Road Segmentation from Satellite Images

Andri Horat, Matthias Karst, Nadine Frischknecht, Sarina Müller
Group: Helium, Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In this project, we tackle the problem of road segmentation from satellite images using machine learning. Given a small dataset, we employ various strategies to improve generalization and prevent overfitting. We explore a novel data augmentation method that adds Perlin Noise in HSV Color Space, existing augmentation methods such as ColorJitter and Affine Warping, acquire additional data using the Google Maps API, and use rich labels to provide more detailed information to the model. We compare different network models, with the Feature Pyramid Network (FPN) using an Efficientnet-b7 backbone performing the best. We further enhance the model's performance through Test-Time-Augmentation (TTA) and fine-tuning. Our best model achieves a Kaggle score of 0.93306.**

I. Introduction

The ability to accurately identify roads from satellite images can provide valuable insights and support decision-making processes in various fields, such as urban planning, transportation, and environmental analysis. In this report, we focus on the task of road segmentation using a limited training set containing 144 satellite images, each along with ground truth masks. Our goal is to determine whether a given 16x16 patch represents a road or not, based on the classification of its pixels. A patch is considered a road patch if at least 25% of its pixels are labeled as road pixels.

To overcome the limitations posed by the small dataset, we explore different techniques for data augmentation and acquisition. Additionally, we investigate various neural network models to find the most suitable architecture for the road segmentation task. Furthermore, we delve into post-processing methods to refine our predictions and achieve optimal results.

II. Pre-processing and Data acquisition

The provided training set is rather small consisting of only 144 samples. Such a dataset size may hinder the generalization capabilities of models and potentially cause overfitting. Learning different weather and lighting as well as diverse geological and environmental conditions may be challenging with a limited dataset, resulting in reduced accuracy and performance on unseen data.

A. Data Augmentation

Our first approach to diversify the existing dataset was to use image augmentation techniques as seen in figure 1:

- **Perlin Noise in HSV Space**: Perlin noise was added in the hue, saturation, and value (HSV) space to introduce random color variations to the satellite images. This augmentation technique aids in making the model more robust to different lighting and weather conditions and color variations in real-world scenarios. Perlin noise has already been demonstrated as a successful augmentation technique [1]. Here, we extend its application to color images by utilizing the HSV color space.
- **ColorJitter**: Similarly ColorJitter was applied to the images to change brightness, contrast, saturation, and hue randomly.
- **Affine Warping with Mirror Padding**: Affine warping was employed to create more data from the training set by applying rotations, translations, mirroring and scaling to the images. Additionally, mirror padding was used, which elegantly provides a reasonable continuation of the roads. This technique effectively augments the dataset, increasing its size and diversity, which helps the model to generalize better.

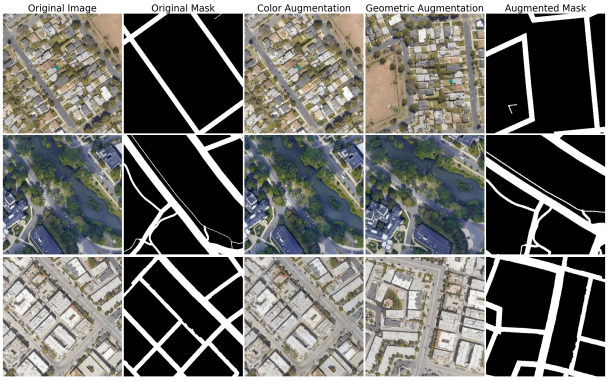

Figure 1: Three examples of augmentation.

B. Google Data

While data augmentation can help to expand the dataset in some capacity, it also is limited as the range of variations is constrained to what already exists in the original dataset, potentially limiting exposure to more diverse real-world scenarios. Further, the model may overfit to the augmented samples rather than generalizing well to unseen data.

To address these limitations and ensure a more comprehensive dataset, we acquired additional data by using the Google Maps API. A custom map key was created to match the black and white labels presented in the training data.

A careful examination of the environments in the training and test set revealed that the provided data most likely

originates from only two geographical regions. We found that the cities of Los Angeles and New Jersey seem to fit the provided data well. In order to best expand the given data and limit computational time, we created 3042 new samples in these two geographic regions.

