 3 to maintain a sufficient number of training samples per fold.

E. Manual post-processing

After examining some examples of our model's outputs. We noted that some post-processing could be useful for certain outputs (see Figure 4) to

1) Remove some hallucinated patches of roads
2) Correct the shape of roads to make them continuous
3) Connect fragmented road pieces that are close to one another.

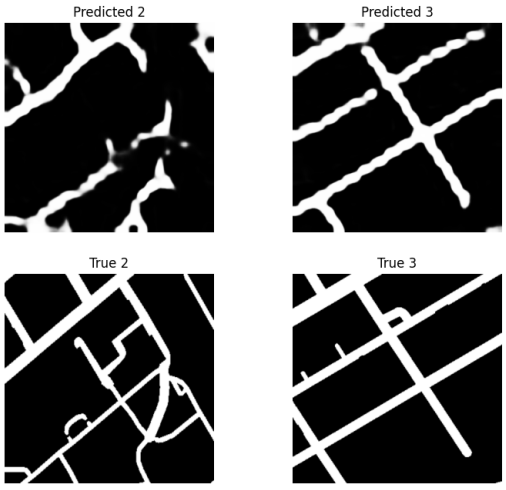


Figure 4: Examples that would benefit from some post-processing

As a first step, we experimented with different manual post-processing techniques. Each function can be executed at a downscaled resolution to reduce computational costs.

1) Removing patches unconnected to borders: Upon examining the provided images, we observed that all roads are connected to the border in some way, allowing a physical car to enter and exit the network. To refine the model's predictions, we employed this function to filter

out road-predicted pixels that are not directly connected to the image border through other predicted pixels. The user can specify the distance at which pixels are considered connected and the threshold at which a pixel value is considered a road (with a default value of 0). This method, implemented in PyTorch, executes quickly but does not significantly improve predictions.

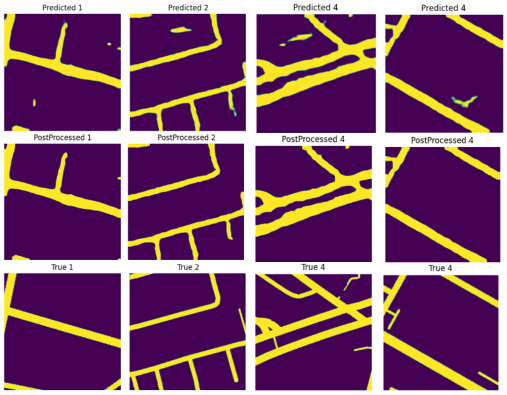


Figure 5: Pre- and post-masking pixel region that are not in contact with pixels touching the borders of the frames (radius=3)

2) Connect fragmented roads: Building upon the previous function, we leverage the requirement that roads must be interconnected to allow a car to travel between them. We group pixels into subnetworks of connected ones and then connect each subnetwork by adding the shortest possible connections between subgroups. Users can specify a maximum distance between groups for connections, as creating a road between groups that are too far apart may be counterproductive. Additionally, users can filter out small groups, assuming that the network may incorrectly predict small regions of aggregated pixels. As with the previous function, users can specify the pixel value threshold to be considered a road. This function operates on a binary approach, where 1 represents a road and 0 represents nothing.

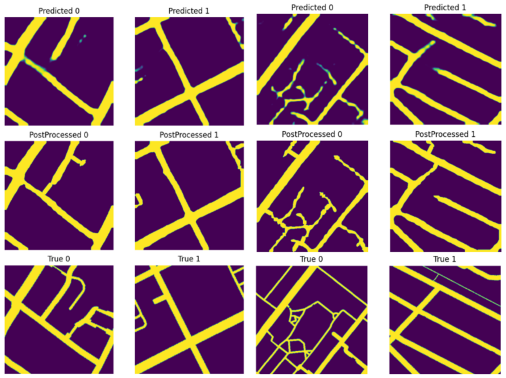


Figure 6: Pre- and post- adding connection between every region of pixels (dist=70, fat=6)

3) Connecting close pixels: This method aims to fill gaps in roads. It examines road pixels near each other and computes the shortest path between them, adding these paths to the predicted road pixels. The goal is to assume that two different roads are at least a certain distance apart and that two close predictions are part of the same road, thereby allowing the gap between them to be filled. However, this method often performs poorly, tending to remove gaps and smooth out corners between perpendicular roads (Figure 7, circled in red). It may require further tweaking to apply only to certain regions. Nevertheless, if these regions were known in advance, there would be no need to compute paths between close pixels.

