Ecole Polytechnique
Fédérale de Lausanne

ENV-540 :Image Processing For Earth Observation

Project Report : Semantic segmentation of Alpine land cover

Swann Destouches
Charlotte Girard
Aurélien Texier

12 janvier 2023

Link to the github repository : `https://github.com/a-texier/IPEO-project-2022.git`

Table des matières

1 Introduction

Land cover classification is a process in which pixels in an image are analyzed and categorized into different classes based on the type of land cover present. This is typically done using remote sensing techniques, such as satellite or aerial imagery, which can provide detailed information about the Earth's surface.
Semantic segmentation is a method allowing to construct maps for such land cover classification, by dividing area in the image in visually meaningful or interesting areas. Nowadays a common method for land cover classification is the use of deep learning. [1] For land cover recognition these method are usually perform on orthoimages, which are aerial photographs that have been geometrically corrected to eliminate distortion caused by the camera's orientation and position. These orthophotos are often limited to three channels of data, corresponding to the red, green, and blue (RGB).
There are various machine learning approaches that can be used for semantic segmentation, including supervised and unsupervised learning methods. This project focused on supervised learning, where a model is trained on a labeled dataset, and the correct class for each pixel regarding the user interpretation before hand. This allows the model to learn the relationships between the pixel values as well as their surrounding and the different land cover classes, and to make predictions about the class of new pixels based on their RGB values. Convolutiononnal Neural Network (CNN) and their extended Fully Convolutional Neural networks (FCN's) have already shown excellent performances for such task, hence this project adopt this solution testing multiple models architectures.[2] [3]

Challenges

Several challenges can arise when performing semantic segmentation of mountaineous land cover using machine learning on RGB bands of orthoimages. Mostly challenge reported in literature is the complexity of terrain : The mountainous terrain of alpine regions can be complex and varied, with steep slopes, cliffs, and other features that can make it difficult to accurately classify land cover. This complexity can make it difficult for machine learning algorithms to accurately identify the boundaries between different land cover classes by adding variability in the data. [4]
Another constrain comes from the dataset and data availability. Firstly land cover classification is mainly used for urbanized areas of great interest, while mountains land cover is less studied [5]. Moreover mountains, due to their geomorphology put together a lot of heterogeneity with sometime rare but key occurrences of land cover, such as glacier for example. This kind of limiting factor can be bypass using methods like data augmentation, investigated in this work. [6]

Objective of this work

The goal for this project is to implement deep learning models to perform semantic segmentation of Alpine terrain. The given dataset contains 12'262 image and labels tiles collected by SwissTopo in the Dents du Midi area (Valais). The image tiles have a spatial resolution of 25 cm and three RGB bands. The labels have a spatial resolution of 50cm. The dataset is divided into training, validation and test sets of 60%, 10%, and 30% of the tiles, respectively 7'357, 1'226 and 3'679 images. Splitting is given and the land coverage categories are explicated in table 1.

TABLE 1 – Categories of land coverage to determine

Category	Number
Grasslands	0
Forest	1
Sparse Forest	2
Water	3
Snow	4
Sparse rocks	5
Scree	6
Rocks	7

2 Methods

2.1 Preprocessing

After loading the data, the first part initiate the dataset. It is split into training, testing and validation datasets according to the csv file given.

2.1.1 Normalize the dataset

Normalization step enable to have uniform distribution of pixel intensity within a band in order to have better accuracy thanks to balance in the data. It is done by computing the mean and standard deviation of the training and validation dataset and normalize the pixel values around a zero-mean and unit standard deviation distribution.

2.1.2 Weighted class

In order to prevent bias in the learning method because of prevalent classes appearing in the dataset, a common solution is to associate weight in the loss function for each class. To do so a class weight is associated within the torch tensor. This class weight is computed using the frequency of pixel within the total pixel count for all the images.

2.1.3 Data augmentation

Data augmentation is used to over-sample the data set for the under represented classes. After a first run of the model, the resulting confusion matrix is interpreted to extract the class that gives bad prediction. Coupled with the pixel count for each classes done with the ground truth, it is possible to define three classes that lack representation and apply data augmentation over these ones.
Applying data augmentation on every images containing under-represented class add too many data for the model to be time efficient. Hence the data augmentation is done by duplicating the image containing more more than 30% and applying a rotation. This new image and its ground truth are added to the dataset. In the initiation of the dataset, a condition is given to extract images with the under-represented class superior to 30% in order to proceed to the data-augmentation.

2.1.4 Fine tuning

Once the best model architecture has been found, some tuning on weighted class and data augmentation methods are implemented to fix imbalanced issues and improve the results. Especially improve the producer's accuracy and therefore the average accuracy instead of the overall accuracy. To really understand how these methods have an impact on the results, we implement first the weighted method alone, then the data augmentation alone, then together and eventually we tune the parameter on both methods manually regarding the several matrix confusion than we obtain.(See part 3)

2.2 Models

Semantic segmentation models have a typical two step architecture : an encoder and a decoder part. At contrary to image classification models like AlexNet, this allows to obtain as an output a classified image of the same size as the input where the different categories are mapped, see figure 1

D. Tuia, D. Marcos, K. Schindler, and B. Le Saux. Deep learning-based semantic segmentation in remote sensing. In *Deep learning for Earth Sciences - A comprehensive approach to remote sensing, climate science and geosciences*. Wiley, 2021.

FIGURE 1 – Image classification and Semantic Segmentation Architecture and outputs.

2.2.1 U-Net

Basic two step architecture to do semantic segmentation is U-net model. This model consists of a series of convolutional and upsampling layers that process the input image and produce a segmentation map as output. The U-Net is a popular choice for image segmentation tasks because it is able to make use of both the high-level context provided by the downsampling layers in the encoder, and the detailed information provided by the upsampling layers in the decoder. This type of architecture consists of an encoder, a bridge, and a decoder forming a U shape when represented as in figure 2. The encoder is a seris of convolutional and downsampling layers that process the input image and extract features at different scales. The bridge consists of a single residual block that processes the features extracted by the encoder. The decoder does a series of upsampling and concatenate residual blocks from the bridge. It processes the features from the bridge and the encoder (through skip connections), and produce a segmentation map as output using a final classifier.

