ROAD SEGMENTATION WITH MACHINE LEARNING

Levin HERTRICH
levin.hertrich@epfl.ch

Eugénie CYROT
eugenie.cyrot@epfl.ch

Gabriel MARIVAL
gabriel.marival@epfl.ch

***Abstract*—This report describes a Machine Learning approach to perform semantic road segmentation from satellite images. For this task training data was augmented and extended with external data, and different machine learning models were tested with tailored loss functions. The best-performing model combines a fine-tuned DeepLabV3+ and a fine-tuned SegFormer model, achieving an F1-score of 0.9140.**

I. INTRODUCTION

Road detection from satellite images is an important task with widespread applications in urban planning, autonomous driving, agriculture, and disaster response. The ability to accurately identify roads from satellite imagery enables efficient infrastructure planning, navigation, and logistics. Road detection is challenging due to the complexity of satellite imagery, including variations in environmental conditions, occlusions, and data imbalance between roads and background.

This report focuses on binary semantic segmentation of roads from satellite images. Our goal is to tackle the challenges in road segmentation by doing data preprocessing, including augmentations and train and evaluate different machine learning models together with tailored loss functions.

II. PROCESSING DATA

A. Data exploration

(a) Original Image (b) Mask

Figure 1: Satellite image and its corresponding mask

The dataset consists of 100 satellite images for training and their corresponding masks of size (400,400) and 50 satellite images for testing of size (608,608). The masks are not binary and their class distribution is imbalanced; only 20% of pixels correspond to a road on a given image. Moreover, the spatial distribution of road pixels is imbalanced too; pixels corresponding to a road seem to be less likely to be located in the top left corner of the satellite image. To avoid any biases, we will augment the dataset through random rotations and flips. Figure 1 presents a satellite image of the training set and its corresponding mask.

B. Data preprocessing

The masks contained in the training set are non binary. To convert them to a binary image, any pixel strictly above the threshold 128 (when the image pixel value range from 0 to 255) is converted into a white pixel and reciprocally, any pixel below the threshold is converted into a black pixel.

C. Data augmentation

To make our model more robust to inputs different from our training set, we have augmented the dataset. The augmentation pipeline used is presented in Table I.

Augmentation	Probability
Horizontal flip	0.5
Vertical flip	0.5
Random 90° rotation	0.5
Shifting, scaling and rotation	0.5
Random brightness contrast	0.5
Gaussian noise (p = 0.5) or Gaussian blur (p = 0.2)	0.7
Color adjustments (hue, saturation, value channel - p = 0.5) or Color interference (p = 0.3)	0.5

Table I: Augmentation pipeline

(a) True mask (b) Augmented image

Figure 2: Satellite image and one augmentation

For each image, we have chosen to perform 5 augmentations using the augmentation pipeline, one example is presented in Figure 2.

D. External Dataset

To increase the training size, we have used an external dataset used for segmentation of aerial images [1]. This dataset consists of satellite images of Paris, Berlin, Chicago and Zürich of varying sizes. The masks are 3 channels images, where blue pixels correspond to roads and red pixels correspond to buildings. In our case, we are only interested in the road labels; we processed the labels to have a binary masks containing only information about the roads. The external dataset is large; thus, we decided to only keep the Chicago images, because of their similarity in terms of roads shapes with our test set. We use the same augmentation pipeline on the Chicago dataset as on the original dataset.

III. Machine Learning Models

In this project we tried different machine learning models. As a starting point, we created a simple baseline using logistic regression. Next, we explored more complex, pretrained models, which we fine-tuned to adapt them to our specific task.

We chose to fine-tune pretrained models due to our limited amount of compute resources as well as the benefits of faster and more efficient training. We applied transfer learning by using the backbone of the pretrained models and adjust the output layer if necessary to align with our binary semantic segmentation task.

A. Logistic regression

Logistic regression estimates the probability of belonging to a certain class; here, a pixel belonging to a road or not. We used sklearn logistic regression with an L2 regularization.

