CSC_5IA23_TA

Computer Vision Project

MOREL Florian

GUINOT Tao

BONNET Thomas

DENIS Matteo

Link to our Github repo

Introduction

In this project, we aim to classify satellite images with their wildfire exposure. The given dataset provides about 40.000 labeled satellite images, being 1 if the zone previously experienced a wildfire, and 0 if not.

To achieve this goal, we propose different methodologies: The first one is a supervised method, using the "new" train set, being a mix of train and validation set from the original dataset. Then, an unsupervised method, with a mixture of visual-encoding and clustering methods. Finally, two semi-supervised methods, by training a first model with our training data, then generating additional examples with pseudo-labels, and improving our classifier.

Contents

1 Supervised learning method

Firstly, it is important to note that we understood using this method was not the goal of this project. However, as we started with it, and achieve a quite decent accuracy, we thought it would be interesting to talk about it.

The most basic illustration to this approach is our `SupervisedClassifier` method. The user can choose between a handmaded `Net`, or a `ResNet50` structure, classifying into two classes: 0 and 1. The `Net` is a basic CNN model composed of 5 `Conv2d` layers, each separated by a `MaxPool2d`, and followed by an `AdaptativeAvgPool2d`. After flattening, the encoded output is passed through 2 `Linear` layers separated by a `Dropout` layer.

We also implemented a `SupervisedViT` method, that works similarly to the preceding, with a ViT pretrained model.

2 Unsupervised learning method

We also implemented a fully unsupervised method, mixing encoding models and clustering models.

For this, we simply encode the images, using either `ViTEncoder` or `ResNetEncoder`, and then passing them into a clustering algorithms, being either `KMeans` or `DBSCAN` (from sklearn) with two classes, assigning each encoded image into 0 or 1.

Finally, it is now easy to compute the accuracy of our method, simply by comparing labels assigned by the clustering algorithm to ground-truth labels.

3 Semi-supervised learning method

The specificity of the exercise is that we could not leverage the labels of the train dataset. Therefore, it has been impossible to apply classical supervised-learning methods on this data, but we could leverage this unlabeled data with semi-supervised methods. We have worked on two different approaches, we called `SemiSupervisedClustering` and `SemiSupervisedSelfTraining`.

3.1 SemiSupervisedClustering

The first method relies on the pseudo annotation of unlabeled data using a clustering method. We assume that two similar images, whatever the type of similarity, have the same class *wildfire* or *nowildfire*. The clustering method, K-means in our study, is performed using encoded representation of the images.

3.1.1 Encoding

We tried two different approaches to encode the images : segformer and vit.

segformer This approach relies on the fine-tuning of the nvidia/mit-b0 model, a transformers based model, pre-trained for semantic segmentation. We trained it on the deepglobe-land-cover-classification-dataset in such a way the model classifies each pixel in one of the following class : Agriculture land, Barren land, Forest land, Range land, Urban land and Water. The number of pixels for each class are then used to create a 6 components vector, being the encoded image. The main advantage of this method is the small number of components an encoded image has. It is a good point for the future clustering. However, The spatial aspect of the image is lost.

The Figure 1 and 2 show the evolution of the loss and the Intersection over Union (IoU) metric computed over the validation set during the training. We can notice a clear improvement of the model to segment the different lands on the deepglobe-land dataset, especially for barren lands and forest lands. However, the qualitative examples given on Figures 3 and 4 show important confusions on some part of the images. It explicitly refers to the mean IoU on Figure 2 reaching only a 50% score, which is rather low to be able to ensure that our segmentation model is really efficient.