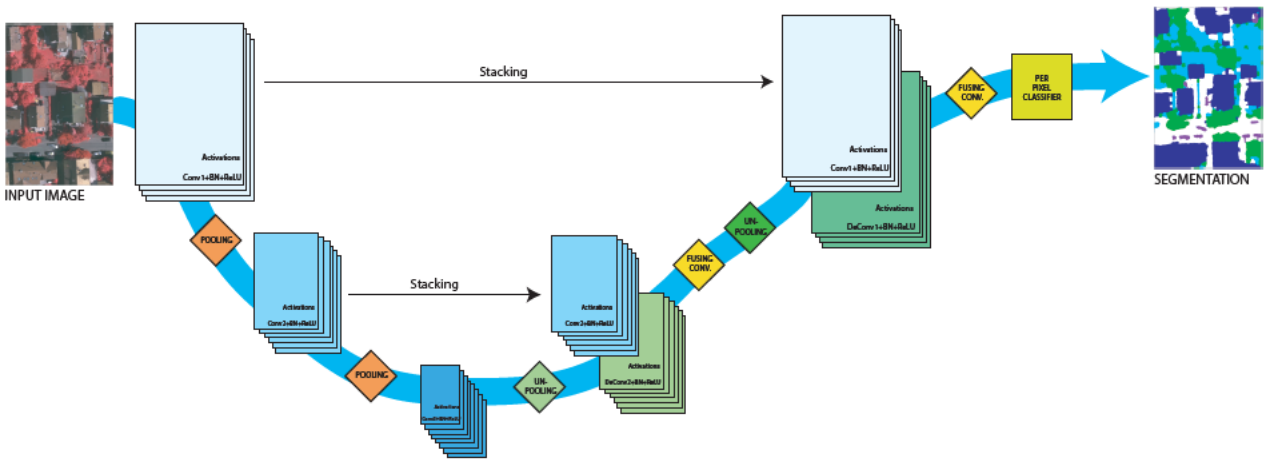


FIGURE 2 – Basic U-Net Architecture

The U-Net model used is defined using the "nn.Module" class from PyTorch's torch. It consists of several submodules proceeding to the down and up pathways of the Unet model. In the encoder part the module downsamples the image tensor by applying max pooling 2x2 and a DoubleConv operation. "DoubleConv" submodule is defined by two consecutive steps of convolutional operations with kernel size 3 and padding 1 to augment the number of channels and reduce their size followed by batch normalization and ReLU activation. It takes as input an image tensor and the number of input and output channels, and it applies two convolutional steps with kernel size 3 and padding 1. The up-pathway, decoder part is composed of the module upsample that concatenate the down-pathway result at the same level with the upsampled layer, a shortcut connection. Then it applies a "DoubleConv" operation. The upsampling is performed using either bilinear interpolation or a convolutional transpose operation. The final output uses a module applying a single convolutional operation to the upsampled layer to produce the final segmentation map. It takes as input the number of output of the last decoder block and outut the number of classes, and it applies a convolutional operation with kernel size 1 to produce the output using a Multi-Layer-Perceptron (MPL) classifier. We also prospect solution with the very same U-Net model by using a Random Forest classifier as final output module instead of the MLP. The classifier is composed of a certain number decision trees that each evaluate the 16 features given for each pixel and predict with these the class of the pixel. In order to be able to run the training in a reasonable time, the tree number is assigned to 10. Results with this method are worst than basic U-Net while the computational time is a lot higher. Hence this solution was abandoned.

2.2.2 ResUnet

Residual U-Net (ResUnet) is a variant of U-Net architecture. In addition to the standard U-Net, the ResUnet model includes residual connections between the encoder and decoder layers as shown

in figure 5. The residual connections in a ResUnet model are implemented using skip connections, which bypass the upsampling step in the network and directly connect the input to the output of each decoder layer. By adding these skip connections and then perform a ReLu activation function, it allows the back-propagated gradient to be different from zero in any case, so that the network can learn to correct the output of the skipped layers, rather than just learning to correct the final output. This can improve the ability of the network to learn from the input data, and can lead to better performance on the image segmentation task.

The model used is the one proposed in the paper of Zhang et al. [7] represented in figure 5. The encoding part is composed of a first encoder that perform the following steps : a 2D convolution reducing the number of channels to 64 with a 3x3 kernel and padding of 1, then a batch normalization and Relu step, followed by two 2D convolution.

The second and third encoding parts are a residual block made of 2 consecutive batch normalization, ReLu activation function and 2D convolution 3x3 with padding of 1 as a block, where a identity convolution of the input is summed as a short connection. The bridge is connected with the same residual block. The forward pathway is composed of three decoder blocks that consist in first upsamples inputs using a bilinear upsampling operation, which increases the height and width of the tensor by a factor of 2. It then concatenates inputs and skip along the channel dimension. Finally, it applies a residual block to the concatenated tensor. The output of the model is the result of applying a single convolutional layer to the output of the third decoder block.

