Project report

Deep learning for semantic image segmentation

AZERINE Lorenzo

CARRON Louis

1 Problem statement

Problem Description

This project focuses on developing a Deep Learning model for semantic image segmentation. Specifically, we address a binary and multi-class segmentation problem involving **five classes**. This report details the binary segmentation problem as the oral presentation was focused on the multiclass segmentation problem. The goal is to accurately classify each pixel in an image into one of the predefined categories. Image segmentation is a challenging task due to :

- **Class imbalance :** Some classes can be underrepresented in the dataset, leading to biased predictions. Especially for the multiclass.
- **Dataset size :** Limited labeled data makes training complex models more difficult. We have only 400 images available.
- **Complexity of the model :** Large architectures require significant computational resources and are prone to overfitting on small datasets. However, thanks to the Gricad cluster we are able to use several powerful GPU to do our calculation.

Dataset Overview

The dataset contains images with corresponding ground truth segmentation masks. Let's have a look at several images.

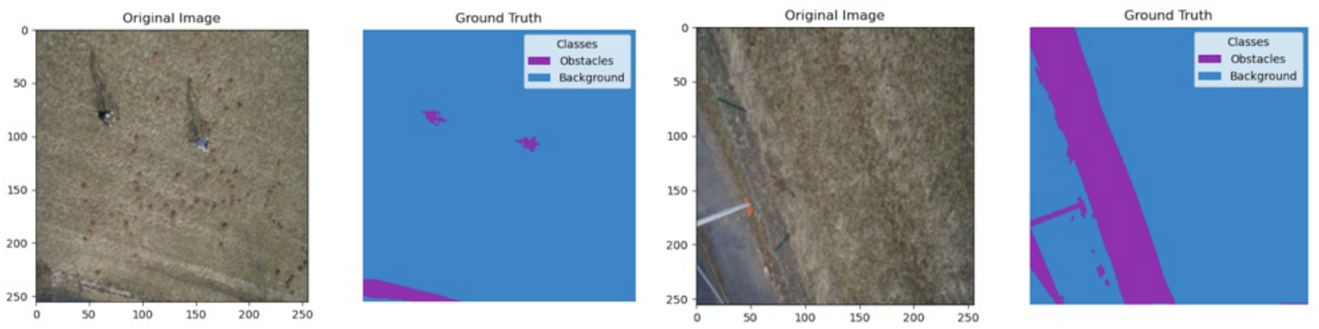


FIGURE 1 – Images and corresponding binary masks

When we look at these two examples, it is hard to tell which one is easy and which one is hard. This situation applies to the entire dataset. A difficult image cannot be precisely defined ; in each image, there are several aspects that can be considered difficult. In our two examples, we can identify several difficulties. On the left image, the bikers can be considered small compared to the rest of the image, and inside their arms, we can see the background, meaning that the model will have to differentiate the background from the bikers' bodies and the bike over a very limited area. Additionally, this image illustrates class imbalance as the background account for more than 80% of the image. This image is not singular, there are several images showing a such class imbalance. A such class imbalance may biased our models. On the second image, the fence can be considered easy because it has a regular rectangular shape, but as the fence is not complete, we can still see the background behind it. This is an issue, as the model might have a hard time deciding whether the background is part of the fence or not. Overall, the dataset is highly complex, presenting significant challenges that demand a robust and well-optimized model to effectively capture its patterns.

Data Protocol & Data Augmentation

The dataset consists of 400 images, we divide it into three parts. First, 50 images are reserved for the test set. The test set is kept the same to ensure consistent comparison of results across different training parameters and models. The remaining images are then randomly split, with 80% allocated for training and 20% for validation. In the end, we are left with 280 images for training. However, due to the complexity of the images—various types of obstacles, differing background colors, and variations in lighting conditions—280 images initially seem insufficient to train a reliable model capable of capturing fine details.

To address this issue, we focus on data augmentation. The goal is to generate ”new” images from the existing dataset. Instead of creating entirely new images, we apply small modifications to the existing ones, enabling the model to train on a more diverse dataset. We chose four augmentation techniques : rotation, horizontal and vertical flipping, zoom and crop, and resizing to the nearest size. For rotation, we added a small zoom afterward to prevent the appearance of black bands on the sides of the images. The zoom factor was determined empirically to ensure that most images have negligible black bands.

2 Models Architectures

U-Net

We started with the U-Net architecture of [1] as the base model. U-Net is based on a CNN architecture designed to process and analyze images. It works by applying convolutional layers that use filters to detect patterns in the input data. These layers are followed by pooling layers, which reduce the spatial dimensions of the data, helping to retain important features while minimizing computational complexity. Finally, fully connected layers are used to interpret the extracted features and make predictions or classifications.
We keep the architecture of the model but we adapt the size of the input and the output. The encoder captures spatial information by downsampling the input image through convolutional and pooling layers. Then the decoder upsamples the features back to the original resolution to produce the segmentation mask. Additionaly, U-Net introduces skip connections that directly link encoder layers to corresponding decoder layers, these connections provide a recovery of details lost during downsampling. Thus, U-Net is expected to be strong at capturing detail on the images at a cost of maybe struggling with global context of the images (background).

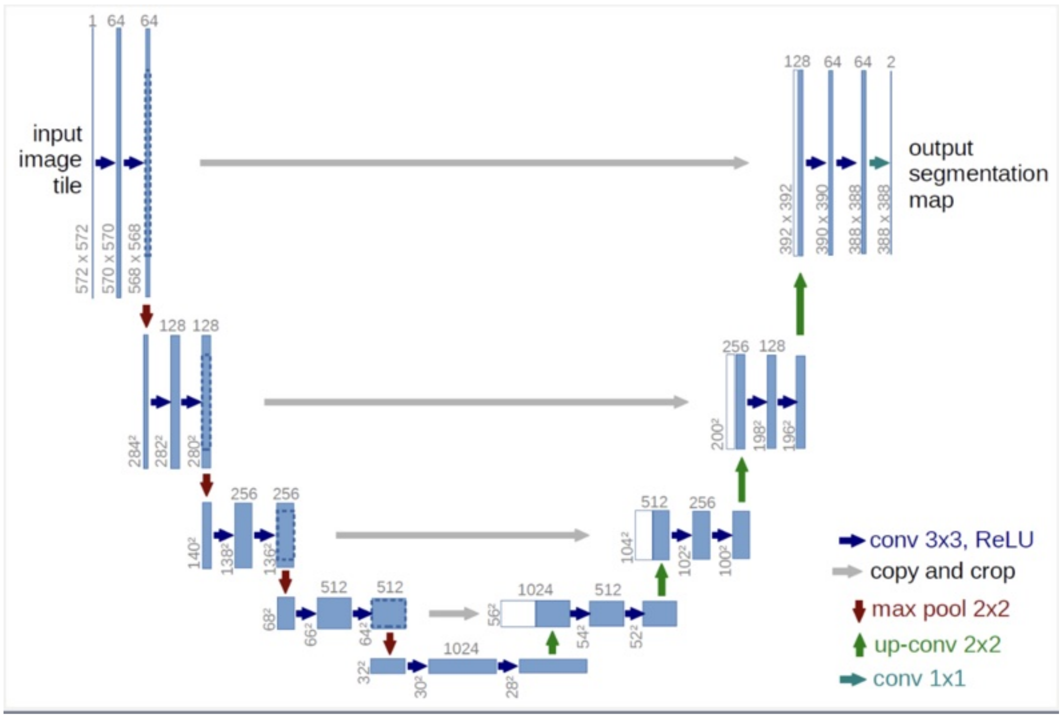


Figure 2 – Architecture of the U-Net

However, after some testing we understood that we need to improve the model so we switch a pre-trained encoder instead of training it by ourselves. We freeze the weights of the pre-trained encoder so it keeps its performances while we train the decoder.

SegFormer

We use the SegFormer architecture of [2]. SegFormer rely on a transformer-based model used for semantic image segmentation. Using the architecture of transformers, it is strong at capturing global relationship in an image. The core of SegFormer is based on transformer-based encoders, which capture ”long-range” dependencies in images, allowing the model to understand global context. The model employs a Multi-Layer Perceptron (MLP) decoder that reconstructs the segmentation map, effectively combining fine-grained details with global information.
Knowing that, we chose segFormer as its capabilities seem complementary to U-Net capabilities.

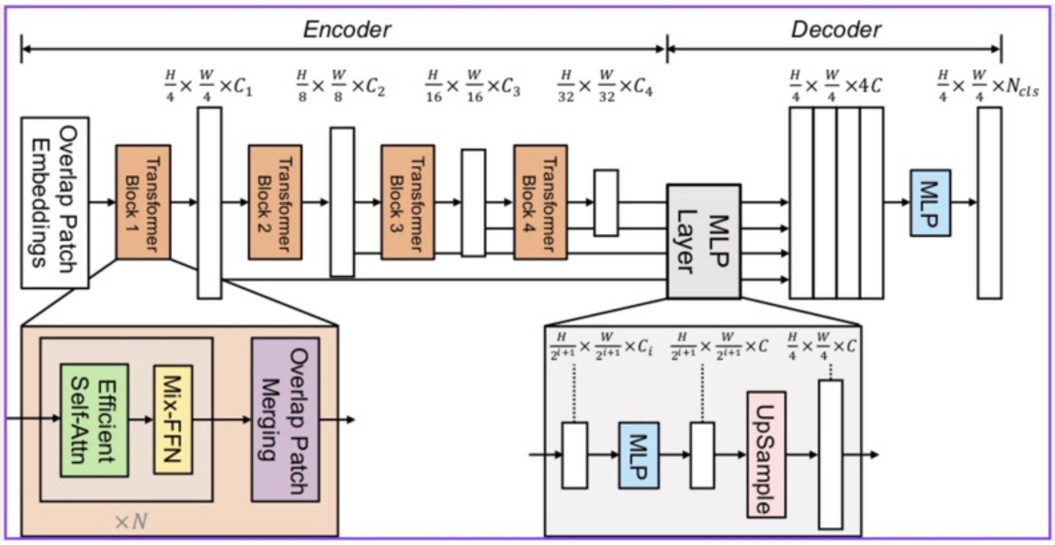

Encoder
Decoder
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
×N
UpSample
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$

FIGURE 3 – SegFormer architecture from Huggingface

The transformer encoder is built using the self-attention mechanism, which allows the model to focus on different parts of the image when processing each pixel. The self-attention mechanism computes relationships between all pixels in the image by calculating attention scores. This score allows the model to understand which pixels are important in relation to others, even if they are far apart in the image. In SegFormer, the encoder is designed to process the image at multiple scales. This means that it extracts features at different levels of resolution, which helps capture both local and global information.
Compared to traditional transformers that apply self-attention to all pixels in the image, SegFormer introduces an attention mechanism that reduces the computational cost while maintaining the ability to capture long-range dependencies. This is achieved by using a attention mechanism that limits the attention to a smaller set of pixels or regions.
On the decoder, instead of a MLP as the last layer we choose to do a convolution to reduce the number of parameters and the computational cost.