B. U-net++

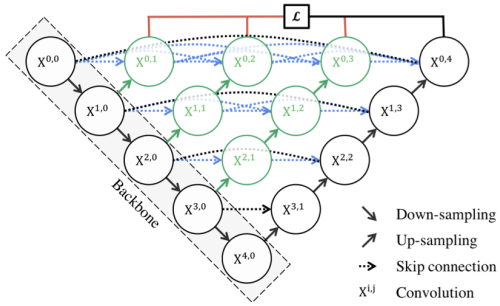


Figure 3: High-level Architecture of Unet++ from [2]

U-Net++ is a fully convolutional neural network designed for semantic image segmentation. It consists of an encoder-decoder architecture with skip connections [2]. The model is based on the well-known U-Net, which was developed for biomedical image segmentation [3]. U-Net++ differs from the original U-Net through its redesigned skip pathways and the introduction of deep supervision [2]. Figure 3 shows the high-level architecture of U-Net++.

We chose U-Net++ for this project because it inherits the strong performance of U-Net for image segmentation tasks while offering improvements across a variety of datasets [2].

For implementation we used a pretrained model from the Segmentation Models Pytorch library [4]. We used a pretrained ResNet50 encoder [5] with weights pretrained on the ImageNet dataset [6]. The ResNet50 encoder provides a good trade-off between model complexity and segmentation performance. Furthermore, ImageNet pretrained weights offer a good foundation for fine-tuning due to the diversity of images included in ImageNet [6].

C. DeepLabV3+

DeepLabV3+ is a deep convolutional neural network designed for semantic image segmentation. It extends the DeepLabV3 architecture by incorporating an encoder-decoder structure to refine segmentation results. Specifically, it uses the DeepLabV3 model [7] as an encoder to extract rich contextual information and a simple decoder module to refine segmentation results along object boundaries [8]. Figure 4 shows the structure of the DeepLabV3+ model.

DeepLabV3+' combination of high-level semantic information from the encoder with low-level features to refine along object boundaries makes it well-suited for segmenting fine road boundaries from satellite images.

To implement the model we used again the Segmentation Models Pytorch library [4]. Initially, we chose again the pretrained ResNet50 encoder [5] with weights pretrained on the ImageNet dataset [6] due to its balance between computational complexity and performance. When training this model with the augmented Chicago dataset, we replaced the ResNet50 encoder with a ResNet101 encoder. Our goal was to determine if the the ResNet101 encoder with pretrained ImageNet weights helps the model to generalize better on the larger dataset.

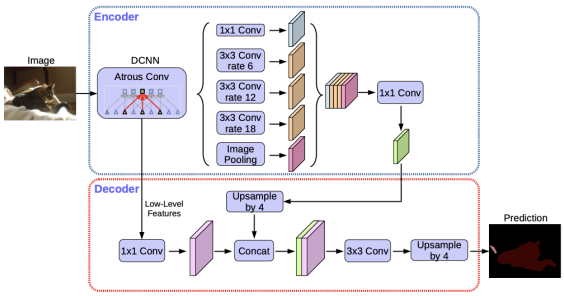


Figure 4: DeepLabV3+ Structure from [8]

D. SegFormer

After obtaining good results with convolutional neural networks, we wanted to test a transfomer model as well. Therefore, we selected the SegFormer model, a simple and efficient but powerful semantic image segmentation framework. It unifies the transformer architecture with lightweight multilayer perceptron decoders [9]. Figure 5 shows the SegFormer architecture.

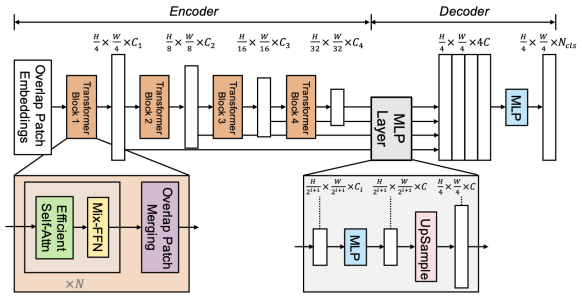


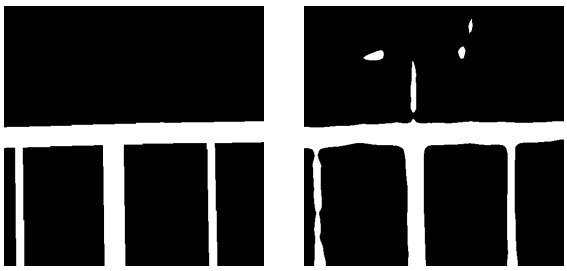
Figure 5: SegFormer Framework from [9]

We chose SegFormer for this project because of its strong segmentation performance and its comparable lightweight architecture.