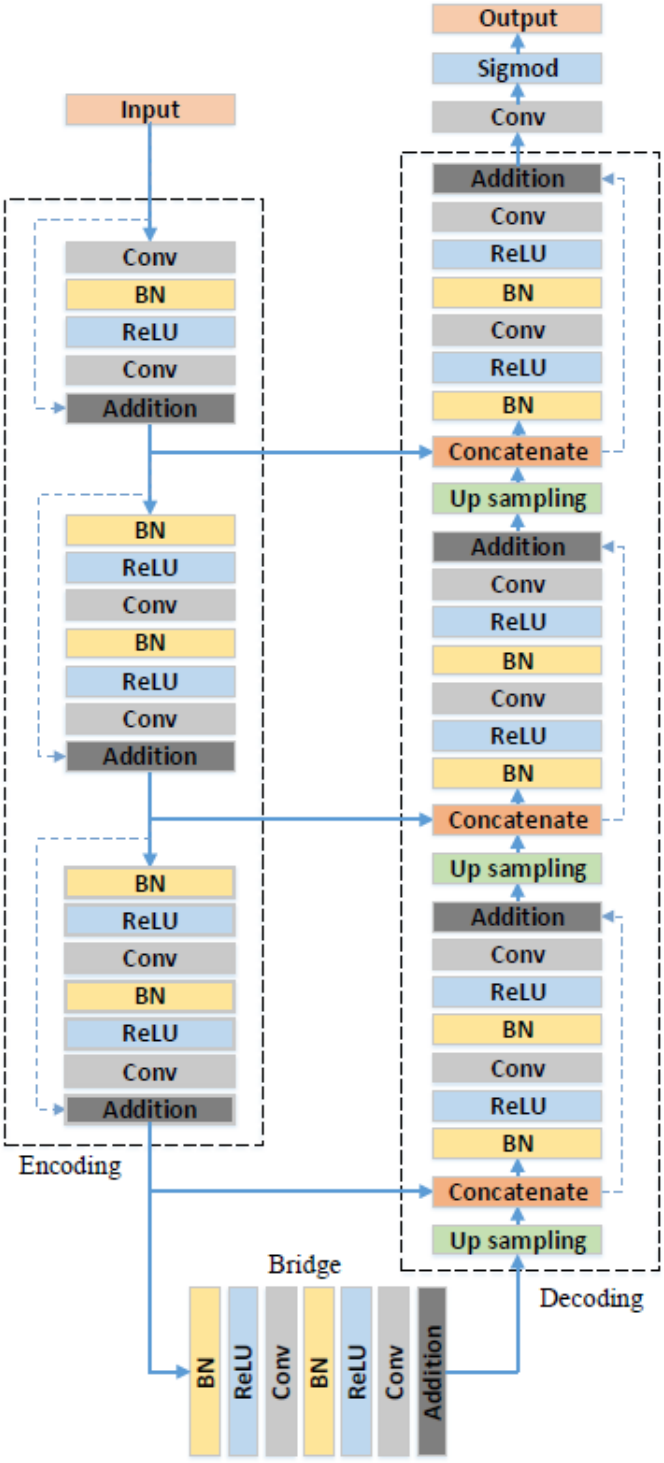


FIGURE 3 – Architecture of the ResUnet used proposed by [7]

2.3 Model optimizer

These models are trained by minimizing a loss function that measures the difference between the predicted segmentation map and the ground truth labels. During training, the model's parameters are updated using an optimization algorithm in order to reduce the loss and improve the performance of the model. Adam (Adaptive Moment Estimation) is used to update the weights of a neural network during the training process. It is an extension of the stochastic gradient descent (SGD) algorithm and combines the benefits of both SGD and the Root Mean Square Propagation (RMSProp) algorithm. Adam works by maintaining an exponentially decaying average of the past gradients and the past squared gradients, and using these averages to adaptively update the model's weights. It computes the learning rates for each parameter individually, which allows it to handle sparse gradients and large gradients with different scales.
Once trained, the model can be used to predict the labels for a new input image by forwarding the image through the model and applying a suitable threshold to the output segmentation map. Then it is important to proceed to validation test in order to tune the hyperparameters in order to avoid overfitting. It is also the step were fine tuning takes place in order to improve the different accuracies of the model.

2.4 Loss functions

The loss function is used to evaluate the performance of a model by comparing the model's predicted output with the ground truth. Two loss function are compared with the best predicting model :

- **Cross entropy loss :** Cross-entropy loss or negative log likelihood loss, is a commonly used loss function for classification tasks. The loss is calculated as the negative logarithm of the predicted probability of the correct class. The predicted probability is calculated using the softmax function, which converts the predicted class scores into a probability distribution over the classes.
- **Dice loss :** The dice coefficient is also often used as a loss function in image segmentation tasks, with the goal being to minimize the value of the dice loss and maximize the overlap between the predicted and ground truth labels.
  On certain cases, it is a very interesting alternative to the cross entropy as it will not give a better overall accuracy but focus more on the average accuracy among the different classes.

2.5 Metrics

This section explains the different metrics that are used to quantify the quality of the results.

2.5.1 Confusion Matrix

Confusion matrix is a table that is used to evaluate the performance of a classification algorithm. It compares the predicted class labels with the true class labels and counts the number of correct and incorrect predictions. Multiple indicators can be extracted such as the overall accuracy. It is the proportion of correct predictions made by the model out of all the predictions made. It is calculated as the sum of the diagonal elements of the confusion matrix divided by the total number of predictions. The average accuracy is the average of the diagonal elements of the confusion matrix, normalized by the number of classes. It gives a measure of the model's performance on each class. If ever the model

fails to predict some classes, there will be lower average accuracy than the overall one.
The user accuracy is the proportion of the actual observations that were correctly predicted by the model. It is calculated per class as the diagonal elements of the confusion matrix for the given class divided by the sum of the elements of the given the column of the confusion matrix.
The producer accuracy is the proportion of the predicted observations that are correct. It is calculated per class as the diagonal elements of the confusion matrix for the given class divided by the sum of the elements in the given row of the confusion matrix.

2.5.2 Dice coefficient

The dice coefficient (or F1-score) is a metric used to compare the similarity of two sets. It is defined as the ratio of the size of the intersection of the two sets to the size of the union of the two sets.

$$\text{Dice(A,B)} = \frac{2\,|AB|}{|A| + |B|}$$

where $|A|$ and $|B|$ are the number of element in the respective sets and $|AB|$ is the number of element overlapping the two sets.
In the context of image segmentation, the dice coefficient is often used to evaluate the performance of a segmentation model by comparing the predicted segmentation mask with the ground truth mask. The dice coefficient ranges from 0 to 1, with a value of 1 indicating a perfect match between the two sets and a value of 0 indicating that the sets have no overlap. [8][9]

2.5.3 Jaccard score

The Jaccard index (or IoU score) is doing the same task as the Dice coefficient but is more punitive regarding single instances of bad classification. It is defined as the overlap over the union :

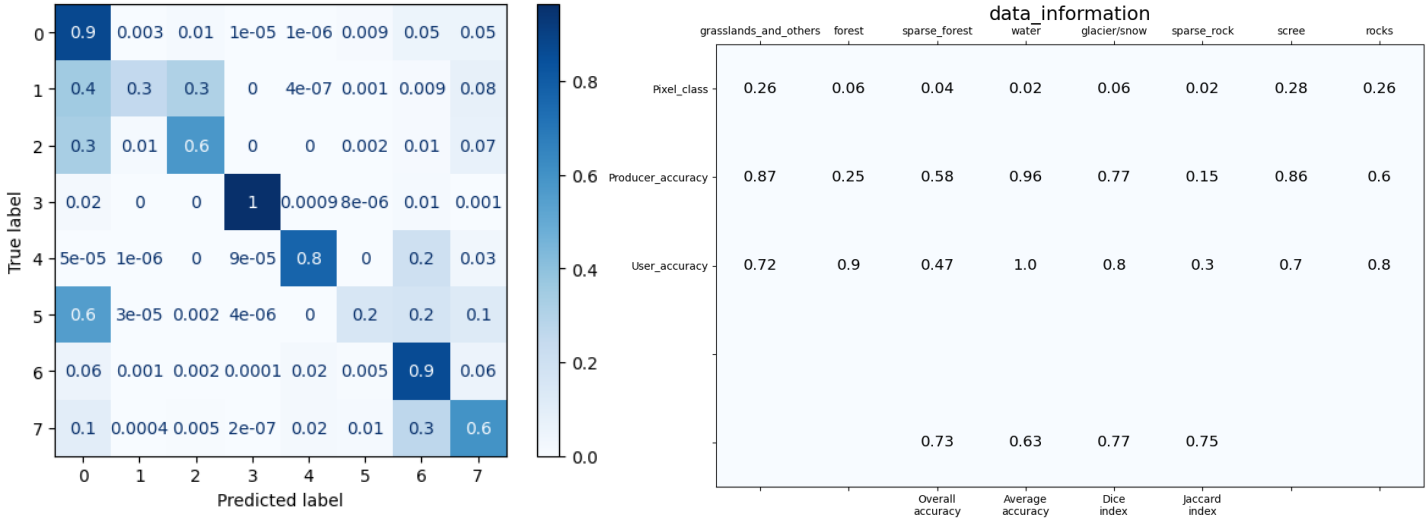
$$\textbf{Jacc(A,B)} = \frac{|AB|}{|A \cup B|}$$