2.1 Uformer

The final model is a combination of the two previous, as explained before U-net is strong at capturing local information while SegFormer is strong as capturing local information. We want to associate these two capabilities so we combine the output of the two model by taking a weighted average.

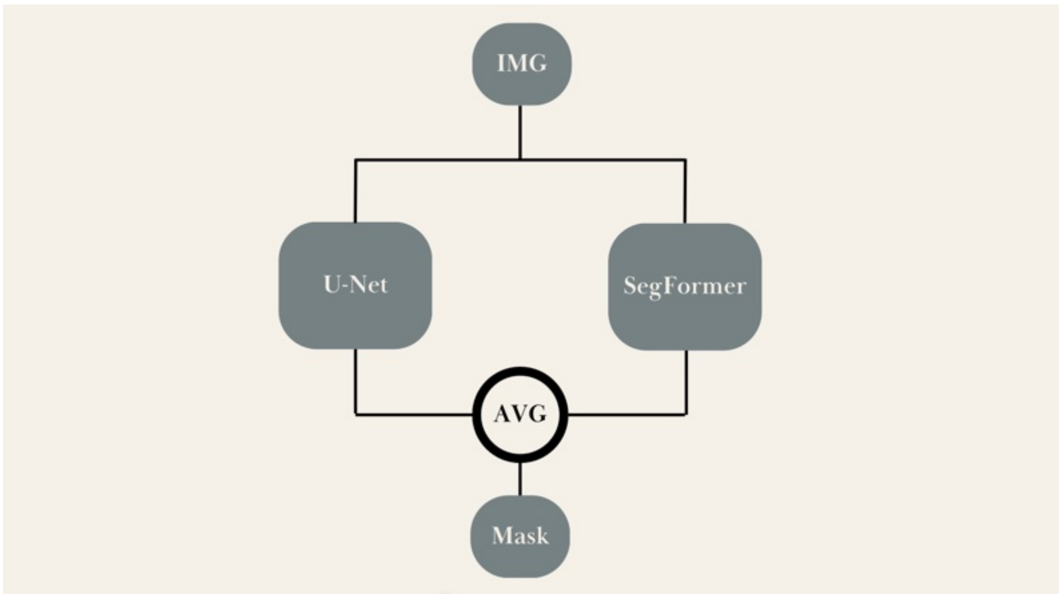


FIGURE 4 – Diagram of the UFormer

Training and Results

Metrics

The first metric is Mean Pixel Accuracy (MPA), is metric measuring the proportion of correctly classified pixels for each class, averaged across all classes. This metric is provides a straightforward idea of the performances of the model however, it is flawed for dataset with class imbalances. If one class dominates the dataset, the model might achieve a high MPA simply by predicting the dominating class correctly most of the time, even if it performs poorly on smaller classes. However on the binary dataset we can consider that the two classes are not imbalanced enough to biaised the MPA.

$$\text{MPA} = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{True Positives}_i}{\text{Total Pixels}_i}$$

The second metric is the Intersection Over Union (IOU). IOU measures the overlap between the predicted segmentation and the ground truth for each class. IOU is very relevant for segmentation as it evaluates how well the predicted segmentation overlaps with the ground truth by taking both false negative and false positive. It also penalize misclassifications and is less affected by class imbalances as it compares results to the gound truth.

$$\text{IoU}_i = \frac{\text{Intersection}_i}{\text{Union}_i} = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i + \text{False Negatives}_i}$$

In the end, we choose to combine these to metrics as MPA offers insights into pixel-wise classification accuracy, while IoU gives a more spatially sensitive measure of segmentation quality, taking into account both overlap and misclassifications.

Results

As we did a lot of training and got a lot of quantitative results we will focus on we think are relevant figures for this report instead of listing every number we have.

Tversky Loss

We chose to use Twversky loss as a loss function. Tversky loss is a loss function useful for imbalanced datasets. It is a generalization of the Dice coefficient, designed to handle cases where the classes are imbalanced. The Tversky loss introduces two parameters, α and β, which control the weight given to false positives and false negatives, respectively.

$$\text{Tversky Loss} = 1 - \frac{TP}{TP + \alpha \cdot FP + \beta \cdot FN}$$

However, in our situation it seems hard to define if we should prefer False negative or false positives. Indeed, this loss function is highly relying on the application to fix the parameters. For instance, for autonomous driving we want penalize false negatives to prevent missing a sign or another car. Conversely, for spam e-mail detection we want to penalize false positives to prevent important e-mail to be sent to the spam box.
However, we still tried to optimize the parameters to enhance model performances.

U-Net

The first model is the U-Net, we started with the whole model as untrained but the model performances were low, the loss couldn't go lower than 0.55. So we rapidly switched to a pre-trained encoder where the results are more interesting. We chose to use an encoder trained on the ImageNet database as it is a very large and well-known database. We fixed the learning rate at 10^{-3} and the batch size at 16.

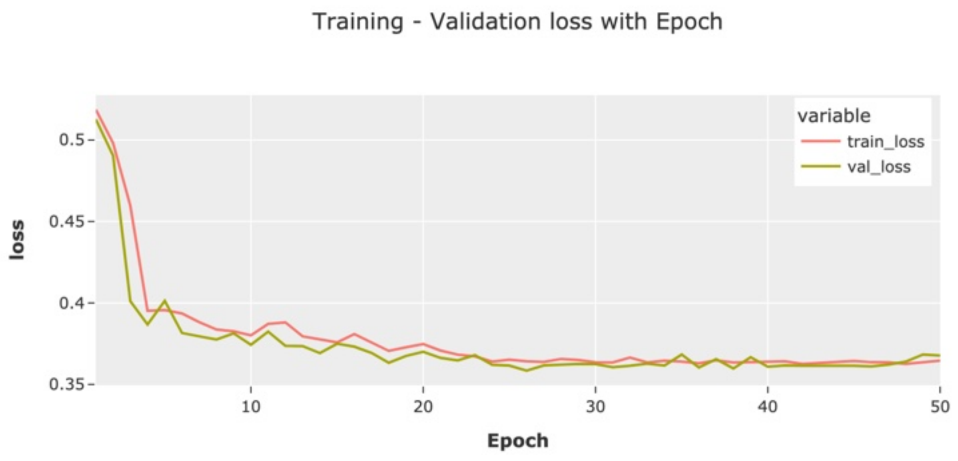


FIGURE 5 – Loss of the binary U-Net with pre-trained encoder

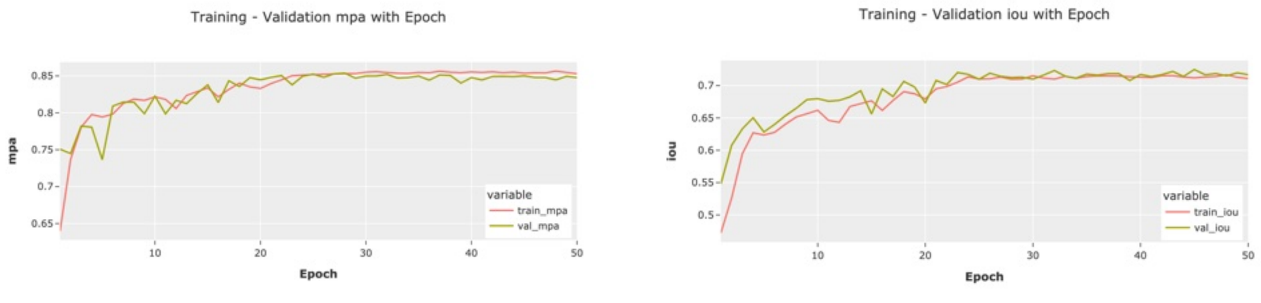


FIGURE 6 – MPA and IOU of the binary U-Net with pre-trained encoder

We notice that the model is effectively learning and the metrics are providing better results after each epoch. However, from around epoch 25 we start to be on a plateau and the model struggle to improve its performances. There are several potential reasons for a such situation :

- Learning rate value : if the learning rate is too high or too low the model will either fluctuate rapidly or move slowly.
- Local minima : The loss function might have local minima where the optimizer gets stuck. These are points where the gradient is close to zero, but they are not the global minimum.
- Insufficient data : if the training data is insufficient the model might be unable to learn pattern in the data leading to a struggle in generalization.

To tackle the first issue we implemented a scheduler to adjust the learning rate during the training. We use reduce on plateau with a factor of 0.1 and a patience of 3, meaning that if the model doesn't improve for 3 epochs the learning rate will be multiplied by 0.1. Then to solve the second issue, we choose to add Adam optimizer, it adapts the learning rate during the training to prevent from situation as described before. However, the results showed were obtained using Adam and reduce on plateau so those solutions are not enough. So the reason might be because of the data. To solve this issue we implement data augmentation and we train with 500, 1000 and 1500 augmented images.

