Semantic Segmentation State of The Art on UAV Imagery for Plant Species Classification

Pierre Lague

Université de Rennes 2 / Université Bretagne Sud

Internship Oral Defense

Table of Contents

Introduction to the project

1. The internship's task is part of the SixP project. It aims to deploy innovative methods of characterization and identification of the vegetation via remote sensing and artificial intelligence.

Objective 1:

Combine remote sensing approaches and in situ species identification via artificial intelligence methods to allow mapping of species present in sites of interest.

Objective 2:

Establish a state of the art on semantic segmentation methods that can be used for our data (UAV imagery) and that achieve significant scores.

Introduction to Semantic Segmentation

Definition

The goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented.

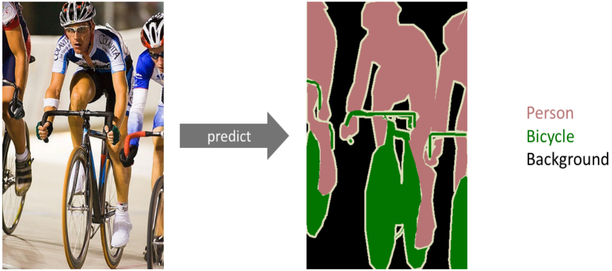


Figure 1: An example of semantic segmentation, where the goal is to predict class labels for each pixel in the image

How to achieve semantic segmentation ?

One popular approach for image segmentation models is to follow an encoder/decoder structure :

- **Down-sample** the spatial resolution of the input, developing lower-resolution feature mappings.
- **Up-sample** the feature representations into a full-resolution segmentation map.

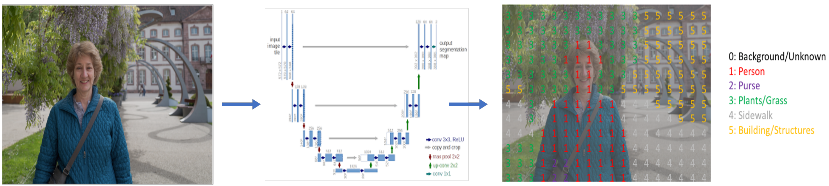


Figure 2: Expected output for semantic segmentation

Table of Contents

Figure 3: The private company's logo "l'Avion Jaune"

Study Sites:

1. Talus -> **513 images (103 test / 410 train)**
2. Chichoue_Bas -> **95 images (19 test / 76 train)**
3. Chichoue_Haut -> **280 images (56 test / 224 train)**
4. Chichoue_Milieu_Bas -> **186 images (38 test / 148 train)**
5. Chichoue_Milieu_Haut -> **295 images (59 test / 236 train)**

Remark

The particularity of the labels are such that a plant is represented by a full circle :

(a) Sample from the Talus site

(b) Ground Truth

Figure 4: An image and it's associated label

Here are the things we can expect during the training phases :

Remark

- The model might have issues detecting the edges of the plant (leaves, highly uncanny shapes ...).
- The amount of data is not as important as the standards for semantic segmentation studies, this will affect the performance of the model.

Table of Contents

Overall Process

The complete process of the study is as follows :

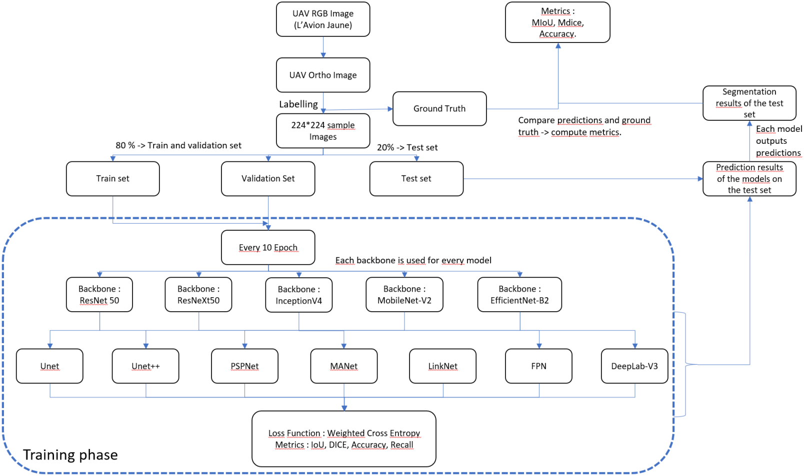


Figure 5: Flow Chart of the project

Tools

"segmentation_models_pytorch" (SMP) allows the access to a variety of architectures and backbones.

Architectures :

- Unet
- Unet++
- PSPNet
- MANet
- LinkNet
- FPN
- DeepLab-V3

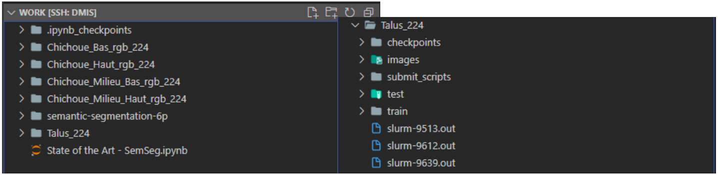
Backbones :

- ResNet-50
- ResNeXt-50
- Inception-V4
- MobileNet-V2
- EfficientNet-B2

Other libraries such as pytorch, numpy and sklearn were used during the implementation of the training / testing phases.

Methods

Our project is organized as follows :



(a) Overall Organization (b) Organization for a study site

Figure 6: Different Levels of Organization

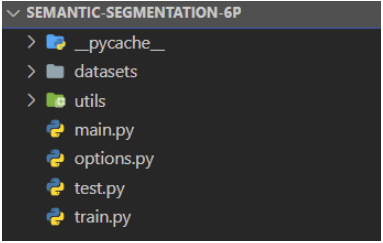


Figure 7: Content of the scripts folder

Table of Contents

Metrics

Remark

- Asses the models quality and capacity to fulfill it's main task.
- Need to use multiple in order to have a reliable evaluation.