where $|A \cup B|$ is the number of element in the union of the two sets.
As the Dice coefficient, its values range from 0 to 1, 0 being the worst and 1 being a perfect score. The Jaccard index and the Dice coefficient are positively corrolated, meaning that if a model is said better than an other by one index, the other one will say the same. [8][9]

3 Results

3.1 Basic U-Net

The confusion matrix figure 4 a) displays the results of the U-Net model normalized per class. It means that the diagonal of the matrix directly highlights the producer's accuracy of each class. We can deduce that the classes 1,2 and 5 are almost not predicted at all by the model, which is due to the imbalanced issue. We later used data-augmentation and weighted classes to deal with it. Moreover the forest and the sparse forest are often predicted as grassland, which is explained by the predominance of this class in the dataset. Figure 4 b) shows the overall accuracy of 73% and an average accuracy of 63%. This means there is a good classification for the prevalent classes but other under represented class fail the classification.



(a) Confusion matrix (b) Data information

FIGURE 4 – Results of the U-Net without dealing with imbalanced data

3.2 Fixing imbalanced data

Imbalanced data refers to a dataset where the distribution of the different classes is not equal. This can be a problem when training a machine learning model, as the model may end up biased towards the more prevalent classes. There is multiple way to fix this problem of data representativity. First it is possible to associate weight to the loss function for each function according to the data distribution. This can help the model to focus more on the underrepresented class during training. To determine the weights for each class we used inverse of the class frequency found with the number of pixel of each class in the ground truth images of the training set. Other ways are to interact with the data them-self, either by over or under sampling in order to increase/reduce the number of data for such under/over represented classes respectively. Data augmentation is a way to oversample.
For each way to fight imbalanced data, we will focus on the matrix confusion, and then the results that we can deduce such as the overall accuracy, the average accuracy and mostly the producer's accuracy. These methods to fix unbalanced dataset are build first on U-Net model and then implemented on the ResUnet. All results are given for 100 training epochs.

3.2.1 Weighed class

The idea is to weigh the loss function whether they belong or not to the majority. A higher weight is assign to the loss for the sample which belong to the under-representative class. The weights are based on the frequency on each class in the whole dataset.

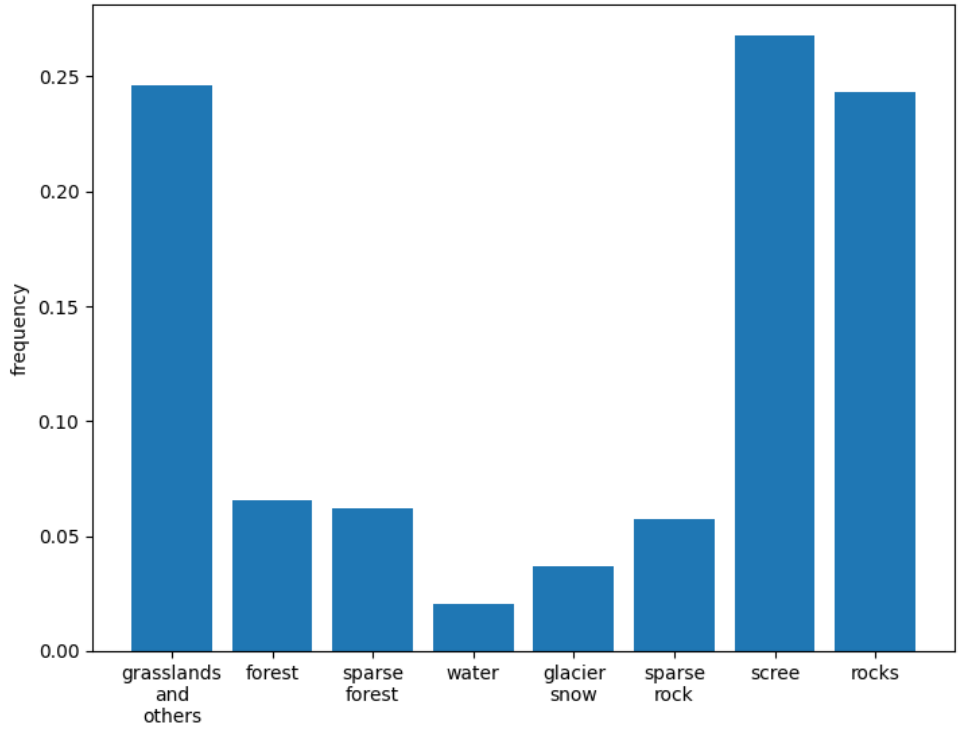
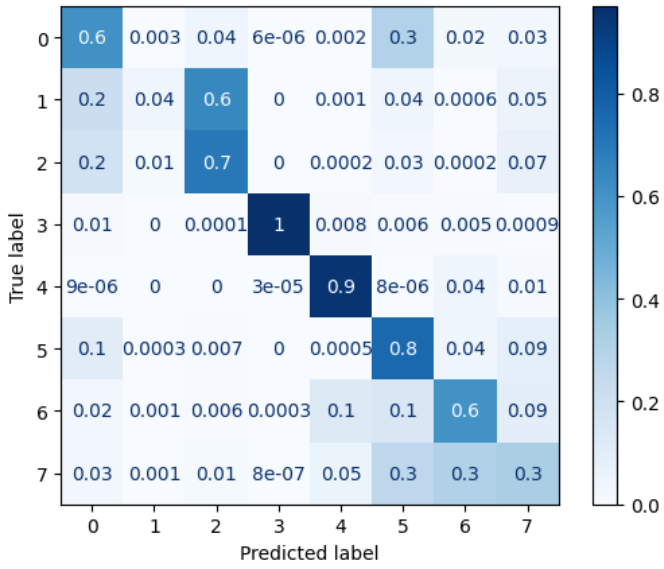


FIGURE 5 – Frequency of each class in the training and validation dataset



Having weighted classes change a lot the producer's accuracy compared to U-Net results (section 3.1). Indeed it has allowed the model to better predict the 5^{th} class and also to make some changes between the classes 0,1,2. However there are still some confusions between forest and sparse forest and the rocks are also confused with the sparse rock and the scree.