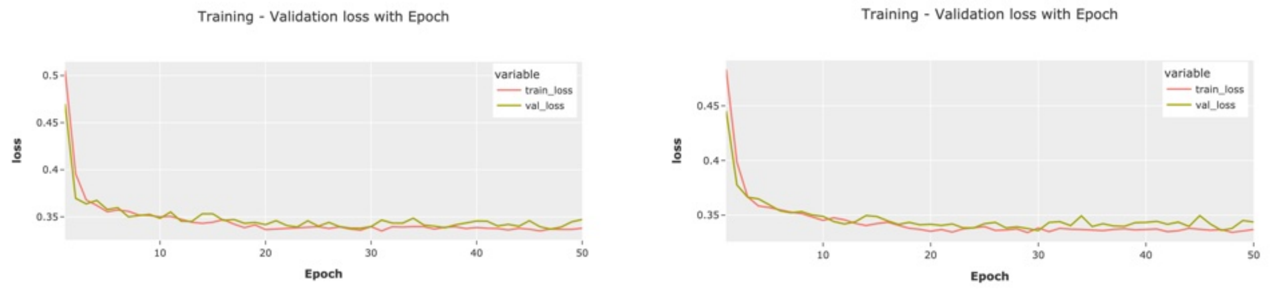


FIGURE 7 – Loss of the binary U-Net with 500 and 1000 augmented images

We see that adding 500 augmented images improved the results as we went from a loss of 0.37 to a loss of 0.33 but we still see a plateau from around epoch 25. If we add more augmented images we don't have better results the model seems to be unable to generalize more. To visualize we can compare prediction without augmentation and with 500 augmented images.

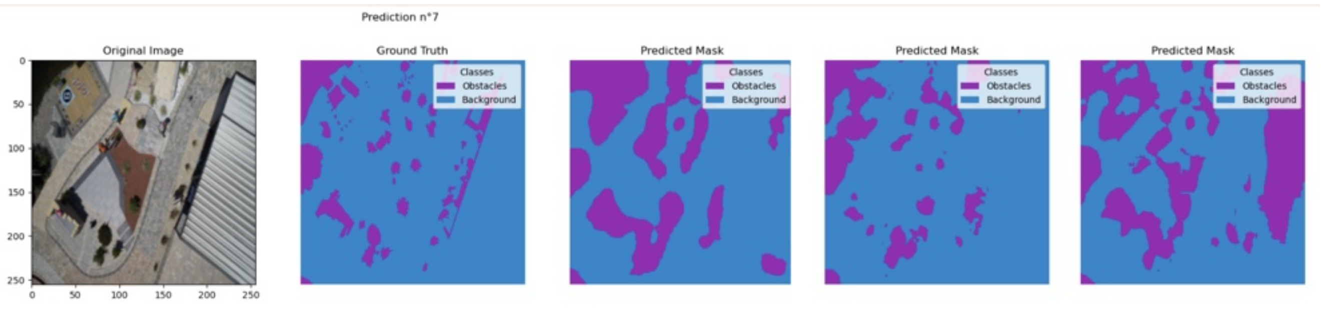
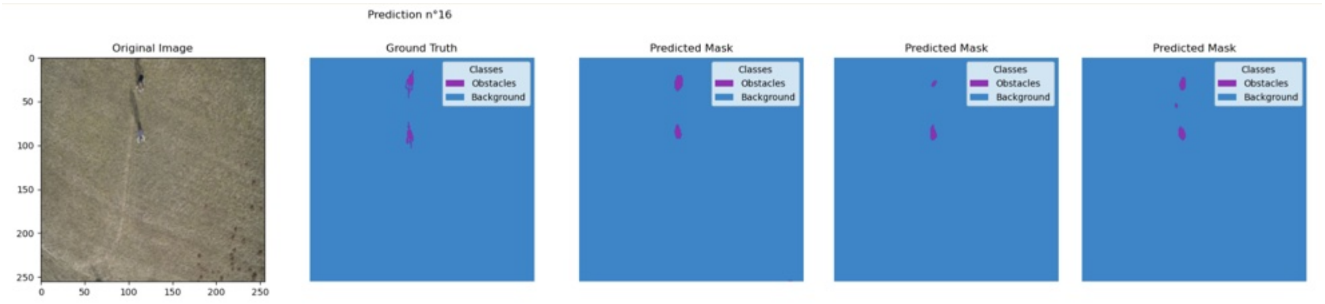


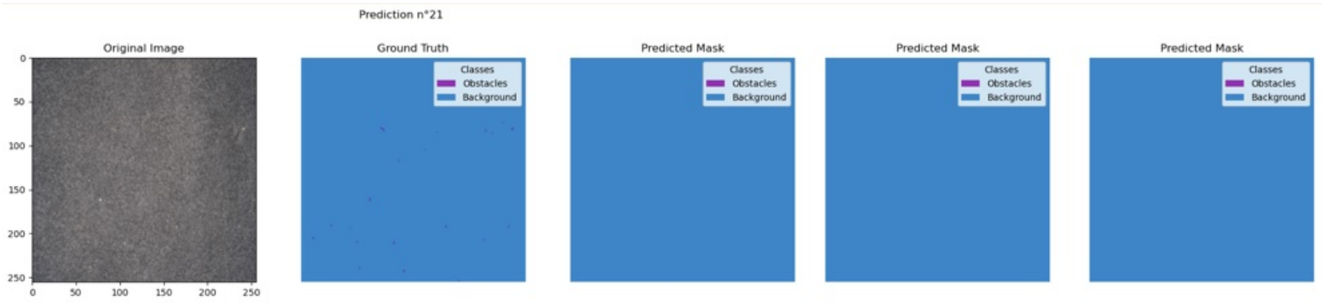
FIGURE 8 – Predicted mask without, with 500 and with 1000 augmented images

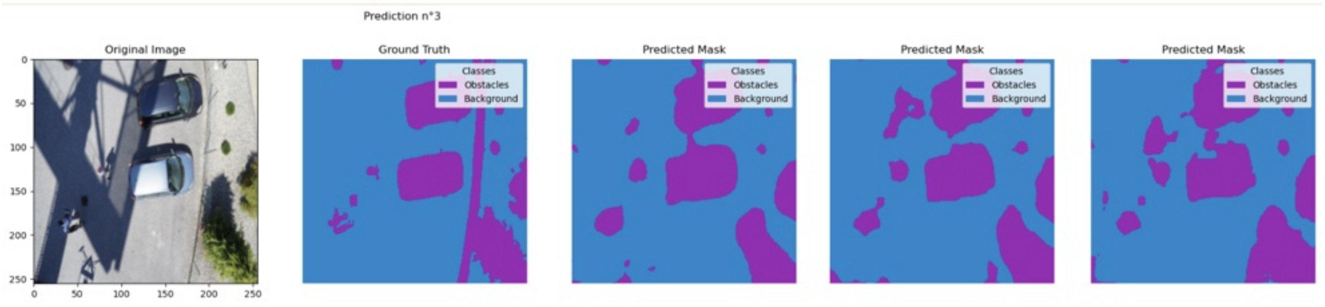
It appears that the model is effectively providing better results for with 500 augmented images but for 1000 augmented images we can't see significant improvement compared to 500 augmented images. We then provide the 5 best and 5 worst results.

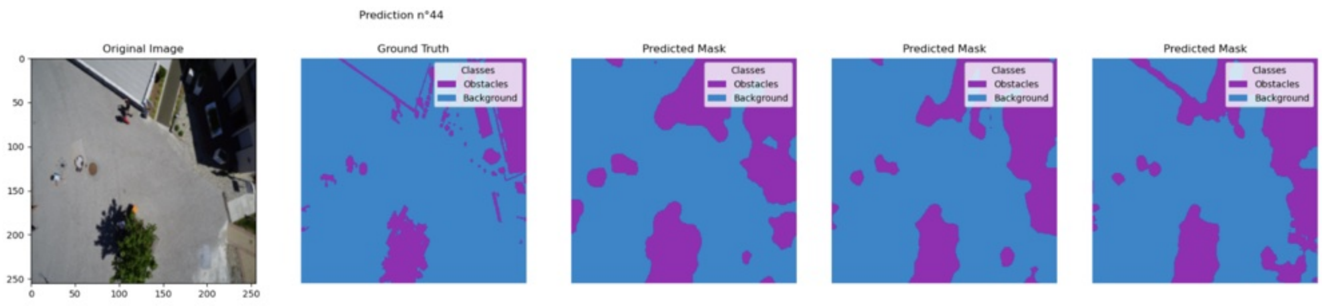
Best Predictions

All the following plots are organized as follow : Initial image, ground truth, predicted mask without data augmentation, predicted mask with 500 augmented images, predicted mask with 1000 augmented images.