For implementation we used the pretrained MiT-b3 model from Hugging Face [10]. We chose the b3 variant because it seems to offer the optimal trade-off between segmentation performance and model size [9]. Since the pretrained model originally predicts 150 classes, which is not suitable for our binary segmentation task, we replaced the original SegFormer decoder head with a custom head consisting of a 1x1 convolutional layer to map the feature channels to the desired number of labels.

E. Combined approach

During manual inspection of the predicted masks from the models, we observed the existence of small misclassified artifacts in some predictions, as shown in Figure 6b. To address this issue, we tested a method that combines the predictions of the previously trained models.



(a) True mask (b) Prediction

Figure 6: True mask and its corresponding predicted mask

The idea behind this approach is that the per-pixel probabilities for these artifacts are often close to 0.5, as the pixels in the artifact regions are difficult to classify confidently. By combining the probabilities from two different models, we expect that one model will classify the pixels in these regions more accurately, assigning a probability farther from 0.5. Averaging the probabilities from both models allows us to smooth out small misclassifications.

For the combination, we selected the two best-performing models: DeepLabV3+ and SegFormer. Because SegFormer performed better than DeepLabV3+ we weighted the prediction from DeepLabV3+ with 0.3 and the prediction from SegFormer with 0.7.

IV. Training and Experimental Setup

Our data is unbalanced, with significantly more background pixels than road pixels. To address this, we evaluated various loss functions, focusing on those designed for unbalanced data. Specifically, we tested Binary Cross Entropy (BCE) Loss, Combo Loss, Dice Loss, Tversky Loss, and Weighted BCE Loss. While BCE Loss is not inherently suited for unbalanced data, its weighted variant accounts for this issue. A detailed explanation of these loss functions is provided in appendix A. The loss functions were evaluated with the original augmented dataset, and the best-performing one was then used when fine-tuning the models with the augmented Chicago dataset.

We randomly split the dataset into train, validation, and test sets, with 20% for testing and 10% of the remaining data for validation. Given the sufficient dataset size and to optimize runtime, k-fold cross-validation was not applied.

To evaluate model performance we used the $F1$-score. Therefore the predictions and ground truth were divided into 16x16 patches, with each patch classified as either road or background based on a specified threshold. Testing different thresholds during training showed 0.25 provided the highest $F1$-score, reflecting the data imbalance.

Each model was trained for 10 epochs, with the validation loss tracked during training. After training, the model from the epoch with the lowest validation loss was selected for evaluation on the test set using the described $F1$-score calculation.

We used the AdamW optimizer with a learning rate of 1e-4. This learning rate performed well across all models. The validation loss indicated overfitting during the 10 epochs, but not immediately at the start. AdamW adapts the effective learning rate for individual parameters based on gradient history, so no further learning rate fine-tuning was performed.

V. Results

In a first step we evaluated the performance of our models using the different loss functions.

Model	BCE	Combo	Dice	Tversky	WBCE
Unet++	0.8709	**0.8767**	0.8605	0.8437	0.8300
DeepLabV3+	0.8822	**0.8855**	0.8422	0.8500	0.8323
SegFormer	**0.8934**	0.8893	0.8907	0.8793	0.8664

Table II: $F1$-scores for different loss functions on the original augmented dataset

Table II shows that the convolutional neural networks (Unet++ and DeepLabV3+) perform best with the Combo Loss function. This result is reasonable, as Combo Loss balances the unbalanced dataset by combining Dice Loss and a modified version of Cross-Entropy Loss. In contrast, SegFormer achieves its highest $F1$-score using standard Binary Cross-Entropy (BCE) Loss. This is expected to some extent because the SegFormer backbone architecture was originally trained with BCE Loss.

After determining the best loss function for each model, we fine-tune the models with the original augmented dataset combined with the augmented Chicago dataset.