FIGURE 6 – Confusion matrix - Weighted class

3.2.2 Data augmentation

The idea here is to oversample some images to increase the total number of pixels of the under-representative classes in the dataset by duplicate and modify the images with more than 30% of the minority classes. (classes 1,2 and 5).

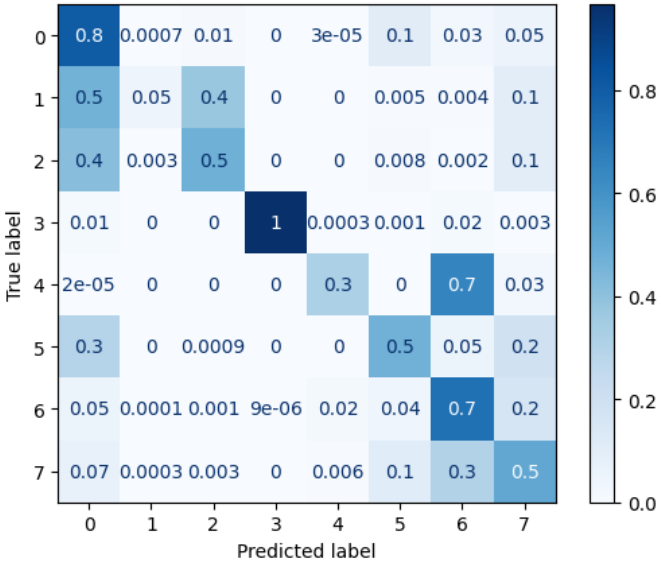


FIGURE 7 – Confusion matrix - Data augmentation

Oversampling alone does not help with the confusion issues between the classes 0,1 and 2, i.e. grassland, sparse forest and forest. Nonetheless it increases the discrimination power between sparse rocks and scree.

3.2.3 Weighted class & Data augmentation

Here we implement together the two methods to fix data imbalance, weighted classes with data augmentation.

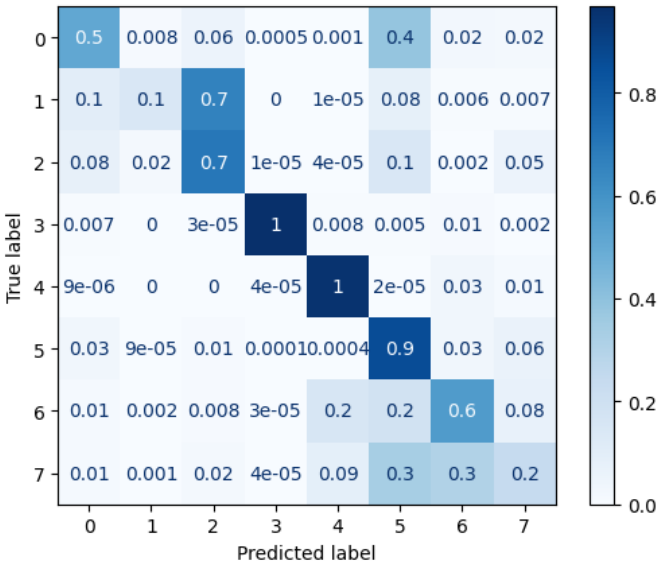
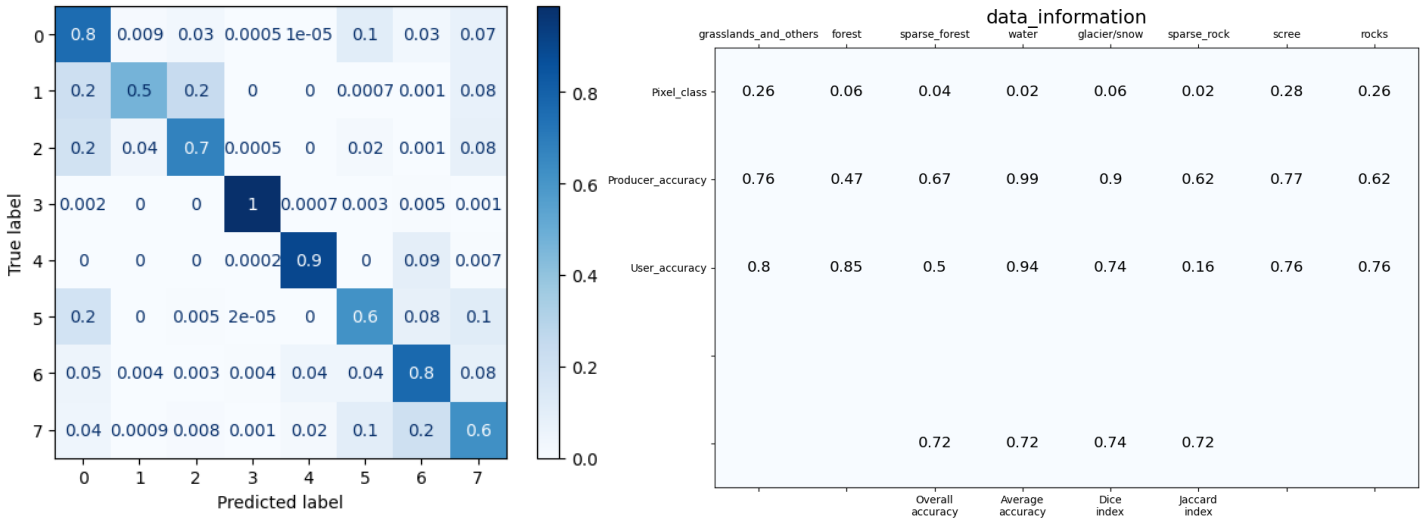


FIGURE 8 – Confusion matrix - Weighted classes + Data augmentation(0.3)

The combination of weighted classes and oversampling do not produce significant improvement compared to weigh classes alone.

3.2.4 Tuning both methods to get the best model

Although the previous implementation do not solve entirely the imbalanced issues, the literature mention that it is quite common to tune some parameters based on the previous confusion matrix to considerably increase the goodness of the results [10]. Indeed each machine learning problem is unique and has to be tune empirically.