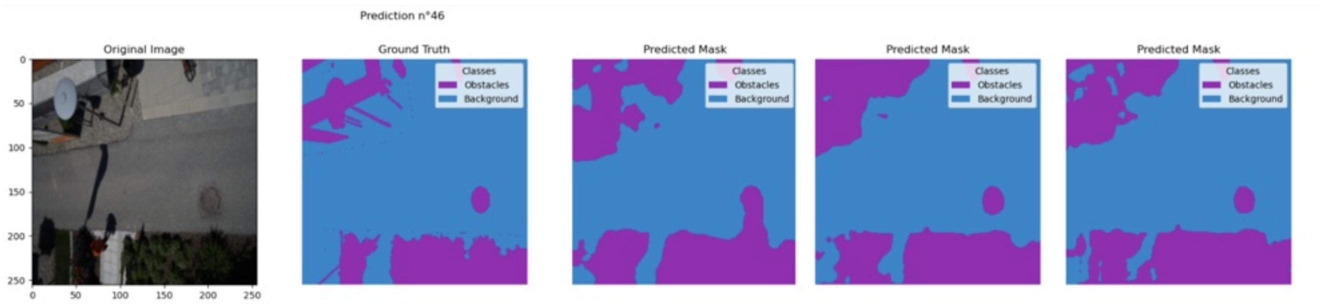
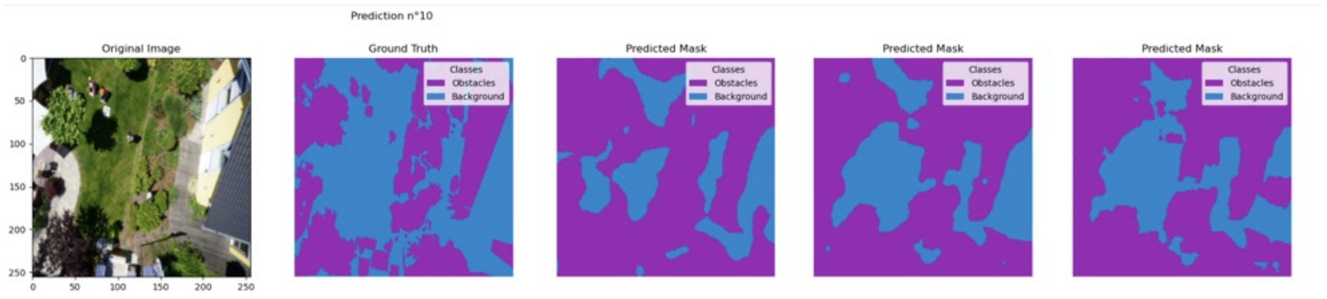
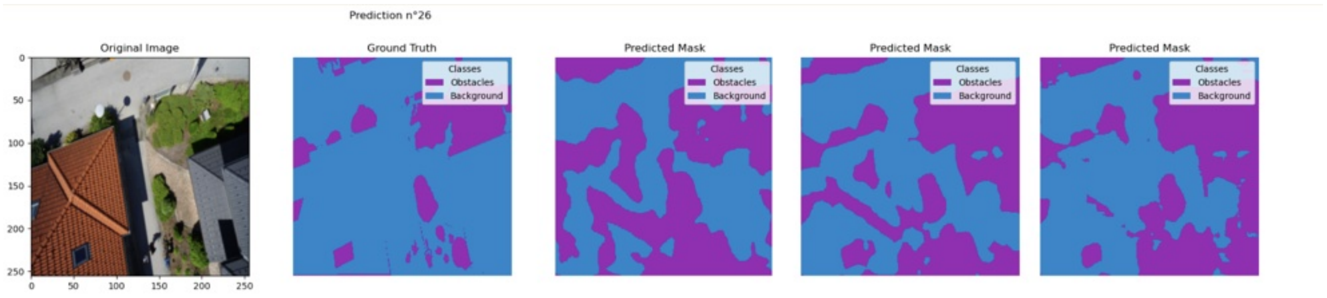
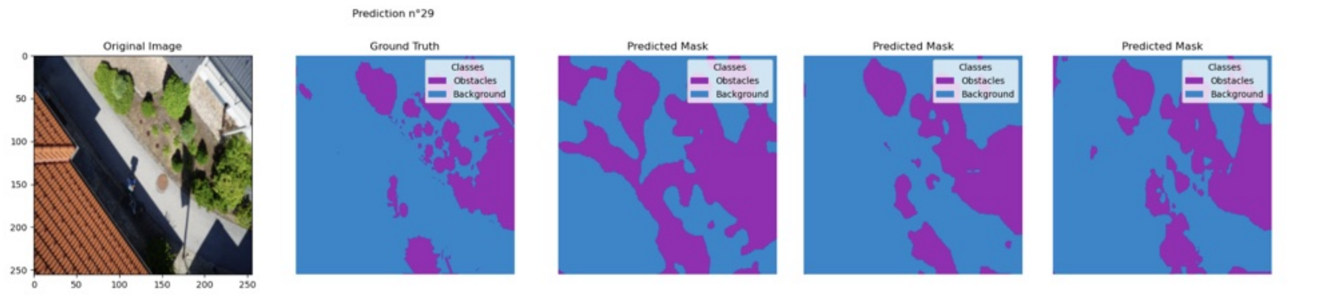


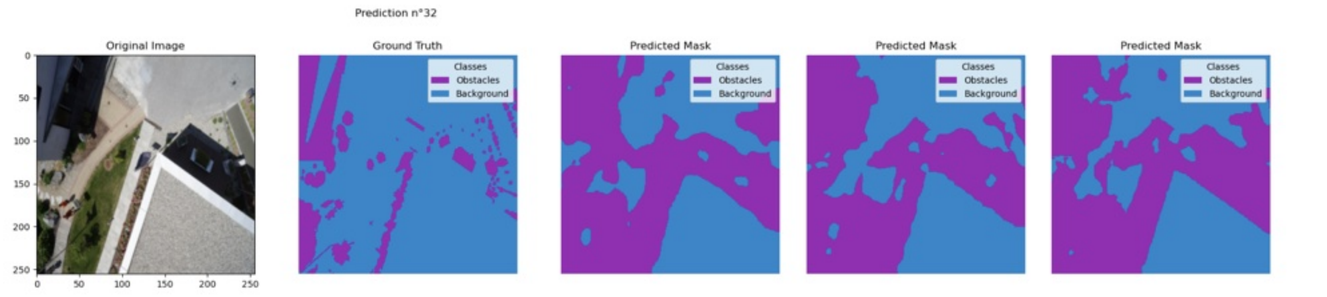
FIGURE 9 – Best predictions

Worst Predictions









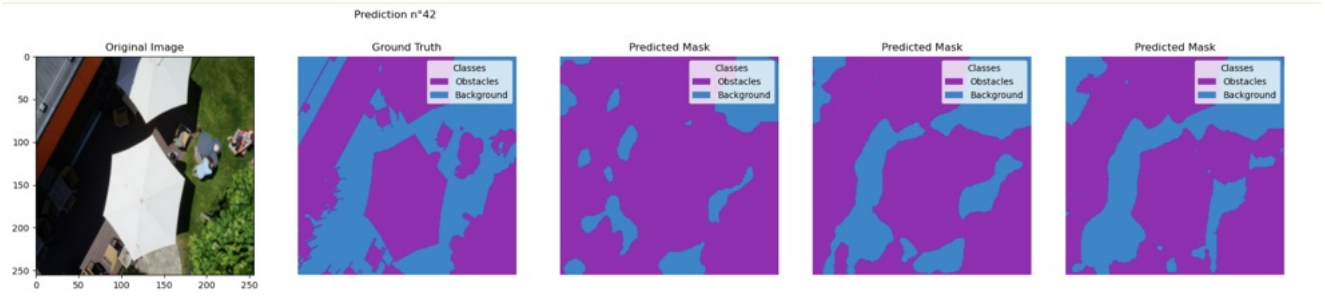


FIGURE 10 – Worst predictions

It is interesting to note that for worst predictions the model with 1000 augmented images provides better results than 500 but for the best images it is the opposite. There is no evident explanation for this situation but a model trained with fewer augmented images might overfit to the limited dataset, leading to better performance on the best predictions that align closely with the training data. However, this overfitting can cause the model to struggle with harder, less-representative cases, leading to poorer performance on the worst predictions. Conversely, the model trained with 1000 augmented images is less prone to overfitting and generalizes better, resulting in improved performance on difficult cases but potentially sacrificing some precision on the easier cases. However, we have to keep in mind that the model is not overfitting on the whole dataset, this situation can be considered as a "local" overfitting otherwise we would see a increasing validation loss curve.

SegFormer

We follow the same structure as the U-Net to train the SegFormer. However, we stick with 500 augmented images max as 1000 images is increasingly the computation time too much compared to the improvement of the results. We start without data augmentation, with optimizer Adam and Reduce learning rate on plateau.

