Internship Report - Semantic Segmentation State of The Art for Ultra High Resolution UAV Imagery

Pierre Lague

August 12, 2022

Contents

Abstract

Airborne sensor systems allow observations of our planet over large areas that may be difficult to access (such as mountain environments). Technological advances are resulting in smaller, lighter systems and allow more accurate observations. Such observations are commonly acquired by UAV (unmanned aerial vehicles) and consists of color, multi-spectral (also covering the invisible spectrum via infrared wavelengths), and 3D point clouds images. These observations allow various fields of work such as in ecology (e.g biomass interaction) and computer science (e.g semantic segmentation). In this report we will try to establish a state of the art of semantic segmentation methods applied to ultra high resolution imagery of mountain sides in order to classify various plant species.

This work is a side task of the SixP project.

1 Introduction to the project

SixP is a research project funded by the French National Research Agency (ANR) from 2020 to 2023. It focuses on the ecology of plant communities developing on former mining sites, whose soils are rich in metals. The study of interactions between plants in these particular environments is at the heart of the project. Studies are also developed to set up innovative methods of characterization and identification of the vegetation via remote sensing and artificial intelligence.

The major objectives of this project are therefore to:

- Characterize the variation of plant interactions along metal gradients in soils
- Identify the multiple constraints for vegetation development in metal-rich environments and their respective consequences on plant interactions
- Combine remote sensing approaches and in situ species identification via artificial intelligence methods to allow mapping of species present in sites of interest

The last objective is the one we're going to work on. Our goal being to establish a state of the art on the semantic segmentation methods that can be used for our data (UAV imagery) and that achieve significant scores.

2 Introduction to Semantic Segmentation

One of the most difficult problems in computer vision has been image segmentation. Image segmentation is different from images classification or object recognition because it is not necessary to know what the visual concepts or objects are beforehand. In contrast to common pixel-based methods, CNN allow for an efficient analysis of image textures, i.e., the contextual signal of multiple neighbouring pixels. CNN have been initially designed for image categorization tasks. A major factor of common CNN architectures for identifying such features are multiple and sequenced pooling layer operator that aggregate the feature maps derived from convolutional layers to a reduced spatial scale. This

increases the robustness and efficiency of the network. The last layer of the network will contain the information whether a feature that is indicative for the target class was visible anywhere in the image or not. However, these models indicate if a class is present or not in an image. We want to know precisely where the class is located amongst other classes. In order to extract context and become spatially aware, the networks have to be fully convolutionnal.

The proven most efficient model in spatial semantic segmentation is the U-net. After being tested on multiple benchmark dataset, it has become the reference in terms of fine-grained semantic segmentation. Even if it's supposedly very efficient for land covering segmentation, the lack of accurate and large reference observations make the analysis very difficult (field operations very hard to organize as well as a high logistic effort). Recently, deep learning based methods have achieved significant progress in semantic segmentation. However, performances tend to drop when the input images differ from the training input images on the trained models. For example, a model trained on images of cities will not perform as well as on images of mountain sides.

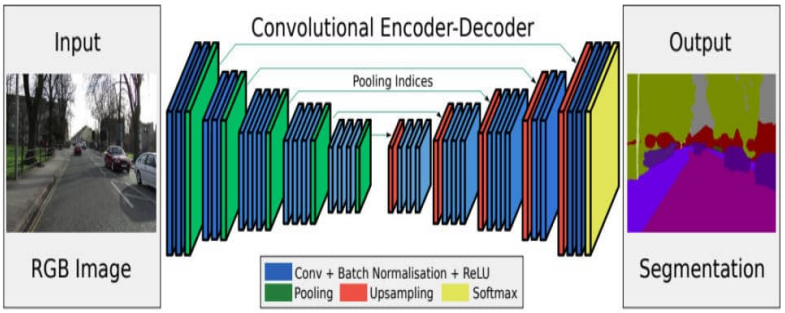

Figure 1: Semantic Segmentation using a U-net architecture

Other models were developed before the U-net, which also seem like good candidates for land covering segmentation such as LinkNet, FPN and PSPNet. We aim to compare each one of them as precisely as possible to find the most suitable model to use in the 6P Project.

3 Related Work

In 2012, the AlexNet [KSH12] won the ILSVRC contest, which is a key event in deep learning. Since then, DCNNs (Deep Convolutional Neural Networks) have followed a lot of development .VGG [SZ14], ResNet [He+16] have been proposed one after another. These frameworks are usually treated as feature extractor and play an import role computer vision tasks, such as object detection [Ren+15], semantic segmentation and scene understanding [Zho+16], etc.