The metrics we decided to use are extracted from a confusion matrix :

Used Metrics:

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

Handling Class Imbalance

Before computing the metrics, we need to handle **Class Imbalance**.

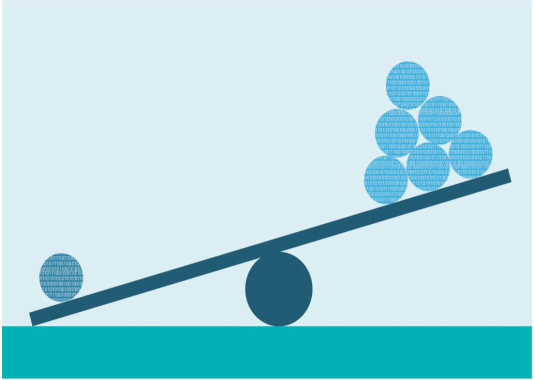


Figure 8: Class imbalance has an important effect on the quality of the predictions

Methods overcoming class imbalance can be divided into two main categories.

Sampling-Based Methods :

Figure 9: Sampling based methods operate directly on the datasets with the aim to balance class distribution

Algorithm-Based methods :

Remark

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'll be using an algorithm-based method called **Weighted Cross Entropy**. To compute the weights for each classes we have two options :

- Outside the inference phase : **static weights**
- Inside the inference phase : **dynamic weights** based on the batch.

Remark

Since we don't have a lot of data, we decided to compute the weights inside the inference phase without having a high computational costs.

Table of Contents

Here are the results we obtained on the Talus Dataset only the best results are displayed :

Network	Accuracy	Mean IoU	Mean DICE
Manet	**21.94%**	**0.0972**	**0.1444**

Table 1: ResNet50 - Talus - 9 classes

Network	Accuracy	Mean IoU	Mean DICE
Unet	**19.99%**	**0.0864**	**0.1345**

Table 2: ResNeXt50 - Talus - 9 classes

PSPNet	**23.49%**	**0.0760**	**0.1233**
LinkNet	**23.43%**	**0.0813**	**0.1266**

Table 3: EfficientNetB2 - Talus - 9 classes

Network	Accuracy	Mean IoU	Mean DICE
Unet++	**25.15%**	**0.0965**	**0.1429**
PSPNet	**25.49%**	**0.0731**	**0.1136**

Table 4: InceptionV4 - Talus - 9 classes

Network	Accuracy	Mean IoU	Mean DICE
PSPNet	**29.86%**	**0.0792**	**0.1181**

Table 5: MobileNetV2 - Talus - 9 classes

Remark

Overall the results are very low and not convincing. There are two reasons for this :

- lack of data
- approximate labelling -> metrics are not entirely reliable

Table of Contents

Object Detection Evaluation

New Method

Change the scale of evaluation : from a **pixel-based** evaluation to an **object-based** evaluation.

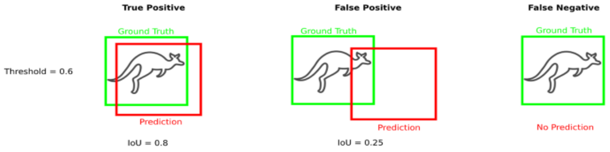


Figure 10: New ways of computing the metrics to evaluate our models

Remark

The goal is to compute the **Average Precision** for each classes and then the **Mean Average Precision**.

Algorithm to compute TP, FP and FN

For i in all plant classes :

P = predicted binary mask for class « i »

M = ground truth binary mask for class « i »

CC = sklearn.measure(m) get all different components of M

For j in all the components of class « i »:

C = binary mask for component j for class « i »

intersection = np.multiply(P, C)

If sum(sum(intersection)) / sum(sum(c)) > threshold :

TP [i] = TP [i] +1

Else :

FN [i] = FN [i] +1

B = get any pixels predicted as class « i » but not on the ground truth

CCB = sklearn.measure(b) get all different components of B

For k in all the components of class « i » not in GT:

C = binary mask for component k for class « i » not in GT

If sum(sum(c)) > area :

FP [i] = FP [i] +1

CC = all components of a class labelled differently

Intersection allows us to assess how much of the prediction is « on » the ground truth

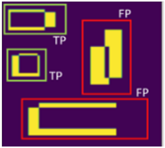


Figure 11: Illustrated algorithm to compute TP, FP and FN with an example

Results with the new method

Steps to evaluate the models

- Include the algorithm into the testing phase
- For all classes and for multiple thresholds, compute Average Precision
- Compute Mean Average Precision

Class	Average Precision
1	0.09787

Table 6: Average Precisions class 1

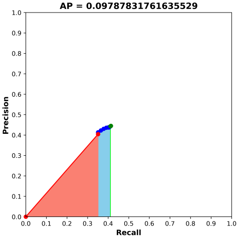


Figure 12: Precision-Recall Curve for Class 1, Average Precision is 0.097

Remark

Mean Average Precision The Average Precision for all the other classes was 0. The Mean Average Precision is 0.009787

Conclusion

The new way of computing the metrics is adapted to the setup of our experiment and our labels. This way we can reliably evaluate our models :

- The Mean Average Precision shows that our models are of very poor quality.
- The discriminant capacities of our models are also very low, even after trying out multiple configurations.

Our models did not achieve significant scores for them to be used on the field. Semantic Segmentation may not be the most effective approach, given the setup of the experiment.

Future Work

Object Detection has a lot of potential. Bounding boxes can easily be constructed out of the labels we already have. The problem has to be less granular in order to achieve better scores.

Thank you for your time.