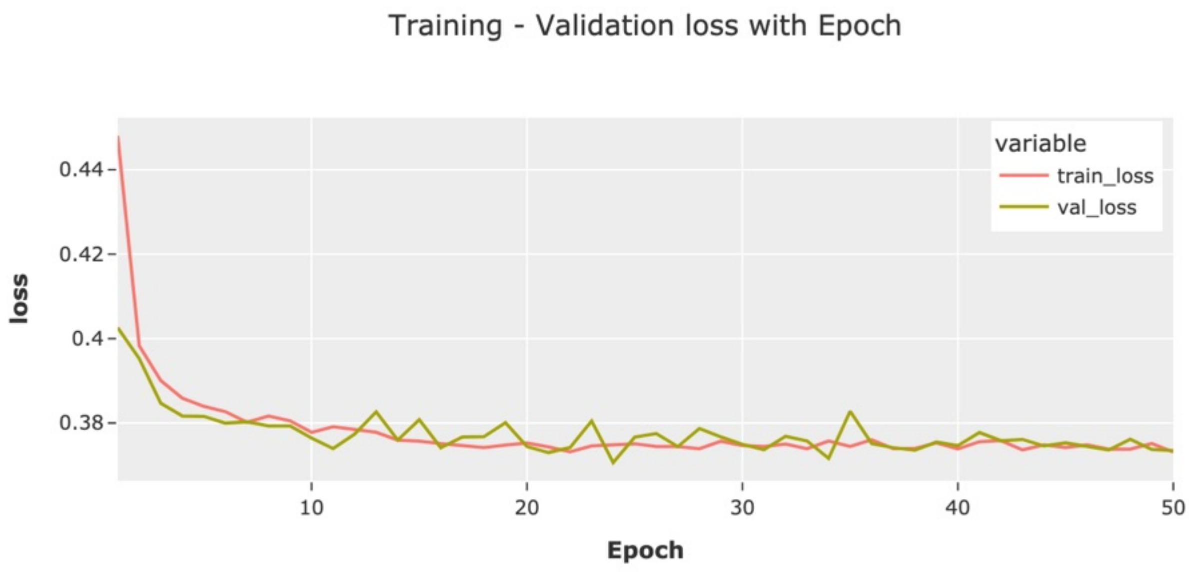


FIGURE 11 – Binary SegFormer loss curve

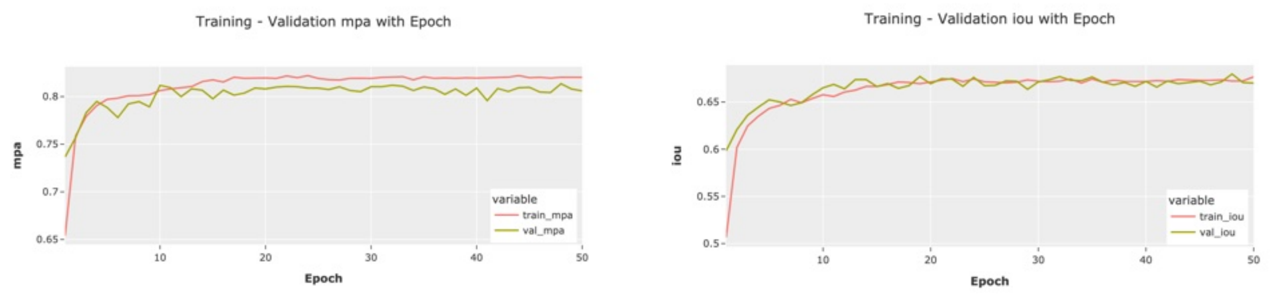


FIGURE 12 – MPA and IOU of the binary SegFormer

Compared to U-Net, we see that the validation loss of the SegFormer is more variable. This might be because the SegFormer architecture, which relies on self-attention mechanisms. These mechanisms are highly sensitive to the input data distribution and can amplify variations during training, especially when the dataset is small as in our situation. Additionally, transformers are more prone to overfitting on smaller datasets because of their high capacity and reliance on long-range dependencies.

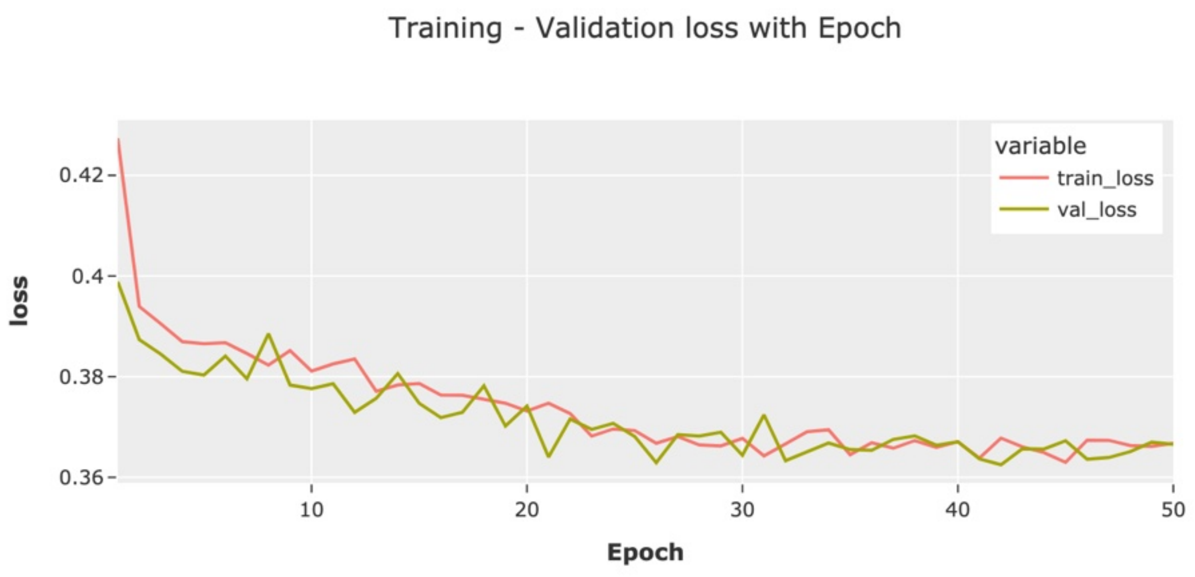
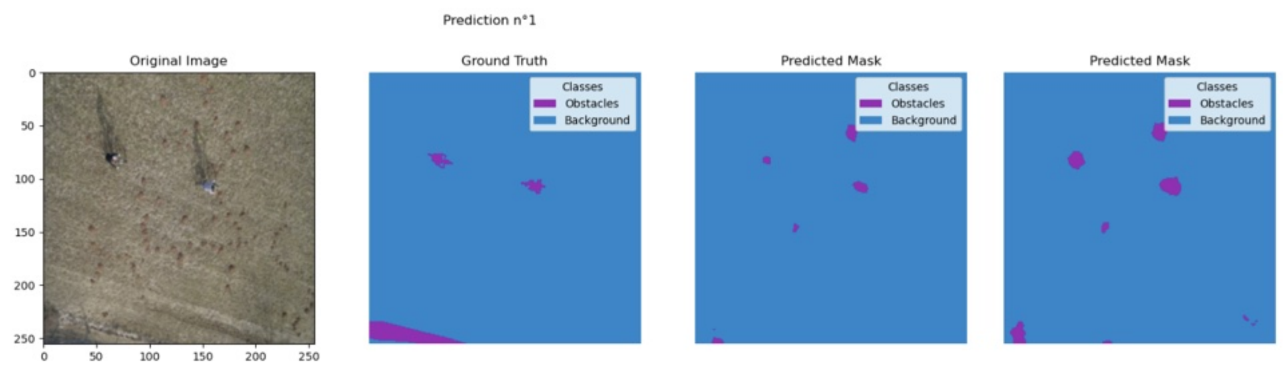