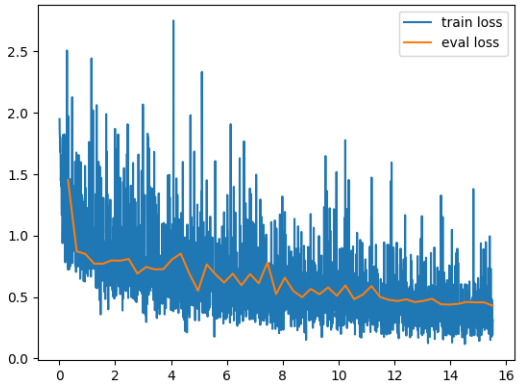

Figure 1: Loss Function of the fine-tuning of segformer

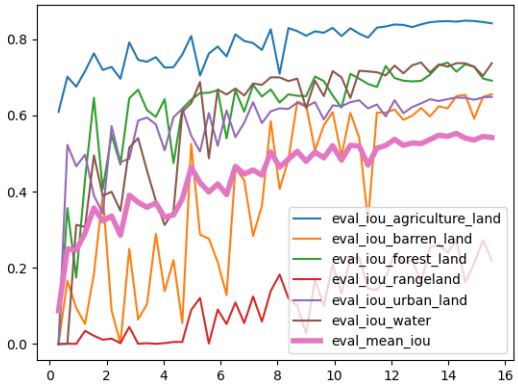

Figure 2: IoU metric on the validation set during the fine-tuning of segformer

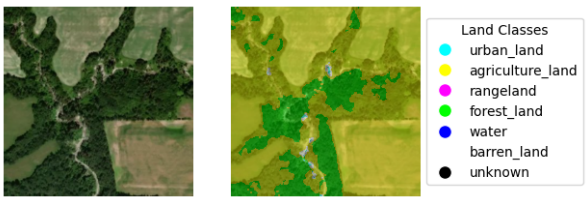

Figure 3: Qualitative result of the trained segformer on a **wildfire** sample

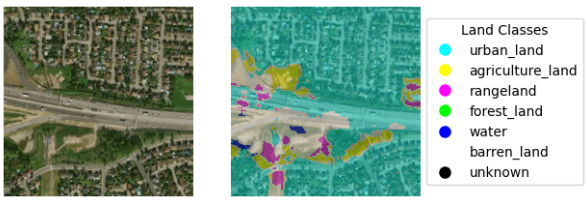

Figure 4: Qualitative result of the trained segformer on a **nowildfire** sample

vit The vit approach is simply an embedding of the images with a Vision Transformer, google/vit-base-patch16-224-in21k, from which we keep only the output of the last hidden layer. An image is then encoded into a vector of 768 dimensions.

3.1.2 Clustering

Clustering is performed on 80% of the validation set, where images are labeled, and 100% of the train set, where images are unlabeled. As un hyperparameter, we fix the number of clusters in order to maximize the heterogeneity of each cluster. Heterogeneity is the percentage of identically labeled samples from the validation set into a single cluster. Theoretically, if our previous assumption was correct (similar images have similar labels) we should have a 100% label homogeneity. As we can see on Figures 5 and 6, it is not what happened. Nevertheless, for each cluster, we got the most represented label among validation samples and assigned this label to the train samples inside. It corresponds to the *Label* column. We have now a bigger and more diversified dataset to train a supervised model, resnet50 in our study, to detect if an image represents a *wildfire* or a *nowildfire* scene. A general architecture of this method is shown on Figure 7.

Cluster	Label	Perc. of labeled data	Label homogeneity	Nb of samples
00	0	13.75 %	95.69 %	8437
01	1	14.60 %	92.52 %	11722
02	1	14.74 %	86.14 %	7932
03	0	13.89 %	85.30 %	7198

Figure 5: Clustering information with the ViT approach

Cluster	label	Perc. of labeled data	Label homogeneity	Nb of samples
00	0	13.47 %	53.50 %	2331
01	1	14.57 %	92.62 %	11988
02	0	14.10 %	73.11 %	17455
03	1	14.71 %	69.83 %	3515