(a) Confusion matrix (b) Data information

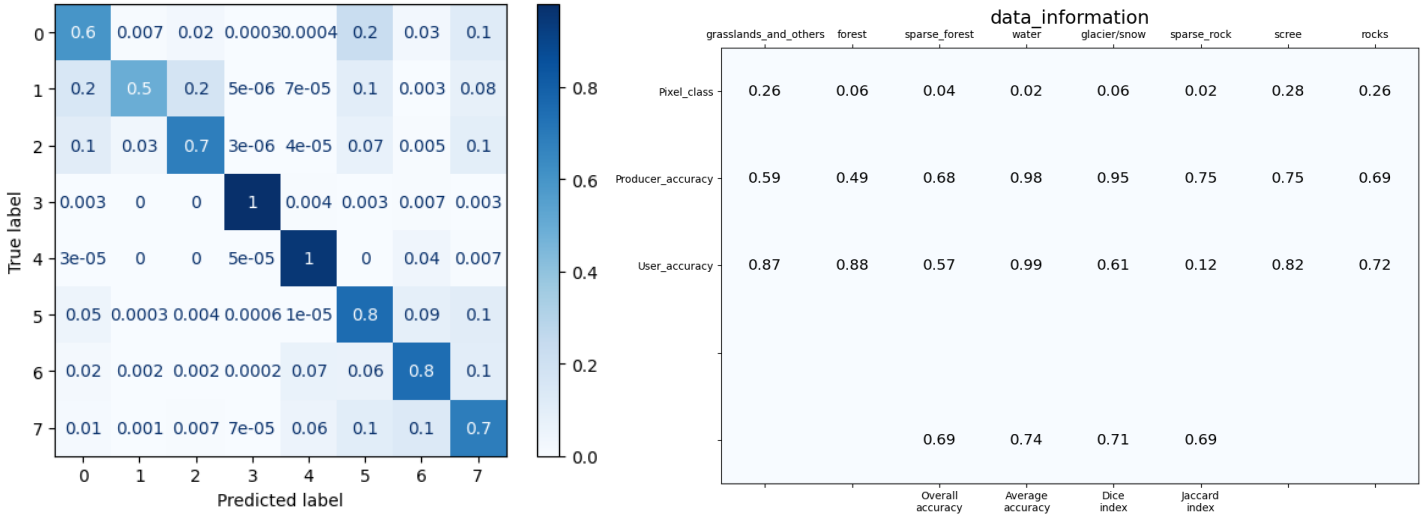
FIGURE 9 – Results of the U-Net with Weighted classes + Data augmentation boosted

From the previous confusion matrix, we boosted some classes with more specific oversampling and weighted classes.
Oversampling : According to the fraction of classes inside the images, we decide to oversampling the dataset as follow : (Class1) $>$ 10%, (Class2) $>$ 50%, (Class5) $>$ 30%, (Class7) $>$ 30%

Weighted classes : Manually boosted the class 0,1,6,7. So the weights has been modified from [4.0579, 15.2193, 16.0435, 49.5676, 27.2547, 17.4573, 3.7309, 4.1093] to [9.0579, 40.2193, 16.0435, 49.5676, 27.2547, 17.4573, 8.7309, 7.1093].
With this method the model achieves an overall accuracy of 72% and and average accuracy of 72% as well.

3.3 Unet with dice-loss

Dice loss function was implemented on the U-Net architecture . It was not responding well to weighted class so this configuration was trained only with the augmented dataset strategy. Results are shown on figure 10. Overall Accuracy is lower than the former U-Net, reaching only 69%, but the average accuracy is improve to 74%. Jaccard and Dice indexes are both lower.

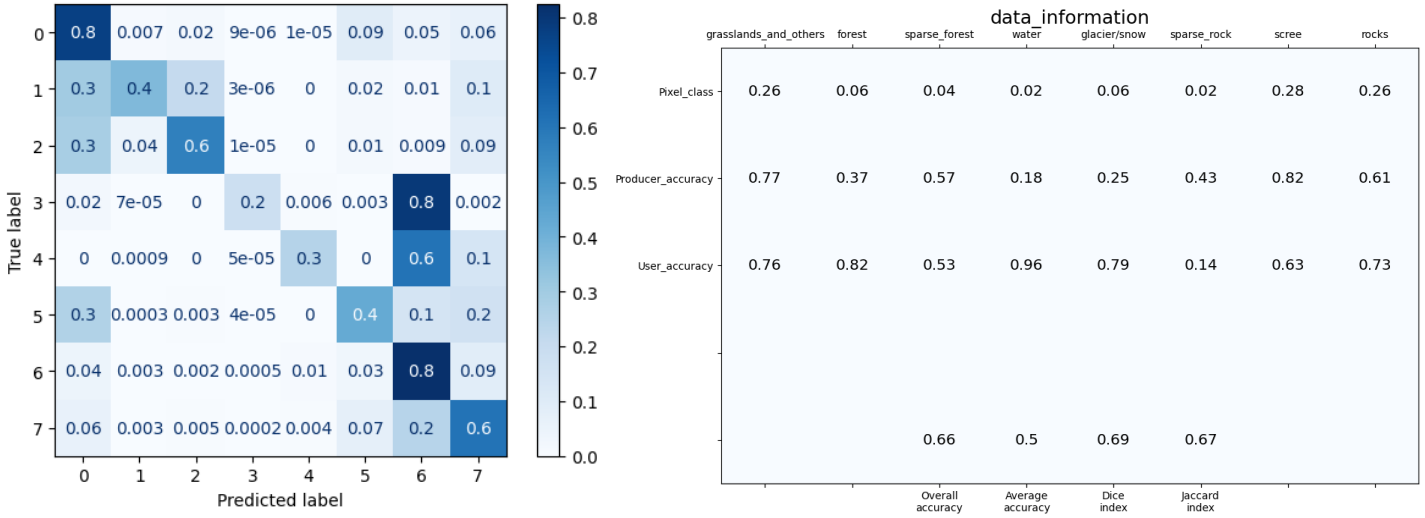


(a) Confusion matrix (b) Data information

FIGURE 10 – Results of the U-Net with dice loss on an augmented dataset

3.4 ResUnet

The ResUnet architecture is implemented directly with Weighted classes and data augmentation but not fine tuning. The result after 100 training epoch is given in figure 11. The accuracy of the prediction is quite low compared to the U-Net implemented. The prediction seems not to be able to discriminate between the different green and mineral parcels, respectively grass and the different forest types, and water, ice and rocks.