Semantic segmentation is a significant branch in computer vision. There are

a considerable number of works focusing on the remote sensing imagery. With the development of deep learning on remote sensing images, image-to-image segmentation becomes the mainstream. Sherrah and Jamie [She16] proposed a deep FCN with no down-sampling to infer a full-resolution label map. Their method employs the strategy of the dilated convolution used in the DeepLab [Che+18] architecture, which uses dilated kernel to enlarge the size of convolution output (very heavy model). Marmanis et al. [Mar+18] embedded boundary detection to the SegNet encoder-decoder architecture. The boundary detection significantly improves semantic segmentation performance with extra model complexity and proves itself very efficient when classes are overlapping. Kampffmeyer et al. [KSJ16] focused on small object segmentation through measuring the uncertainty for DCNNs. This approach achieves high overall accuracy as well as good accuracy for small objects. When it comes to UAV high resolution imagery Yu et al. [Yu+22] have enhanced the advanced version of the unet model (Unet++) and compared it to other architectures combined with different encoders.

4 The Data

The data for this project was collected by a private company called "L'Avion Jaune". It took pictures from a drone above mountain sides. In total we have four study sites : Talus, Chichoue_Bas, Chichoue_Haut, Chichoue_Milieu_Bas, Chichoue_Milieu_Haut. During this study, we will be only focused on the Talus site. The images are 224*224 pixels and each site has it's unique labeling indexation (0 to X for X classes, some classes can appear in multiple sites and class 1 of site A may differ from class 1 of site B). For each RGB Image (Figure 2) a ground truth is associated (Figure 3). Each different color in the ground truth represents a specific species/class.

Figure 2: RGB Image - Sample from the test test of the Talus site

Figure 3: Ground Truth Image - Sample from the test test of the Talus site

5 Methods and Tools

5.0.1 Methods

This report is part of a bigger overall process shown in Figure 4. It focuses on comparing different models and backbones in order to establish which model and hyperparameters are best suited for our images. The data processing part where we create our datasets from ortho Images as well as the labeling process was done by Florent Guiotte. The rest of the process is the heart of this report. After having generated our datasets composed of 224*224 RGB images and labels, we compare different combinations of trained models and backbones using cross validation to see which one has the best results. Finally we test the combinations and observe different metrics that allow us to evaluate our models. We did not perform any transfer learning during the project for it did not significantly improve the models.

5.0.2 Tools

In order to test out different deep learning network architectures, we had to take into account the time it would take to complete the analysis and my abilities to manipulate such networks. We found a library called "segmentation_models_pytorch" (SMP) which allows a very fast use and implementation of various segmentation models and backbones. This library also proposes different evaluation metrics, however we decided to compute them on or own using other libraries. We also used some more standard librairies : pytorch, numpy, sklearn etc.