FIGURE 13 – Binary SegFormer loss curve with 500 augmented images

Best Predictions



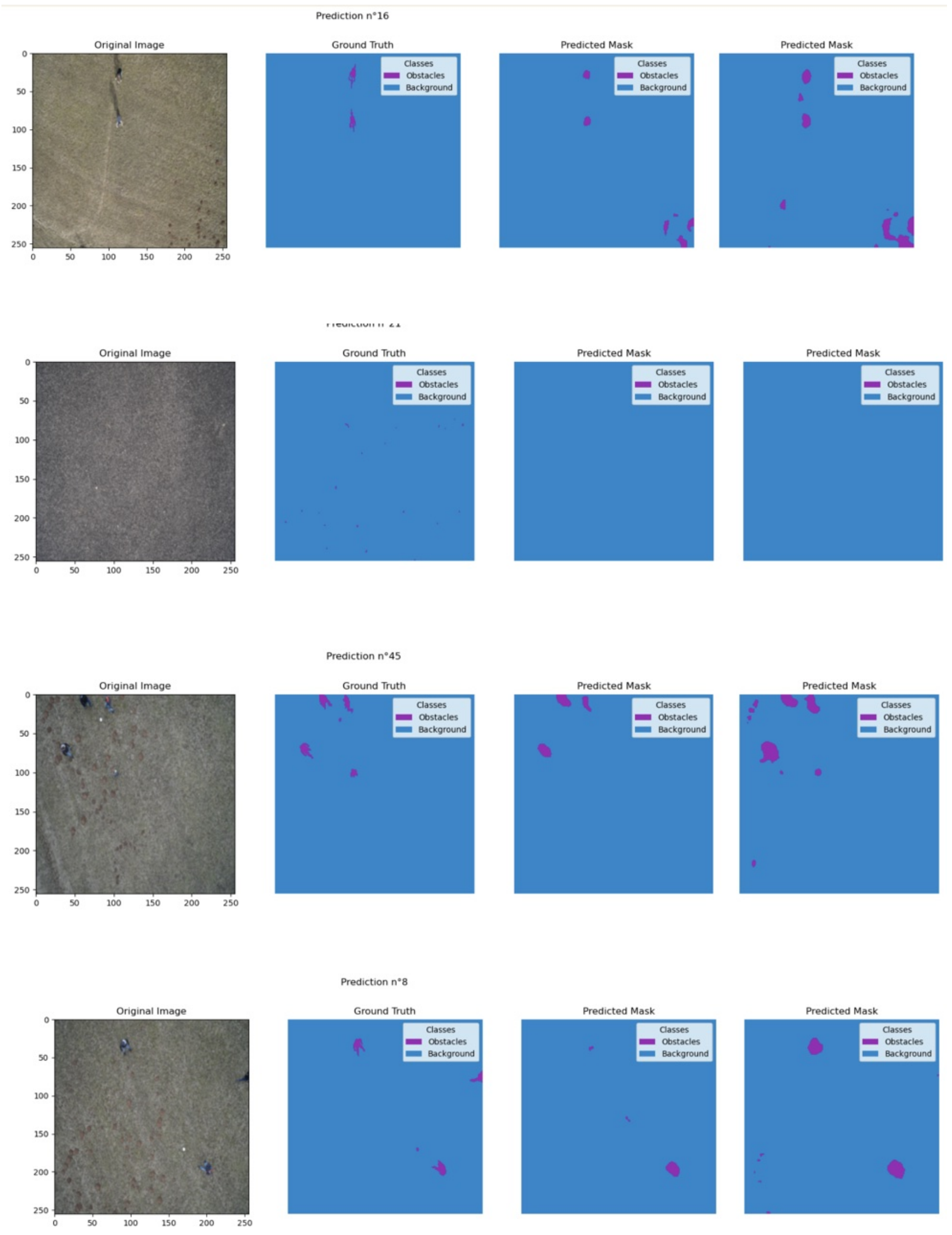
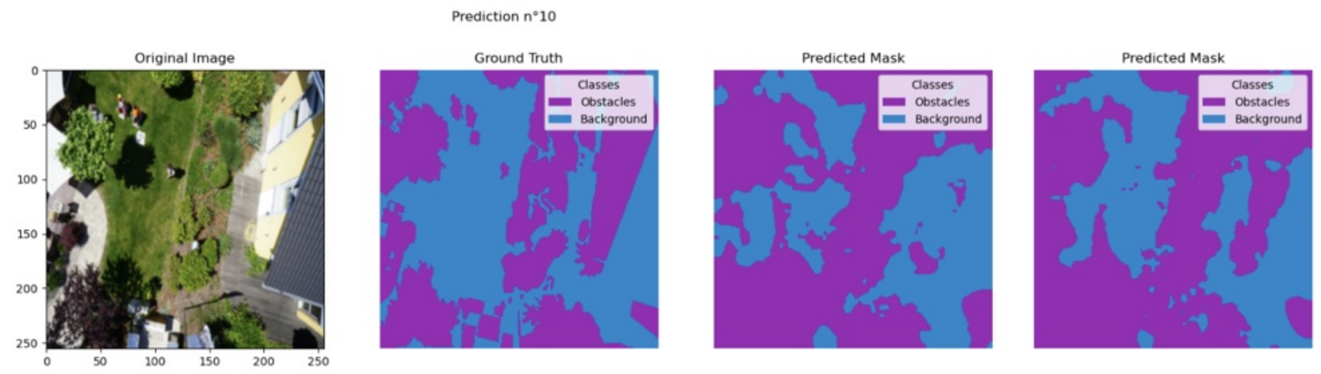
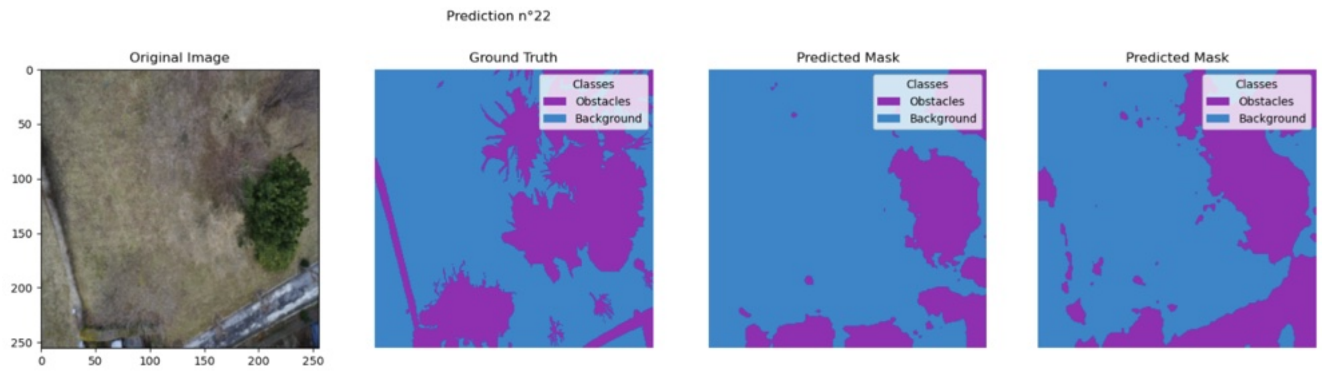
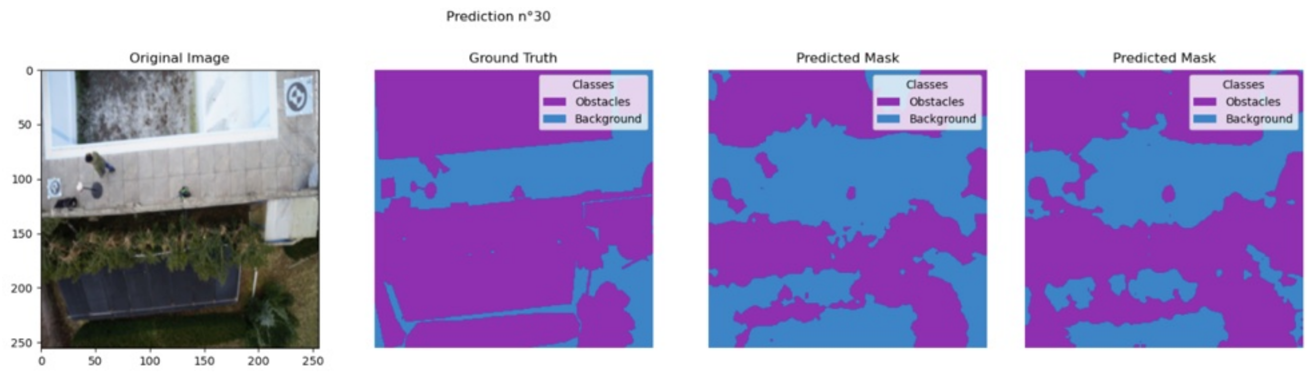


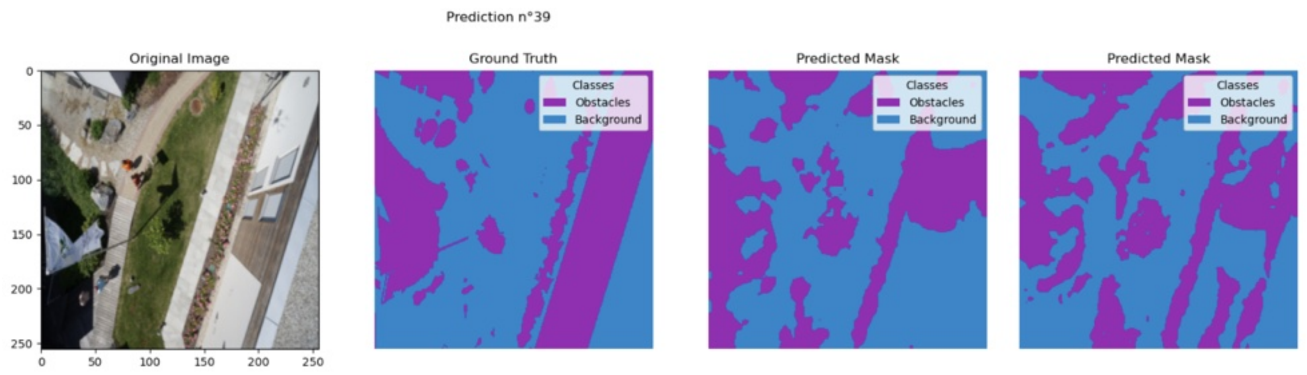
FIGURE 14 – Best predictions

Worst Predictions









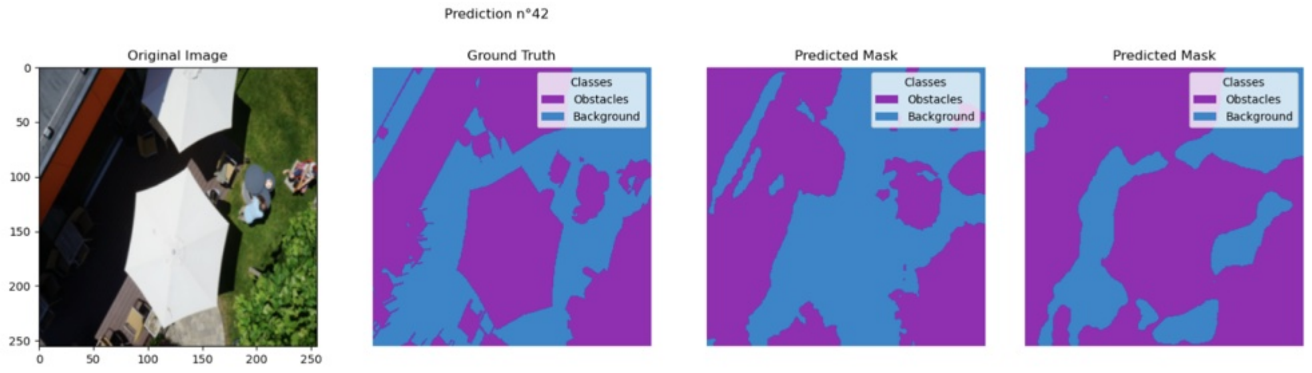


FIGURE 15 – Worst predictions

First, we see that almost all of the best predictions tend to occur in images containing mostly one class because these regions are simpler for the model to classify. The model can predict a single class when there is little ambiguity, leading to higher accuracy. This is especially true in cases of class imbalance, where the model may be biased toward the dominant class.

Then, without surprise most of the best and worst predictions are shared with U-Net. However we notice some differences on the masks. SegFormer provides a sharper mask with well defined transitions between classes while U-Net masks are coarser. This result is consistent with what we expected from the two models so we can combine them to see if we get a better model.

2.2 UFormer

We train and test the UFormer with the U-Net and the SegFormer trained with 500 augmented images. Let's have a look at the results.