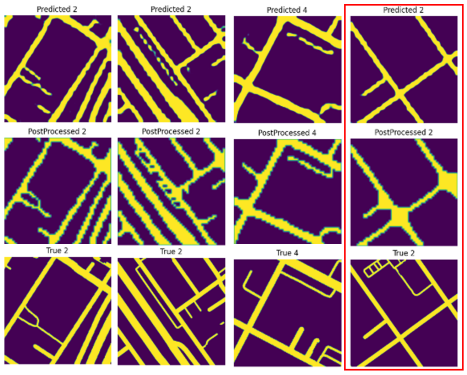


Figure 7: Pre- and post- ensure all close apart pixels are connected in the shortest way possible connect_all_close_pixels(dist=3 (red: 6), downscale=8)

F. Automatic post-processing

Given the limited success of manual post-processing approaches in improving our model's outputs, we explored an alternative method. Our objective was to train a refinement model on corrupted ground truth images using iterative application of random erasing to learn road connectivity in fragmented road contexts. We opted to train a classical U-Net architecture [3] as it should be capable to leverage local and global information well across the image. We pretrained this model on images from the DeepGlobe Road Extraction Dataset, which we corrupted using random erasing with small patches and random erasing with larger patches. Unlike previous approaches, we filled these patches with the value 0 instead of a random value to mimic our SegFormer model's failure to detect roads. The goal of this pretraining was to obtain a model that could be fine-tuned on the outputs of our SegFormer model to achieve the same three objectives as manual post-processing.

III. Results

A. Data augmentation results

The following results were obtained by training our model with 5-fold cross-validation, a fixed learning rate, and DiceLoss as the loss function. We empirically demonstrate that color jitter can slightly worsen predictions, while random erasing slightly improves them. Additionally, we observe that data augmentation is beneficial initially, but increasing the number of augmentations does not lead to further significant improvements.

	Scores			
	∅	R	R + CJ	R + RE + CJ
N = 0	0.91166	-	-	-
N = 5	-	0.92262	0.92242	0.92324
N = 10	-	-	-	0.92402

Table I: N : Number of passes through the dataset, R : rotations, RE : Random Erasing, CJ : Color Jitter. Scores from Kaggle

B. Losses and Cross-Validation results

Our results demonstrate that DiceLoss consistently performs better than BCELoss. Additionally, employing the k-fold cross-validation method consistently yields better performance. We also noticed that using a polynomial learning rate scheduler to vary the learning rate did help very slightly for the DiceLoss, but the differences are negligible. This could be due to the fact that our implementation of the DiceLoss leads to a pretty smooth objective function similar to the BCELoss.

Scores		
model	BCELoss	DiceLoss
Vanilla SegFormer	0.90044	0.90651
SegFormer with data augmentation	0.9118	0.9152
SegFormer with data augmentation and K-fold	0.91971	0.92324
SegFormer with data augmentation, K-fold and LR scheduler	0.91993	0.92182

Table II: Comparison between DiceLoss and BCELoss (Scores from Kaggle)

We observed that our model struggled with accurately predicting very small roads. To address this, we experimented with a loss function that assigns equal importance to the classification of small roads and larger ones. This was achieved by computing the loss on the prediction gradient, which, in our case, represents the borders of the roads. Specifically, we calculated the DiceLoss on the gradients of both the ground truth and the predicted segmentations. However, during training, this approach frequently resulted in vanishing gradients without leading to significant performance improvements. We concluded that small roads were challenging to extract from the data due to their high occlusion rates and their nature as narrow paths that often lack the main characteristics of larger roads.