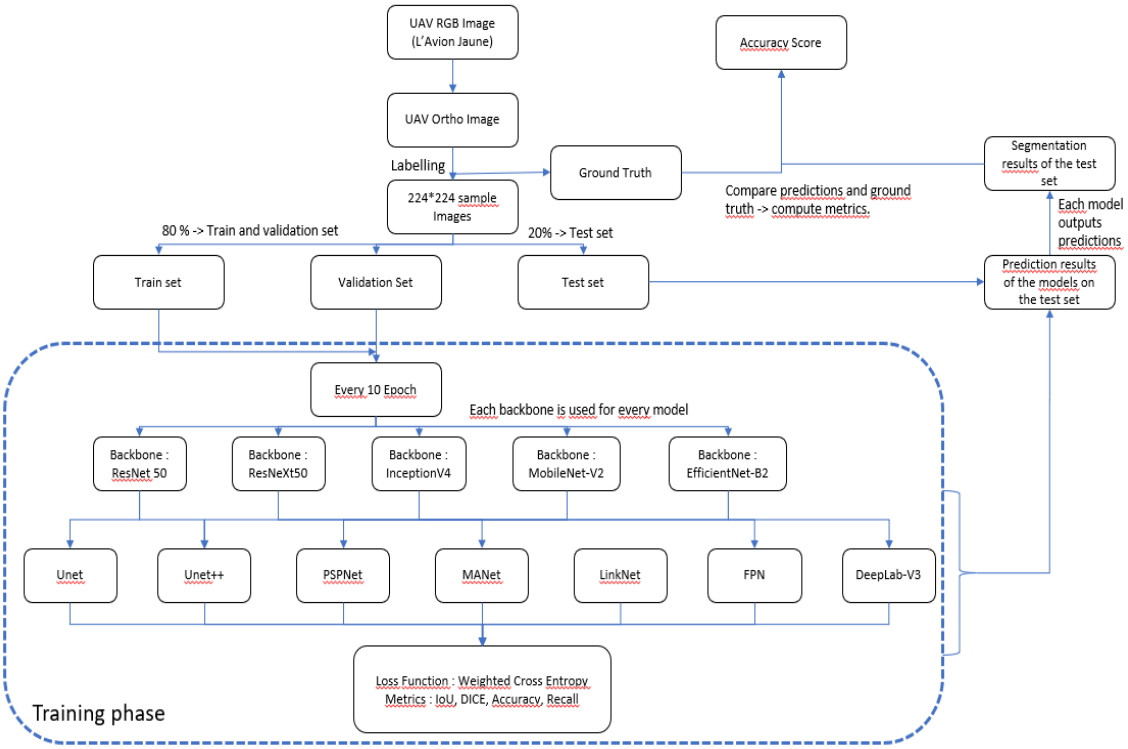

Figure 4: Flow Chart of the project

6 Common Evaluation Metrics

In order to evaluate a segmentation model and it's ability to properly classify pixels, the evaluation indicators are precision (accuracy), IoU (intersection over union), F1 Score (Dice) and recall. Precision refers to the percentage of pixels that were correctly classified in the output result. IoU refers to the overlap rate between the generated output frame and the original labeled frame (i.e intersection over union). It's relevancy in semantic segmentation comes from the fact that each class has a defined area and that it is possible to compute the difference between the ground truth and the prediction. Recall rate refers to the proportion of correctly matched pixels in the ground truth. F1 Score is the average of precision and recall.

- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $IoU = \frac{TP}{TP+FP+FN}$
- $F1 = \frac{Precision*Recall}{Precision+Recall} * 2$

In order to synthesize our results we also computed the Mean Dice Score and the Mean IoU for all classes. This way we can associate a global understandable score to an architecture.

6.1 Weighted Cross Entropy

A common issue with "real world" datasets, specially the ones with no state of the art, is the imbalance of observed object classes. In our case, the objects are plants that are classified in multiple species from one site to another. However, one or multiple of these species may be over represented in the dataset which creates "class imbalance". This problem can have a detrimental effect on classification performance of neural networks trained on such datasets. Methods overcoming class imbalance can be divided into two main categories. The first category are **sampling-based methods** that operate directly on a dataset with the aim to balance its class distribution. In their basic versions, the dataset is balanced by increasing the number of instances from "minority" classes and by decreasing the number of instances from "majority" classes, respectively. The second category are **algorithm-based methods**. They make use of cost-based training and decision thresholding. The idea behind these strategies is to assign different costs to classification mistakes for different classes. In this project we adapt to this constraint via an algorithm-level approach called **weighted cross entropy**.

There are two ways of computing the weights that will be used in the cross entropy loss function. The first one is to compute the weights outside of the training phase. In this case, the weights are computed using the inverse-frequency method over the whole training set and won't change during the inference process. The other way is to compute the weights for each batch during the epochs. That way, the weights are adapted to the batch and insure good class balance for the inference phase. Since we don't have a very large dataset, the computing cost is not too high.

After testing these two methods individually on the Talus site using the Unet with the ResNet-50 backbone, we compared the overall metrics and decided to choose the second method that provides dynamic weights throughout the inference phase.

7 Results

For each training and testing sessions, we store the evolution of the metrics throughout the session in tensorboards in order to visualize predictions and graphs. The average time of training varied from 1h for the light-weight models and encoders (Mobile-Net V2 backbone) to 2h for the heavy-weights models and encoders (Inception backbone).

7.1 First Testing Phase : Talus Site

The first step in testing our models was to test them on the Talus Site. After testing all of our models, we selected the best configurations for the study site (a configuration being a pair {network, encoder}). The following observations were noted :

- The need for data is important and will most likely improve the models
- The fact that there are multiple classes affects the model's performances

- The quality of the labeling affects the model's ability to extract context from images and provide good segmentation

Seeing the results, we quickly realised that some models were overfitting during the training sessions. The overftit began around epoch 30, so we reduced the numbers of epoch from 100 to 30 in order to gain some time and only save fitted checkpoints. The following tables resume the results we obtained on the Talus site. The best configuration will be highlighted in bold, the caption specifies the encoder that was used.