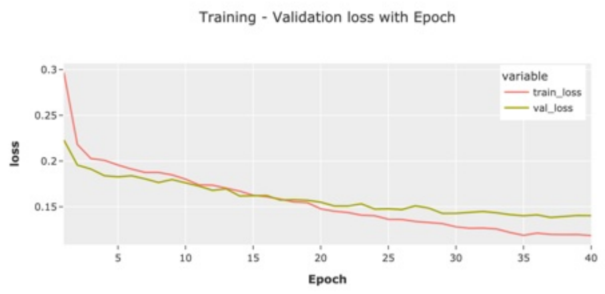


FIGURE 16 – Loss of the UFormer

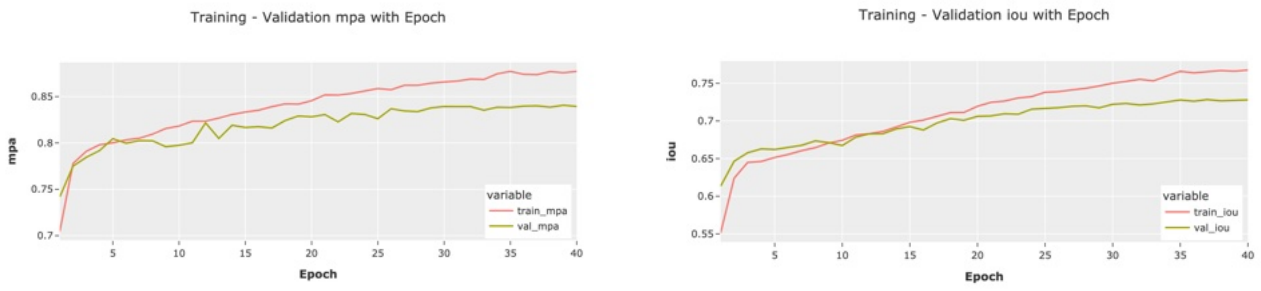
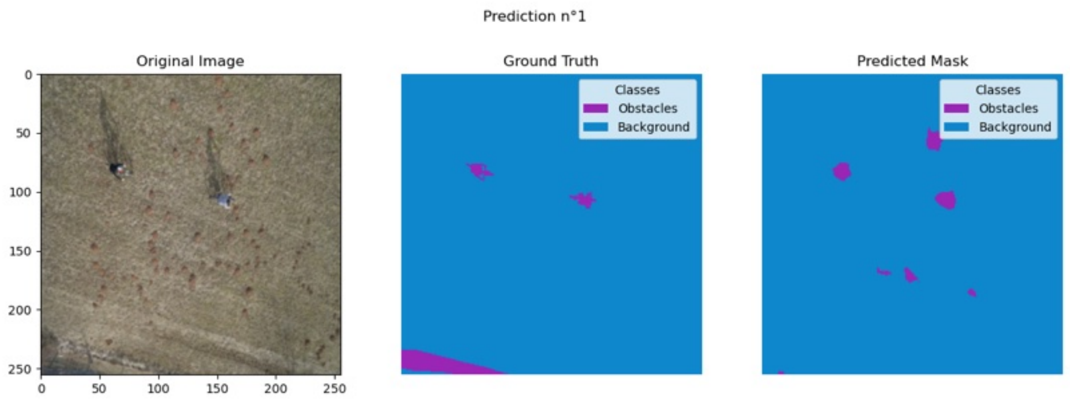
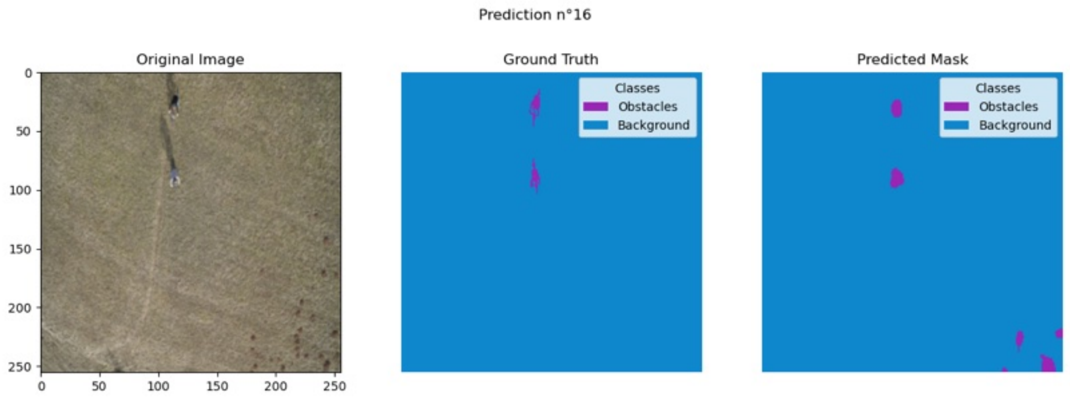


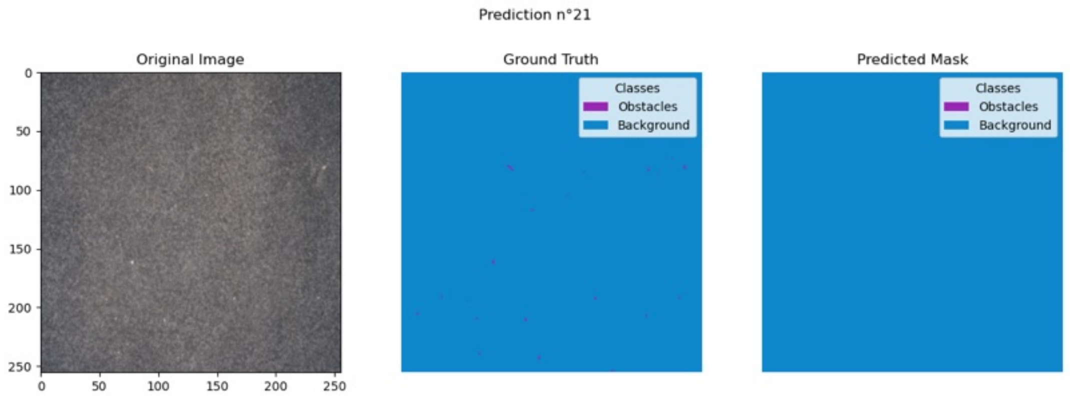
FIGURE 17 – MPA and IOU of the binary UFormer

Compared to the previously studied model, the main difference is that for each metric, as epoch increases the difference between training and validation is getting higher indicating potential overfitting. As UFormer is combining parameters of both U-Net and SegFormer, it becomes heavily parametrized leading to an increased sensitivity to overfitting.

Best Predictions







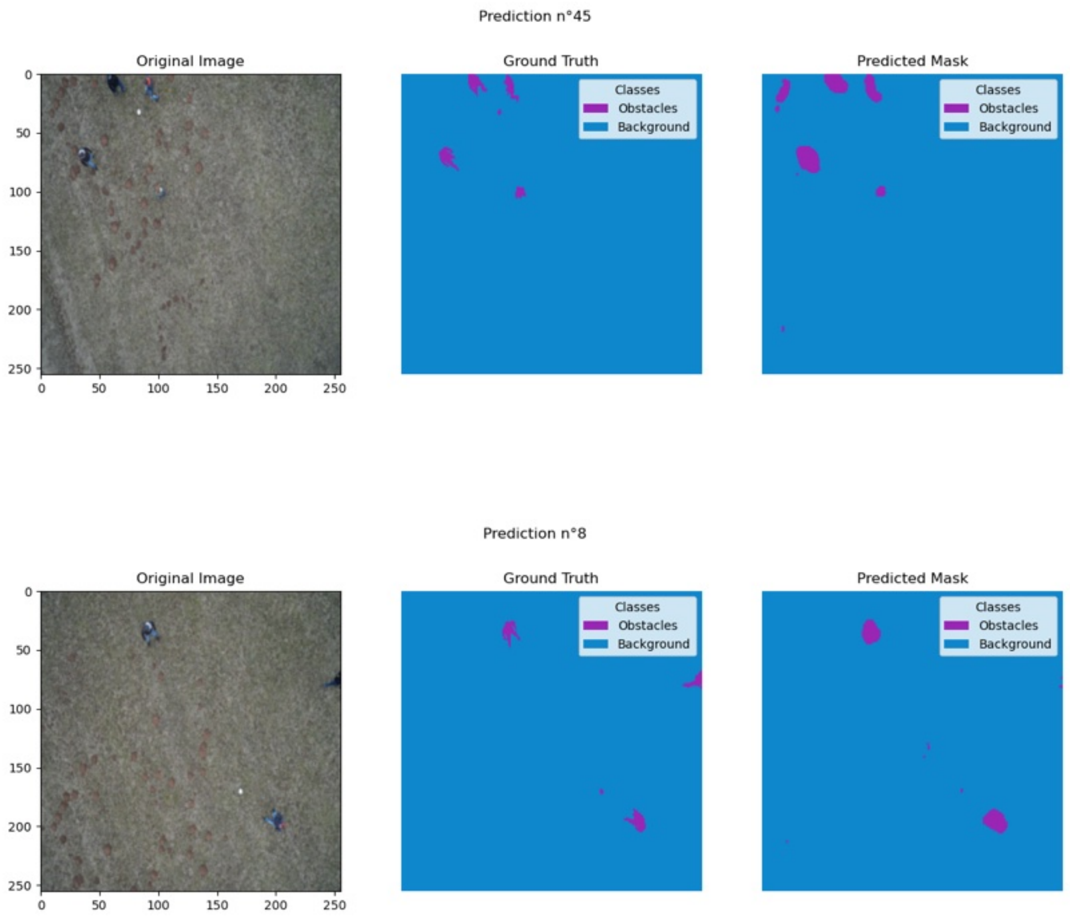
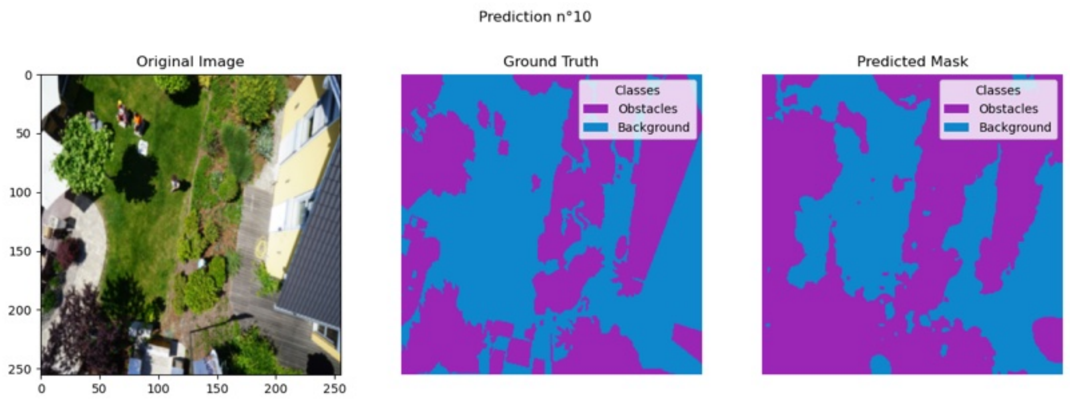
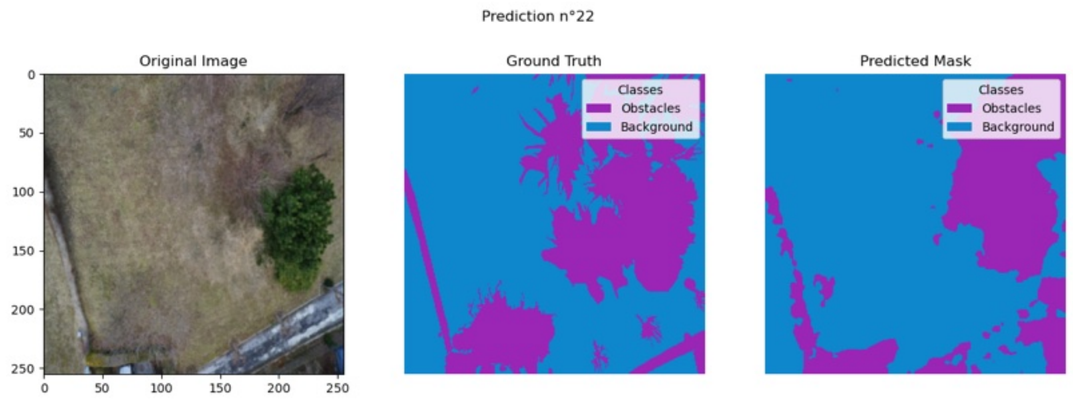
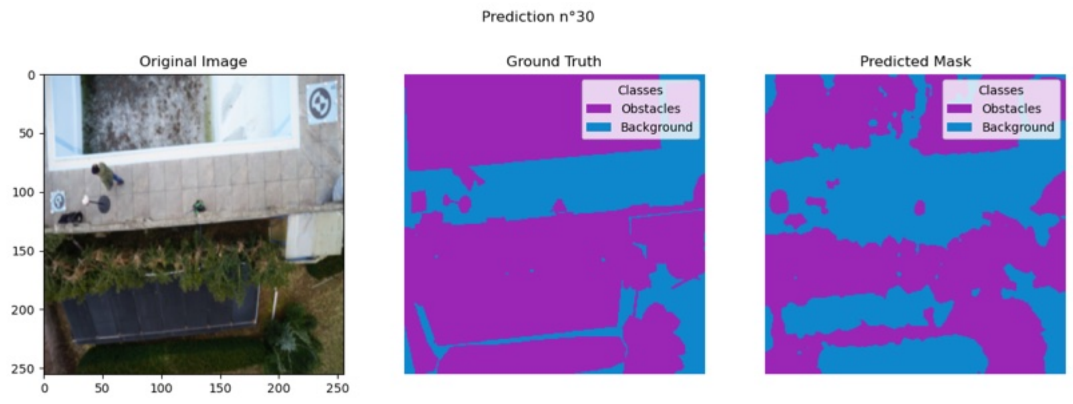