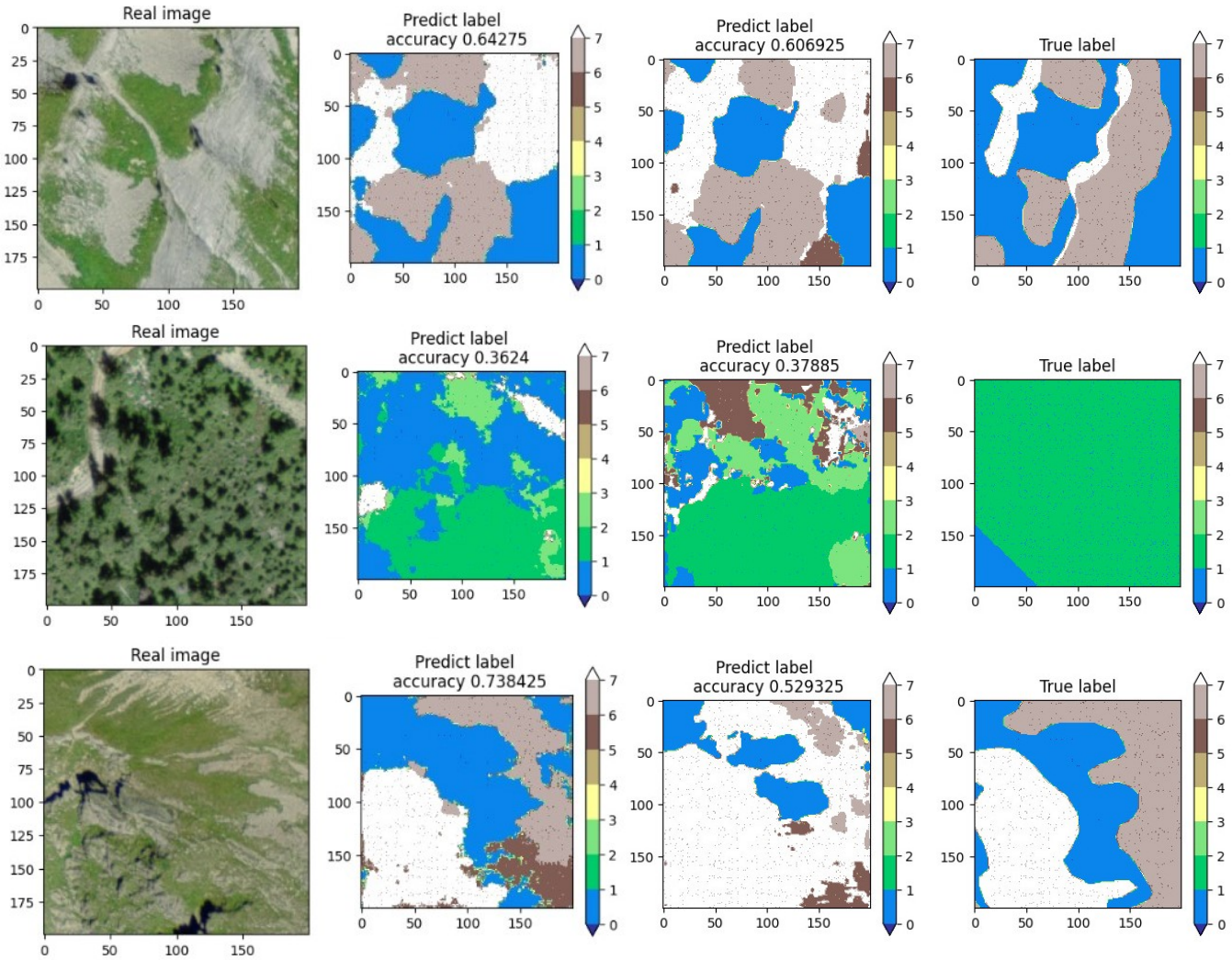
(a) Confusion matrix (b) Data information

FIGURE 11 – Results of the ResUnet with class weights on an augmented dataset

3.5 Summary

TABLE 2 – Summary of the results obtained with the different models. WC = Weigthed Classes, DA = Data Augmentation, DL = Dice loss function

Model	Overall Acc. [%]	Average Acc. [%]	Dice score [%]	Jaccard score [%]
Basic U-Net	**73**	63	**77**	**75**
WC + DA U-Net	72	72	74	72
DL + DA U-Net	69	**74**	71	69
WC + DA ResUnet	66	50	69	67



(a) U-Net fine tuned with Cross Entropy loss (b) U-Net with Dice loss

FIGURE 12 – Visual comparison of the predicted versus ground truth labels for both models.

4 Discussion

U-Net was the first model implemented and the best in term of accuracy and computational time. After the fine tuning of its parameters the Dice and Jaccard Index values are lower with this model than the first raw U-Net. It means the overlapping of the dataset is reduced. The overall accuracy is also reduced compared to the 73% of the raw U-Net model. Still the average accuracy is way better. The major driver for the model improvement is the weighing of classes. Data augmentation alone could considerably worsen the prediction, but put together with a fine tuning of the data augmentation and weights, the results are satisfying.
Regarding the dice-loss function implementations, the overall accuracy is not as good as the best results with weighted class. However, the average accuracy is slightly better, which makes it a better model to choose if the goal is to optimize the per-class accuracy. Introduction of weight class in this later architecture is expected to improve the prediction, still when we tried to implement it, the results were worst. Interestingly here, data augmentation only allows to obtains very good results, demonstrating that the tools to improve segmentation models are really dependant of their architecture, hence difficult to compare.
Another aspect that could have been tuned is the deepness of the models. However, the size of the images being not optimize to be divided more than 4 times, it is causing troubles to go deeper than 4 max-pools with a U-Net model.
Implementation of ResUnet is not successful either. The prediction seems somehow random, only managing to decide between green and mineral patches. No metrics was shown to be higher than both precedent U-Net models. This supports the statement that improvements to be made are quite unique to each model and it is difficult to predict how they would behave.
Surprisingly the Dice and Jaccard indexes are always worsen while the average accuracy increases. Even more surprising, when using the dice-loss function, the Dice score is even lower than with the Cross entropy loss, while it is expected to be the highest as the method tries to maximize it. When looking at figure 12 it is possible to argue that overlapping of the true and predicted classes is lower in the Dice-loss architecture model than the Cross-entropy loss one. In fact a greater part of the rocky images is not classified (white), explaining the lowered Dice and Jaccard indexes. Based on this hypothesis, the best model to obtain a good accuracy with most of the images classified in an existing class is the fine tuned U-Net, even if it does not appear to be the one with any maximum accuracy score.
Figure 13 shows multiple comparison of images in the dataset with their true and predicted labels using the fine tuned U-Net architecture. It shows the true labels do not describe well the true land cover on the image. The model built within this project is way more accurate. It segments finely the different land coverage zones, while the ground truth is mostly a big rough estimation of the area limits. This could explain the lowering of Jaccard and Dice index while the model gets better in term of per-class accuracy. This initial low quality of the ground-truth does not allow to precisely compute the accuracy of the predictions. In fact they must be better than the one computed using the given ground-truth.