Network	Accuracy	Mean IoU	Mean DICE
Unet	17.86%	0.0720	0.1170
Unet++	17.49%	0.0796	0.1282
PSPNet	20.70%	0.0716	0.1186
Manet	**21.94%**	**0.0972**	**0.1444**
LinkNet	18.78%	0.0837	0.1331
FPN	19.50%	0.0771	0.1193
DeepLab-V3	15.23%	0.0380	0.0662

Table 1: ResNet50 - Talus - 9 classes

Network	Accuracy	Mean IoU	Mean DICE
Unet	**19.99%**	**0.0864**	**0.1345**
Unet++	16.87%	0.0782	0.1259
PSPNet	18.24%	0.0393	0.0711
Manet	14.66%	0.0660	0.1103
LinkNet	19.85%	0.0716	0.1123
FPN	18.52%	0.0726	0.1085
DeepLab-V3	12.70%	0.0463	0.0803

Table 2: ResNeXt50 - Talus - 9 classes

Network	Accuracy	Mean IoU	Mean DICE
Unet	10.12%	0.0376	0.0683
Unet++	8.54%	0.0355	0.0647
PSPNet	**23.49%**	**0.0760**	**0.1233**
Manet	9.42%	0.0410	0.0739
LinkNet	**23.43%**	**0.0813**	**0.1266**
FPN	12.14%	0.0472	0.0832
DeepLab-V3	6.72%	0.0263	0.0492

Table 3: EfficientNetB2 - Talus - 9 classes

Network	Accuracy	Mean IoU	Mean DICE
Unet	18.02%	0.0798	0.1242
Unet++	**25.15%**	**0.0965**	**0.1429**
PSPNet	**25.49**%	**0.0731**	**0.1136**
Manet	20.85%	0.0889	0.1313
LinkNet	13.77%	0.0706	0.1180
FPN	19.79%	0.0867	0.1349
DeepLab-V3	X	X	X

Table 4: InceptionV4 - Talus - 9 classes

Network	Accuracy	Mean IoU	Mean DICE
Unet	16.56%	0.0777	0.1254
Unet++	16.39%	0.0654	0.1056
PSPNet	**29.86**%	**0.0792**	**0.1181**
Manet	16.98%	0.0731	0.1225
LinkNet	8.46%	0.0313	0.0582
FPN	9.94%	0.0343	0.0633
DeepLab-V3	6.74%	0.0287	0.0520

Table 5: MobileNetV2 - Talus - 9 classes

7.2 Comments and Analysis

Overall, the metrics show very poor results. This is due to the fact that there is a major problem with our labeled data. Since our masks are not pixel-accurate, the different models are not able to extract precise information from the context of an image. Regarding the highest scores for each encoder, the accuracy is always lower than 30%, the mean IoU and mean DICE shows that only very little of the prediction was located onto the mask.

Given the quality of our labeled data, the other problem is to know whether our metrics are reliable or not. Since our setup isn't "by the book" according to the standard state of the art projects (accurate masks, important number of data ...), we can't take standard pixelwise metrics for semantic segmentation and apply it to non pixel-accurate masks. This is why, during the mid-intership meeting with Javiera Castillo, Sebastien Lefevre and myself, we decided to orient our work towards finding a reliable and accurate metric that took into account the quality of our labeled data.

8 Finding an Optimal Metric

8.1 Object Detection Evaluation

Since we were not able to find segmentation models that returned consistent results and that our labeled data is lacking precision, we looked into the matter of object detection. The metrics used in Object Detection are called the same as in Semantic Segmentation, however, they do not take into account each pixels of the ground truth and the prediction but a context and the presence of an object or not. As in Semantic Segmentation, we need to compute the metrics from the confusion matrix (TP, FP, TN, FN) in order to compute our evaluation metrics (IoU, Recall, Precision ...).

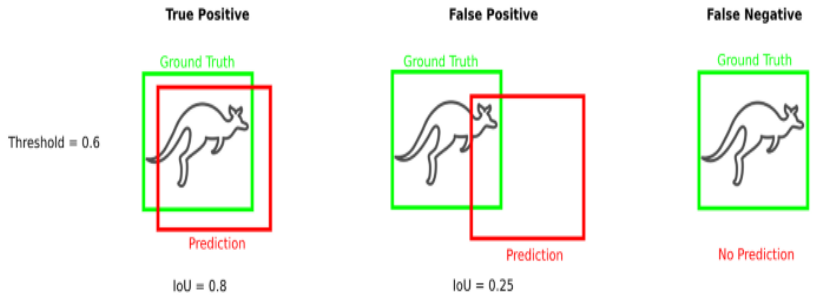

Figure 5: Representation of the metrics with object detection

In our case however, we won't be using bounding boxes but the global shapes of our labeled data and our predictions.