Figure 6: Clustering information with the segformer approach

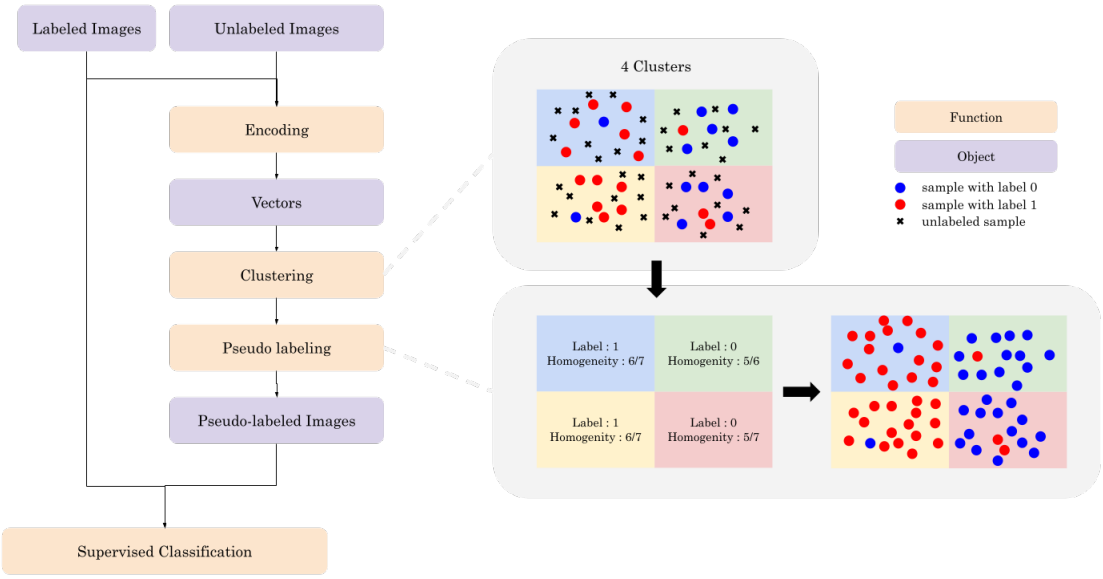

Figure 7: Functional view of the `SemiSupervisedClustering` method

3.1.3 Limitations

The performances of this method are limited by the accuracy of the clustering. If the unlabeled data are not correctly pseudo-labeled, the model is trained on biased data and becomes biased. Given the label homogeneity we have in Figure 6 and 5, we can expect an accuracy limited by the badly pseudo labeled data, because there are. This method could work with smaller amount of labeled data, but is inefficient for the dataset we are working on, where a supervised method on only labeled data reaches a 99% accuracy.

3.2 SemiSupervisedSelfTraining

The second method we tried to improve the performance of the based supervised classifier is inspired by an article from Amini and al. named Self-Training: A Survey. The idea is to iteratively train a classifier in N training steps, by pseudo-labeling samples with the highest confident prediction made by the trained model at each step $n < N$. We assume that if our model is highly confident for a prediction, it probably means that the prediction is the gound truth. The general architecture of the method is shown on Figure 8.

The main challenge of this method is to fix hyperparameters : threadshold and proportion. The threadshold is the minimum confidence to consider in order to pseudo-label an example. The proportion refers to the maximum of data points to pseudo-label at each step n. The goal is to find the best hyperparameters to really improve the performances of our model at each step n without biasing it by introducing too many pseudo-labels, it would comfort it in its bad predictions. We manually set the threshold to 99% and the proportion to 1000 newly pseudo-labeled samples.