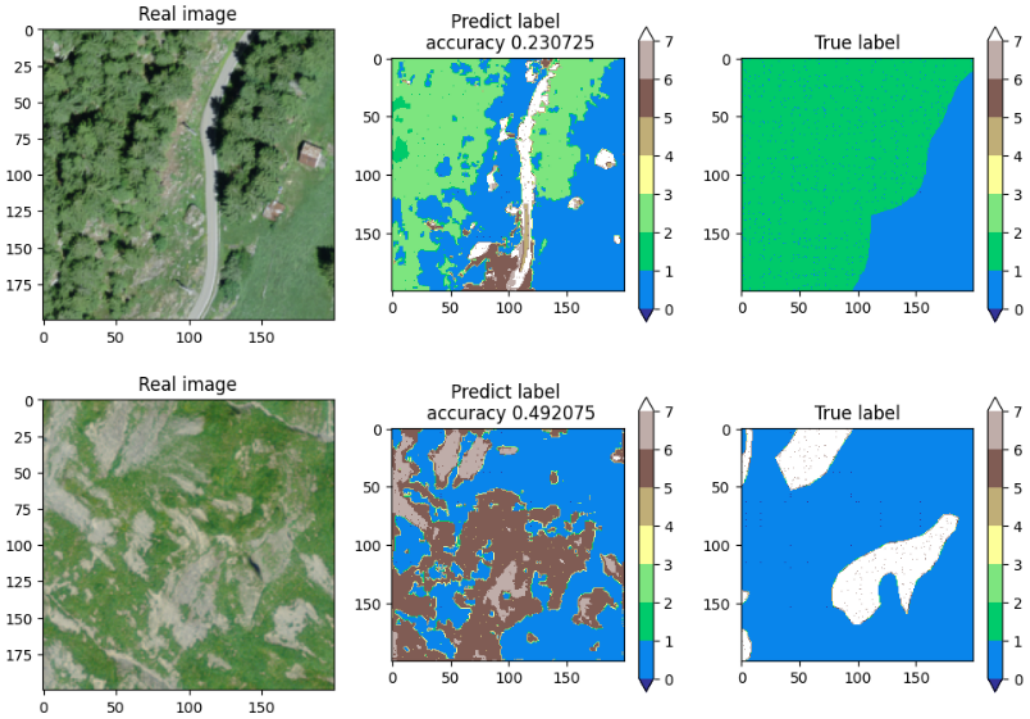


FIGURE 13 – Comparison between predicted and true label using U-Net architecture with weight and data augmentation after fine tuning.

5 Conclusion

The project proposed multiple solution for semantic segmentation task of alpine land. First results obtained with only normalization of the images in the pre-process highlighted the need of accounting to class representativity in the dataset and fix it in order to get less biased prediction. Over the two methods, U-Net shown the best prediction in term of overall and average accuracy. Per-class accuracy was at its best with the Dice-loss function and data augmentation method. With cross-entropy gradient the accuracy scores were higher for the basic than the tuned model, still the average accuracy is better in the later, resulting of a better prediction in term of under represented classes. Multiple metrics where used and compared. For this project confusion matrix and related accuracies were shown to be the better indicators, notably the Average accuracy.
Finally when comparing the predicted and ground truth labels, the real accuracy must be even higher than the one computed because to this low quality hand interpretation. Hence the use of Deep Learning for semantic segmentation of RGB mountainous images is demonstrated. It could be enhance in multiple way. Fine tuning could potentially be improved. One could reshape the images in order to allow the implementation of deeper models. Some other literature also use further information, like Digital Elevation Model (DEM) and geological maps. [4] [11] The use of others bands such as NIR could help further the models, notably to decide between forest and grassland. Still these later improvement would make the data-acquisition more expensive.

Références

[1] Mo, Y., Wu, Y., Yang, X., Liu, F., & Liao, Y. (2022). Review the state-of-the-art technologies of semantic segmentation based on deep learning. Neurocomputing, 493, 626-646.

[2] Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., & Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. arXiv preprint arXiv :1704.06857.

[3] Maxwell, A. E., Bester, M. S., Guillen, L. A., Ramezan, C. A., Carpinello, D. J., Fan, Y., ... & Pyron, J. L. (2020). Semantic segmentation deep learning for extracting surface mine extents from historic topographic maps. Remote Sensing, 12(24), 4145.

[4] Gao, L., Luo, J., Xia, L., Wu, T., Sun, Y., & Liu, H. (2019). Topographic constrained land cover classification in mountain areas using fully convolutional network. International Journal of Remote Sensing, 40(18), 7127-7152.

[5] Hoeser, T., Bachofer, F., & Kuenzer, C. (2020). Object detection and image segmentation with deep learning on Earth observation data : A review—Part II : Applications. Remote Sensing, 12(18), 3053.

[6] Zhang, X., Wu, B., Zhu, L., Tian, F., & Zhang, M. (2018). Land use mapping in the Three Gorges Reservoir Area based on semantic segmentation deep learning method. arXiv preprint arXiv :1804.00498.

[7] Zhang, Z., Liu, Q., & Wang, Y. (2018). Road extraction by deep residual u-net. IEEE Geoscience and Remote Sensing Letters, 15(5), 749-753.

[8] AI and Social Science – Brendan O'Connor, *F-scores, Dice, and Jaccard set similarity* `https://brenocon.com/blog/2012/04/f-scores-dice-and-jaccard-set-similarity/`

[9] Medium, *Metrics to Evaluate your Semantic Segmentation Model*, `https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2`

[10] Johnson, J.M., Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J Big Data* 6, 27 (2019). `https://doi.org/10.1186/s40537-019-0192-5`

[11] Petliak H, Cerovski-Darriau C, Zaliva V, Stock J. Where's the Rock : Using Convolutional Neural Networks to Improve Land Cover Classification. Remote Sensing. 2019 ; 11(19) :2211.`https://doi.org/10.3390/rs11192211`