Model	$F1$-score
Logistic regression	0.398
Unet++	0.8601
DeepLabV3+	0.9005
SegFormer	0.9128
Combined	**0.9140**

Table III: $F1$-scores on the original augmented dataset combined with the augmented Chicago dataset

Table III shows that all models, except Unet++, benefit from the addition of the augmented Chicago dataset during training. The lower performance of U-Net++ could be attributed to overfitting, as we did not increase the encoder size when training with the additional data.

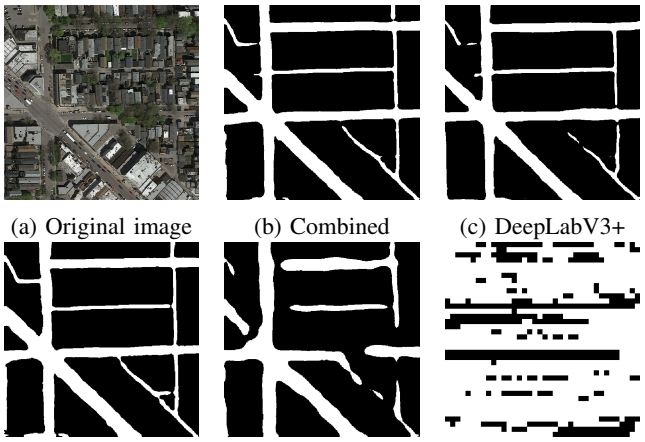
Our best-performing model is the combined approach of SegFormer and DeepLabV3+, which achieves an $F1$-score of **0.9140**.

VI. Discussion

Our results demonstrate that, for the given dataset, a transformer-based architecture provides the best performance, with SegFormer being the single best-performing model. We found that its performance can be further improved by combining it with the DeepLabV3+ model, which helps to reduce misclassified artifacts. This is an interesting result, as it shows that instead of optimizing one single model, results can also be improved by combining existing models.

In Figure 7, we can see the predictions across the models we tested for a sample test image. A manual inspection of the predictions aligns with the quantitative results, where SegFormer produces significantly better predictions compared to models like U-Net++ while logistic regression performs poorly, with predictions appearing almost random. This shows that for complex tasks like semantic road segmentation, more sophisticated neural network architectures are better suited.

We have also observed that the choice of loss function is crucial for model performance, especially when working with unbalanced data. Combo Loss proved effective for the convolutional models we used, adressing the imbalance between road and background pixels in our data. The fact that BCE Loss worked best for SegFormer shows that when fine-tuning a model, the loss function used to train the backbone of the model has to be taken into account.



(a) Original image (b) Combined (c) DeepLabV3+

(d) SegFormer (e) U-Net++ (f) Logistic Regression

Figure 7: Original image from the test set and its corresponding predicted masks for each model

Additionally, our results indicate that model performance can be enhanced by incorporating additional training data, if the model has sufficient capacity to avoid overfitting. For instance, while SegFormer and DeepLabV3+ benefited from the augmented Chicago dataset, U-Net++ struggled, likely due to its smaller encoder.

VII. Conclusion

In this project we investigated machine learning methods for semantic road segmentation from satellite images. We evaluated a logistic regression model, deep convolutional architectures with U-Net++ and DeepLabV3+ and SegFormer, a transformer based model.

Our work shows that the choice of loss function and dataset size significantly impacts model performance. Combining predictions from different models reduced misclassifications and led to improved segmentation results. Notably, our best approach achieved an $F1$-score of 0.9410, demonstrating its effectiveness.

Future work could further explore model combination strategies, such as using more models or optimizing the weighting of predictions.

VIII. Ethical risks

Road segmentation uses satellite images for training and use; our work could be exploited for purposes such as unauthorized tracking or mapping of private properties. The misuse of this technology could lead to several negative scenarios, depending on the intent and context of the application.

This could affect companies, government agencies such as the military and even an individual.

For example, a malicious actor could use satellite imagery to monitor the manufacturing facilities of competing companies and track their organizational growth and operational expansion over time. This type of industrial espionage could provide unfair competitive advantages and undermine corporate confidentiality. Similarly, governments could use our work to monitor private property for undeclared construction activity, enabling stricter enforcement of tax laws. While this may seem benign or even beneficial from a regulatory perspective, it raises significant privacy concerns. In the event of war, the military could even automatically plan attacks based on road detection by inputting a satellite image of the other country.