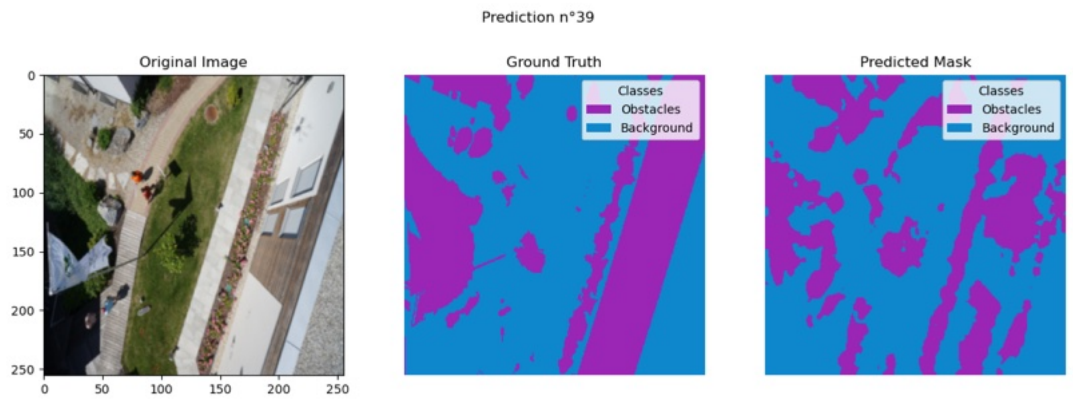
FIGURE 18 – Best predictions

Worst Predictions




Prediction n°22
Original Image
Ground Truth
Predicted Mask
Classes
Obstacles
Background
0
50
100
150
200
250


Prediction n°30
Original Image
Ground Truth
Predicted Mask
Classes
Obstacles
Background
0
50
100
150
200
250


Prediction n°39
Original Image
Ground Truth
Predicted Mask
Classes
Obstacles
Background
0
50
100
150
200
250

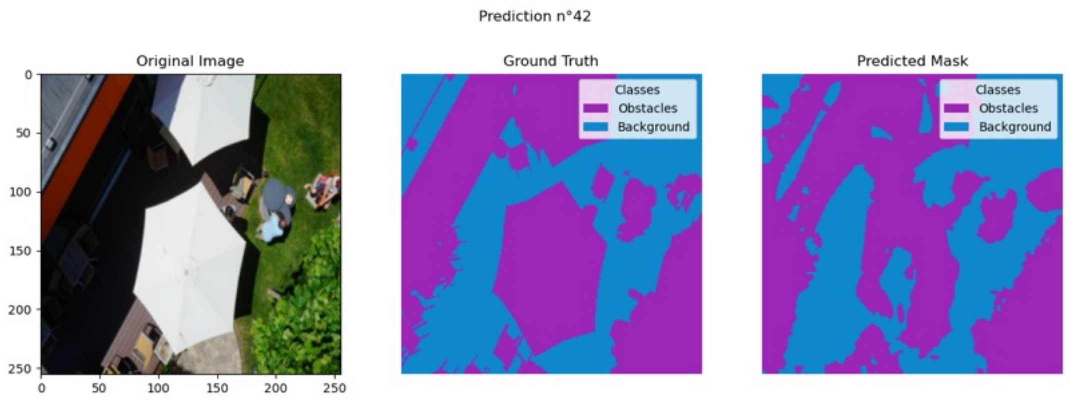


Figure 19 – Worst predictions

To compare those results to both U-Net and SegFormer we will focus on the metrics. But visually, we notice that fine details are better captured on worst images but to a certain limit visible on bikers on best images. As always, it seems that there is a trade-off between capturing fine detail and global information. The parasol is also intersting, it was not captured by SegFormer but captured by U-Net leading to a half detected parasol in UFormer.

Final Results

As briefly explained before, we believe that adding 500 augmented images is a good trade-off between computation time and model performances. This is the reason why we decided to stick with aroung 500 augmented images rather than 1000 or above.

Model (500 Augmented images)	Test Loss	Test mIoU	Test mPA	GPU(s)	Time
SegFormer	0.373	0.826	0.870	1	13 min
U-Net	0.345	0.783	0.842	1	31 min
UFormer	0.146	0.853	0.890	X	X

Table 1 – Results for binary segmentation models.

UFormer outperforms the other models for binary segmentation, it provides the best results. This can be attributed to its architecture, which is specifically designed to capture both local and global information efficiently as discussed before.

Other various tuning

Separable Convolution

To reduce computational cost and time, we tried to implement separable convolution for both encoder-decoder and only decoder. Separable convolutions uses the assumption that spatial and channel-wise information can be decoupled to reduce computational cost. As a results, we decreased calculation time by 20% on average but the results and especially the loss curve were worst. This might be because drone images are capturing complex environments with varied textures, such as buildings, roads, vegetation, and water. These patterns require the model to understand spatial relationships across multiple channels and regions while the separable convolution is removing this information. Another reason can be the lack of data, as separable convolution is using less information on each image we may need more images to get the same result as without separable convolution.

Gricad

For this project we used Gricad a lot in various situation :

— We coded on our personal computers then we send the code to gricad via rsync.

— We used bigfoot and the repository bettik for training and saving data regarding the models.

— We used only 1 GPU for the binary segmentation but 4 for the multiclass.

— We did not notice any big issues with gricad, everything worked as wished and we now know how to use a cluster, work remotely and take into consideration that we are not alone and need to adapt when we use the cluster.

Conclusion

In this project, we explored the use of Deep Learning for semantic image segmentation, with a focus on datasets containing two and five classes. We faced several challenges that impacted the overall performance of the models.

A key limitation was the size of the dataset, with only 400 images available. While data augmentation techniques helped solve some of the issues by generating a more diverse set of training images, the model's ability to generalize remained constrained. Increasing the dataset size, either through synthetic data generation or by obtaining additional labeled data, would likely improve the model's robustness and ability to generalize to unseen data.

Another challenge was the complexity of the images themselves, which often involved unique patterns and fine details. Additionally, masks were sometimes corrupted or false misleading the training of the model.

The final model, the UFormer can be improved by optimizing the weights of the average between the two models. Additionally, although Tversky loss showed potential in handling class imbalances, the specific choice of the hyperparameters α and β remained a challenge. Further experimentation with these parameters, perhaps guided by a more detailed analysis of the class distributions, could provide better results.

In conclusion, while the models performed reasonably well, the limitations of data size, model complexity, and class imbalance highlight areas for improvement. Future work should focus on expanding the dataset, refining loss functions, and experimenting with more advanced architectures to achieve better segmentation performance.

Work Balance

Lorenzo	Louis
50	50

Références

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv :1505.04597*, 2015.

[2] Enze Xie, Yiming Chen, Zhichao Zhang, Yuchen Yu, Lin Zhang, Zheng Li, Zicheng Li, Yuxiao Xu, Yixuan Li, and Yi Li. Segformer : Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv :2105.15203*, 2021.