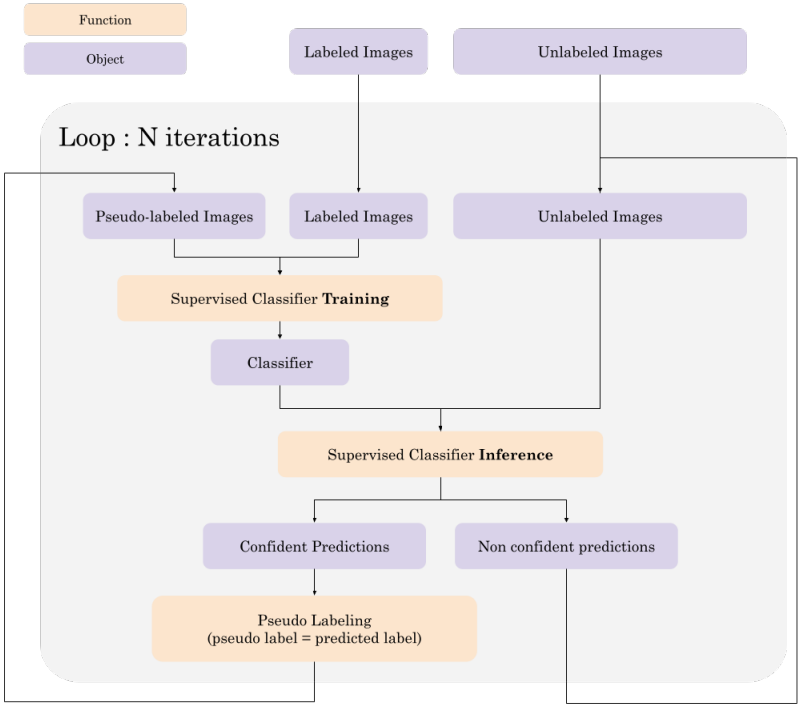

Figure 8: Functional view of the `SemiSupervisedSelfTraining` method

4 Results

To compare the different method, we used the Accuracy metrics. The class are balanced, this metrics seems adapted. Here is how our different approaches and methods compare on the full test dataset:

Approach	**Method**	**Accuracy**
Supervised approach	`s_classifier_resnet`	0.9911
	`s_vit`	0.9895
Unsupervised approach	`us_clustering_vit`	0.9074
	`us_clustering_resnet_net`	0.8838
Semi-supervised approach	`ss_selftraining_resnet`	0.9898
	`ss_clustering_vit_resnet`	0.9751
	`ss_clustering_segformer_net`	0.8943

Table 1: Comparison of different approaches based on accuracy.

As we can see on the Table 4, the best accuracy is reached with the fully supervised method using a resnet50 classifier. It shows that despite some attempts to improve the performances of the prediction leveraging unlabeled data, the semi-supervised learning methods are ineficient. For the self-training method, we wanted to implement our own model, beginning by the simplest way to do it : a threadshold and a maximum proportion. However, some papers like FlexMatch or FreeMatch, introducing adaptative threadsholdind or curriculum pseudo labeling, shown clear improvement on classification on common datasets like CIFAR-10 or CIFAR-100. It would be an idea to pursue in the future.

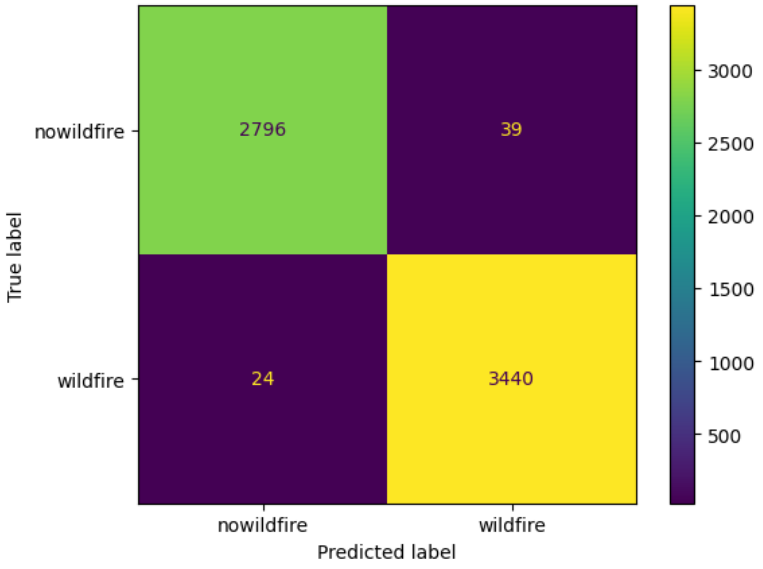

Figure 9: Confusion matrix of the `s_classifier_resnet` method