This risk can range from likely and not too severe for the example of tracking private property for taxes, to severe but less likely for planning attacks in a war.

In our case, we could not take this risk into account as it is too complex to detect if an image is from a private property or a government property. Moreover, determining the intent behind the use of the model is an inherently difficult task. To try to mitigate, the model could be private or identify user activity to monitor the use and block malicious users.

References

[1] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning aerial image segmentation from online maps," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 11, p. 6054–6068, Nov. 2017. [Online]. Available: http://dx.doi.org/10.1109/TGRS.2017.2719738

[2] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1807.10165

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

[4] P. Iakubovskii, "Segmentation models pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[7] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587

[8] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *CoRR*, vol. abs/1802.02611, 2018. [Online]. Available: http://arxiv.org/abs/1802.02611

[9] 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 *Neural Information Processing Systems (NeurIPS)*, 2021.

[10] H. Face, "Segformer model overview," https://huggingface.co/docs/transformers/en/model_doc/segformer#overview, 2024, accessed: 2024-06-17.

[11] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2020, pp. 1–7.

APPENDIX A.
LOSS FUNCTIONS

In this project, we used different loss functions in order to deal with class imbalances and improve the performance of our models. [11]

A. Binary Cross-Entropy

Cross-Entropy is a measure of dissimilarity of two probability distributions. In our case of binary classification, we can use the Binary Cross-Entropy defined as follows :

$$L_{BCE}(y, \hat{y}) = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}))$$

This loss seek to quantify the differences in information content between the actual and predicted image masks. It often works best with balanced data.

B. Weighted Binary Cross-Entropy

Weighted Binary Cross-Entropy is a variant of Binary Cross-Entropy, designed to handle skewed data by attributes different weights to the positive and negative examples.

$$L_{WBCE}(y, \hat{y}) = -(w_0 * y \log(\hat{y}) + w_1(1 - y) \cdot \log(1 - \hat{y}))$$

In our road segmentation task, this approach can be particularly effective when addressing the imbalance in the spatial distribution of road pixels, as the majority of pixels often correspond to the background.

C. Dice Loss

The Dice loss is a metric derived from the Dice Similarity Coefficient, which quantifies the similarity/overlap between the predicted and true segmentation of an image.

$$DSC = \frac{\text{True positives}}{\text{Number of positives + Number of false positives}}$$

The Dice loss is then defined as follows :

$$DL(y, \hat{p}) = 1 - \frac{2y \cdot \hat{p} + 1}{y + \hat{p} + 1}$$

where y represents the ground truth mask of the image, and $\hat{p}$ the predicted segmentation. The +1 is added to ensure the fraction not to be undefined in cases where y = $\hat{p}$ = 0.

Dice Loss is often well-suited for imbalanced data, as it prioritizes the overlap between foreground pixels (i.e road regions) rather than giving equal importance to all pixels.

D. Tversky Loss

Tversky Loss is a generalization of the Dice Loss, designed to address the limitations of Dice Loss in highly imbalanced datasets. It adds a weight β to false positives (FP) and false negatives (FN), making it particularly useful for segmentation tasks where one class is significantly underrepresented.

Similarly to Dice loss, we define the Tversky index as :

$$TI(y, \hat{p}) = \frac{y \cdot \hat{p} + 1}{y \cdot \hat{p} + \beta(1 - p)\hat{p} + (1 - \beta)y(1 - \hat{p}) + 1}$$

and the Tversky loss as :

$$TL(y, \hat{p}) = 1 - \frac{y \cdot \hat{p} + 1}{y \cdot \hat{p} + \beta(1 - p)\hat{p} + (1 - \beta)y(1 - \hat{p}) + 1}$$

It usually works well with balanced data .

E. Combo Loss

Combo loss is a combination of Dice Loss (DL) and a modified Cross-Entropy (L_{m-BCE}), by computing a weighted sum of both. It is defined as :

$$CL(y, \hat{y}) = \alpha \cdot L_{m-BCE}(y, \hat{y}) + (1 - \alpha) \cdot DL(y, \hat{y})$$

$$L_{m-BCE} = -\frac{1}{N} \sum_{i}^{N} \beta(y - \log(\hat{y}) + (1 - \beta)(1 - y) \log(1 - \hat{y})$$