C. Post-processing results

1) Manual Post-processing results: This section presents the results of various manual post-processing methods. As previously mentioned, these methods did not significantly improve the outcomes compared to the initial model results. While they may provide localized improvements, they generally cause more issues than they resolve. For example, a method might correctly connect two roads but lack sufficient information to distinguish between an incomplete road and a building misidentified as a road. Similarly, algorithms designed to add new roads require additional data not available from the basic model output.

Post-processing method	Score
Raw model baseline	0.92324
Mask connected through border (radius=3)	0.91613
Connect road (dist=70, fat=6)	0.91508
Connect all close pixels (dist=3, downscale=8)	0.24328

Table III: Manual Post-processing Kaggle scores

2) Automatic post-processing: We tested our best SegFormer models with three different U-Net refinement models. The first one was initialized with random values and fully trained from scratch using the SegFormer outputs as data and the ground truth as the mask. The other two were pretrained with random erasing using small occlusions and large occlusions, respectively. Surprisingly, our hypothesis that random erasing would make roads stand out did not prove to be correct. In reality, the best-performing refinement model was the one that was not pre-trained for the task.

Automatic post-processing method	Score
Raw model baseline	0.92324
Refinement UNet No Pretraining	**0.92486**
Refinement UNet Small RE Pretraining	0.92102
Refinement UNet Large RE Pretraining	0.92411

Table IV: Automatic Post-processing Kaggle scores

Once again, our results indicate that post-processing does not significantly improve our model's test scores. However, we argue that this work could still be valuable and may be enhanced with further research. Upon examining the final outputs, we discovered that our refinement model could indeed reinforce road borders and connect road fragments together when the distance between them was not too large. The refinement model is particularly effective at transforming very small and narrow roads into well-defined roads. However, it struggles when a significant patch of non-road pixels separates a road into two segments. Figure 8 visually highlights the gains provided by our refinement model, although these improvements are not significant in terms of score metrics.

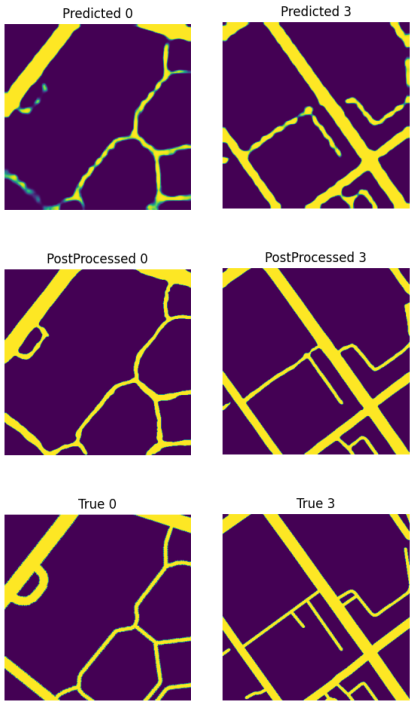


Figure 8: Automatic post processing visual gains

IV. Discussion

Our work highlights the effectiveness of transformer-based models in road segmentation tasks. We demonstrate how to effectively address data imbalances and challenges posed by small dataset sizes in this context. Contrary to our expectations, manual post-processing proved difficult to implement effectively. However, automatic post-processing shows promising potential. A clear future direction could be the specialization of post-processing models to specific tasks, such as learning road orientation and enhancing road connectivity, as well as the development of specific architectures that enhance road connectivity in the post-processing setting. As a final discussion point, it remains clear to us that post-processing can only be effective when the underlying model performs well. Therefore, improving the primary model may remain the most important focus for future advancements.

V. Summary

In this study, we developed and evaluated a transformer-based model for road segmentation in aerial images. Our approach includes data augmentation, cross-validation, loss function optimization and extensive post-processing techniques. While manual processing techniques did not significantly improve model performance, automatic post-processing using a refinement U-Net model demonstrated potential for improving road connectivity and refining predictions.

References

[1] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," 2017. [Online]. Available: https://arxiv.org/abs/1708.04896

[2] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 12 077–12 090. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/64f1f27bf1b4ec22924fd0acb550c235-Paper.pdf

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597