C. Rich Labels

Rich features were computed from our labels (as seen in figure 2) to provide the model with more detailed information that could be leveraged in the post-processing step:

- **Signed Distance Function (SDF)**: We added a SDF to enable the model to have a better understanding of the road topology and to facilitate more accurate road boundary extraction. Notably, the SDF was computed as the distance to the road center as opposed to the road border since the road width in the ground truth labels appears to be inconsistent and hard to predict.
- **Road Width**: Instead road width information was added as a separate label to help the model produce more contextually relevant road extraction results and better deal with partial occlusion.
- **Road Directions**: Road directions were labeled to provide the model with a way of understanding road layouts augmenting the road width label in order to better predict road borders and fill in occlusions.

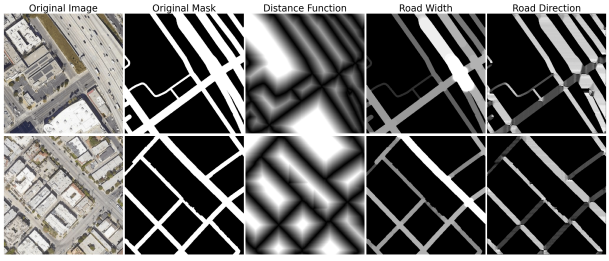

Figure 2: Two samples with rich labels.

III. Models and Approaches

A. U-Net

Initially, the plan was to use the U-Net [2] only as a baseline. But then we also investigated an improved version of the U-Net, that employs pre-trained decoders from the ResNet152 Architecture. This way we leverage both the power of the ResNet decoder to capture complex patterns but also the benefit of the U-Net to capture and retain fine and coarse details.

B. DeepLab

We initially adopted the DeepLabv3 ResNet50 Architecture [3] for its convenience, as it is readily available in the torchvision package and its ability to learn quickly. As the project progressed, we discovered that there are more suitable architectures for our specific task. For example, the DeepLab Architecture excels at handling different scales, which is not crucial for our setup.

C. Feature Pyramid Network

In our setting, a model would greatly benefit from a large receptive field. Because there are many large-scale features, that provide the model with the necessary hints to confidently predict the continuation of a street in a section that is occluded by trees or buildings. The Feature Pyramid Network architecture is designed to extract features at multiple scales, which makes it a perfect fit. We used the model Efficientnet-b7 with pretrained weights.

D. Loss functions

The choice of the training objective is crucial for the successful training of the model. We evaluated four different loss functions: F1 Loss, Mean Squared Error, and Cross Entropy Loss with 2 Classes, and Binary Cross Entropy Loss all with the same configuration of 25 epochs with DeepLab. Table I shows the score obtained on Kaggle with the given loss function. Judging from these results, we decided to stick with the Binary cross-entropy loss.

Loss function	Kaggle Score
F1 Loss	0.79666
Mean Squared Error Loss	0.86275
Binary Cross Entropy Loss	**0.86840**
Cross Entropy Loss with 2 Classes	0.83868

Table I: The Binary Cross Entropy Loss outperformed the other loss functions under the same conditions.

IV. Post-processing

A. Test-Time-Augmentation

The idea of Test-Time-Augmentation (TTA) is to query the model with differently modified versions of the input. The obtained range of output can then be used to get more stable predictions. In our case, TTA gives two benefits. Firstly, we can make the prediction as smooth as we want by using a lot of samples. Secondly, we can get valuable information from the distribution of outputs. For example, the standard deviation over the different predicted probabilities gives us a measure of the uncertainty. During TTA, we apply our color and geometric augmentations to get a wider range of inputs. Since we implemented our own geometric augmentation, we can extract the parameters and invert the transformation to map the output back into the original frame (see figure 4). We can then aggregate the obtained outputs with nan-median, nan-mean, and nan-std to get the mode prediction, mean prediction, and standard deviation of predictions (see figure 3. Since we used Sigmoid as the last activation function, we can do the aggregation before the activation function, which gives more weight to the more

extreme predictions. However, in practice, it seems not to make a notable difference.

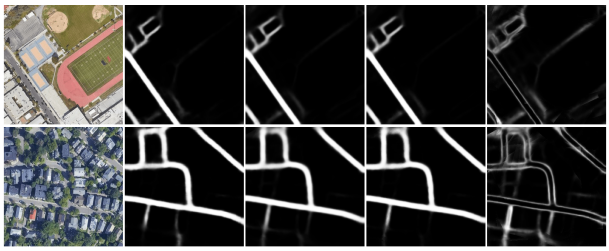

Figure 3: Three examples of Test-Time-Augmentation on test images with 50x samples.