The following pseudo-code illustrates the procedure :

Algorithm 1 Commpute TP, FP, FN

for $iteration = 1, 2, \ldots K$ **do** ▷ loop for each class
    TP, FP, FN = 0
    $P = (Pred == i)$ ▷ Get predicted binary mask for class i
    $M = (Mask == i)$ ▷ Get binary ground truth mask for class i
    $CC = sklearn.measure.label(M)$ ▷ label all the connected regions of M
    for c in CC **do**
        $intersection = Pred * c$ ▷ Get predicted pixels within c
        if $\frac{sum(intersection}{sum(c)} > \lambda$ **then** ▷ Intersection is an array of binary masks
            $TP+ = 1$
        else
            $FN+ = 1$
        end if
    end for
    $B = P - (M * P)$ ▷ Get all pixels predicted as i, but not in the ground truth
    $CCB = sklearn.measure.label(B)$ ▷ Label all connected regions of prediction for class i
    for c in CCB **do**
        if $sum(c) >= \alpha$ **then**
            $FP+ = 1$
        end if
    end for
end for

Here is an example that illustrates the expected results (6).

Figure 6: From left to right : mask, prediction and mask != prediction.

With the previous mask and prediction as inputs for our algorithm, we have the following results (6) :

X	White Class
TP	2
FP	2
FN	0

Table 6: Counting TP, FP and FN at threshold $\lambda = 0.8$

In fact, two of the dummy predictions have at least 80% of their area on the mask, so they are counted as true positives. It's not the case for the other two, that means they are counted as false positives. This allows us to have a reliable number of TP, FN and FP in order to compute precision and recall for each of our classes.

8.2 Results With the New Method

After including this step into the inference phase of our *test.py* script, we implemented a way of plotting the precision-recall curve and compute the Area Under the Roc Curve (eg. figure7). Each point in the curve is computed for a different threshold for the IoU (from 0.1 to 0.7 with a 0.05 step).

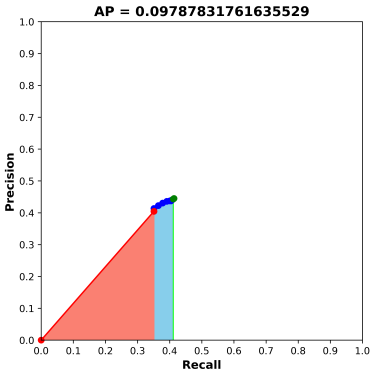

Figure 7: Precision-Recall Curve for Class 1, Average Precision is 0.097

The overall results we got during the test phase using the new method shows that our classification is in fact of poor quality. Only one class is detected by the model and correctly predicted. After computing the precision and recall for each classes, we've computed the Mean Average Precision using the "all-points interpolation method" : **mAP = 0.009787867343884947**. Because of the poor classification, only few predictions were counted as True Positives, almost all of the other predictions were counted as False Negatives. The problem here is the spatial location of the prediction, being completely out of the mask's bounds.

8.2.1 Comments

It's much more likely that object detection will work better with the quality of our labeling and the amount of data we have. This requires a first step of pre-processing in order to transform the masks into bounding boxes and then to train object detection models (eg: YOLO) on our data. The main problem would change from "Which species of plants are on the image and where are they ?" to "Where are the plants?" for a less granular problem.

9 Conclusion of the study

During this analysis, we've adopted different approaches on a semantic segmentation problem. The first one was to set a State of The Art for our data with the most commonly used networks and encoders. We've obtained very poor and not convincing results. However, given our data and the labels, the standard metrics did not apply and therefore did not return reliable and trustworthy results (we could not say for sure that our models were not performing well). We decided to explore the evaluation methods in object detection in order to have a metric that is specifically adapted to our problem. Theoretically, the new metric returns reliable values for the amount of TP, FP and FN in a prediction. However the segmentation models did not return good predictions and our new evaluation showed that they were not performing well at all.

The quality of the labeling and the ineffectiveness of segmentation models on our data has led us to the conclusion that it would be better to lead an "Object Detection" study on the data, for the quality of the labeling was too approximate.

10 Aknowledgements

During this internship I've had the opportunity to be supervised mainly by Javiera Castillo and from afar by Sebastien Lefevre and Thomas Corpetti. I am extremely grateful for their time, their pedagogy and the opportunity they have given me to explore and study a branch of artificial intelligence that I am not familiar with. I would like to thank the people at the LETG Lab in Rennes as well for their warm welcome.