Figure 4: Two samples of Test-Time-Augmentation on test images.

B. Thresholds

Since our approach is assessed by the F1 score, it is not straightforward to find the optimal patch prediction, from the probability output. The naive way to get a prediction for each 16x16 patch is to use a threshold to decide if a pixel should be labeled as road and a second threshold to decide for each patch, whether enough pixels are labeled as road. We expected it to be easy to find the optimal threshold because we can efficiently try all combinations with increments of 0.01 (see figure 5). However, what we found is that the F1 score decreases systematically with the number of samples that are considered. This is because, on a small set of samples, you can gain an increased F1 score by trading off recall against precision. However, this results in a most likely less accurate prediction the For example, if only one patch is in the foreground, you need to really find this patch to not get a recall of 0. In this situation, it would make sense to sacrifice a lot of precision to improve the F1 score. This explains why the optimal thresholds varied a lot across the samples. Thresholds used for the submission were optimized on a 5% split not seen during training.

C. Multilayer Perceptron

The main idea was to have a Multilayer Perceptron (MLP) optimized on the accuracy measure at hand (F1 score) which

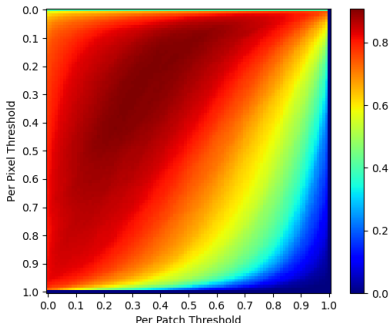

Figure 5: The F1 score computed over a set of 10 images for every pair of thresholds.

is a strictly more powerful approach than the thresholds described in the previous section. By training a separate MLP with ReLU activations, we can directly address the trade-off between precision and recall by gradient descent on the F1 score. First, we get all outputs of a patch, and we extract the quantiles $[0, 0.1, 0.2, ..., 0.9, 1]$ for each information we gather during TTA This makes sure that the MLP does not learn spatial patterns but only comes up with a complex set of thresholds. The intuition behind this approach was that the MLP operates on the output of an already powerful segmentation model, that should have leveraged every bit of information available in the dataset.

V. Results & Discussion

We compare our different configurations with the two baselines U-Net and DeepLab. Table V-A shows the comparison of our configurations. Indicated with + are Add-Ons, meaning that this feature was added to the model above it.

Configuration	F1 Score
Unet trained on google data	0.88424
Unet+ResNet Decoder trained on google data 21 epochs	0.90684
+ TTA 50x	0.90975
DeepLab trained on train data without augmentations	0.90202
DeepLab trained on train data	0.90244
DeepLab trained on google data 34 epochs	0.90328
+ TTA 50x	0.90660
+ MLP trained on 20x TTA output of train data	0.91784
+ DeepLab finetuned on train data	0.92591
FPN Efficientnet-b7 on google data 30 epochs	0.91704
+ TTA 50x	0.92068
+ MLP trained on 20x TTA output of train data	0.92017
+ FPN Efficientnet-b7 finetuned on train data	0.92875
FPN Efficientnet-b7 on google data 30 epochs	0.91704
+ FPN Efficientnet-b7 finetuned on train data	0.92406
+ TTA 50x	0.93024
FPN Efficientnet-b7 + TTA 50x on full data *	**0.93306**

Table II: All models were trained for 30 epochs on 90% of the data if not indicated otherwise. The F1 score was computed on Kaggle. *Trained with a special training configuration documented in section V-H

A. Results of Google Data

We observed that training the model on the additional Google data resulted in only a very minor improvement (as can be seen in). It also needed more epochs to train to achieve a similar result because the data set was larger. We expected a much bigger improvement from vastly expanding the training set, but it appears that even though the provided training set is very small in size it still covers the input space sufficiently for our chosen models.

B. Results of Data Augmentation

As indicated in table III, the data augmentations only gave a negligible improvement in F1 score. Given a detailed exploration of the dataset, we suspect that changing the scale of the images is problematic since all images in the dataset have the same scale. Furthermore since all images are presumably taken from just two regions with consistent lighting, it is also possible that changing the rotation might be problematic, because the shadow directions lose consistency which makes them harder to understand. Additionally as we saw for the Google data, it appears that the small provided dataset is already sufficient for our chosen models.

Configuration	Kaggle Score
DeepLab without augmentations	0.90202
DeepLab with augmentations	0.90244

Table III: The DeepLab model was trained for 30 epochs on 90% of the given training data and the submission was done with values (0.5, 0.25) as per pixel and per patch threshold.

C. Rich Labels

Even though the Deeplab model was able to predict the rich labels well, all approaches to improve our final prediction from the rich labels were unsuccessful.

D. Results of MLP

When adding MLP to the DeepLab model we gained a large improvement of 1.1%. For the FPN on the other hand, MLP did not improve the model any further because the FPN already learned those features.

E. Results of Test-Time-Augmentation

Test-Time-Augmentation consistently increased the Kaggle score by at least 0.3-0.6%. This makes it a very reliable improvement in our case.

F. Results of Fine-tuning

When training the best models on 90% of the train set we gained a substantial improvement of their score by 0.7-1%.

G. Comparison to baselines

Our two baselines U-Net (0.88424) and DeepLab (0.90202) lie above the TA baseline on Kaggle (0.86380). By adding the ResNet Decoder and TTA we were able to improve U-Net by 2.6%, which is close to the DeepLab baseline. With all our Add-Ons, we could improve DeepLab by 2.4%. The FPN scored better than DeepLab from the start, and with the best configuration of Add-Ons we could get a score 2.8% above the DeepLab, 4.6% above the U-Net and 6.7% above the TA baseline.

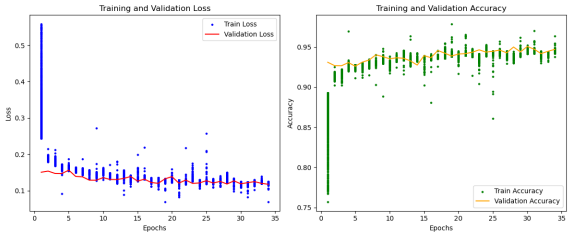

Figure 6: Loss and Accuracy for training DeepLab. Very similar for U-Net.

H. Competitive Score

To get a competitive score, we trained again the FPN on the complete dataset and fixed the scale of the geometric augmentations. We also added a second training run on the dataset with a lower learning rate. For the submission, we reused the thresholds (0.35, 0.35) from previous runs. The resulting Kaggle score is 0.93306.

I. Issues

An issue with the data set is that some of the ground truth labels do not match exactly with the roads. The labels are offset at some points and they have a different width than in the satellite image. When trying to compare our models, we noticed that the F1 score is not a proper scoring function, which means that in some cases you can perform better according to the F1 score with a worse model, as illustrated by Resin [4] in his paper in example 2.1. The F1 score is also not convex, which makes it a not ideal training objective.[5]

VI. Conclusion

Image augmentations and training on the Google data set gave a much smaller boost in F1 score than expected. Adding MLP improved DeepLab a lot but did not help the FPN. However, Test-Time-Augmentation was a reliable source of improvement for all models and fine-tuning notably increased the score of our models as well. We conclude that our best architecture is the Feature Pyramid Network with the Efficientnet-b7 backbone and the biggest improvement was achieved using Test-Time-Augmentation. In future work we would focus on more sophisticated loss functions like the Dice and Focal Loss as well as ensembles of multiple models.

REFERENCES

[1] H.-J. Bae, C.-W. Kim, N. Kim, B. Park, N. Kim, J. B. Seo, and S. M. Lee, "A perlin noise-based augmentation strategy for deep learning with small data samples of hrct images," *Scientific Reports*, vol. 8, 12 2018.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017.

[4] J. Resin, "From classification accuracy to proper scoring rules: Elicitability of probabilistic top list predictions," 2023.

[5] Y. Nan, K. M. Chai, W. S. Lee, and H. L. Chieu, "Optimizing f-measure: A tale of two approaches," 2012.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Road Segmentation of Satellite Images

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Horat	Andri
Karst	Matthias
Frischknecht	Nadine
Müller	Sarina

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 29.07.2023	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.