Semantic segmentation of alpine land cover

Noah Kaltenrieder and Théo Damiani
EPF Lausanne, Switzerland

Github Repository: https://github.com/shoutizix/IPEO$_{Project}$

***Abstract*—This project has been part of the course Image processing for Earth observation, and aims to develop a model that performs semantic segmentation of satellite images. The images are from the Dents du Midi area, so this is alpine land. We implemented a model with a U-Net architecture and tried multiple losses, but the frequency of each class being really imbalanced leads to not convincing results. Thus we tried to perform transfer-learning, with the pre-trained model DeepLabV3 and it gave us the best result that we could get, e.g 98% accuracy for the least frequent class.**

I. Introduction

A. Task Description

The goal of this project is to develop an automatic model for the segmentation of satellite images. Indeed, the creation of segmentation labels is a tedious task that is nowadays often done by hand. Moreover, due to the increasing amount of data available, an automatic method for producing topographic maps covering vegetation and terrain would be very useful for research in environmental sciences for example.

Based on Deep Learning, the model aims to perform semantic segmentation of alpine land cover. The main challenges of the project are the uneven distribution of classes in the dataset and the high similarity of pixels between some different classes.

B. Data

The dataset[1] contains 12'262 satellite images from the Dents du Midi area, Swiss Valais. Each image is associated with a label annotated by the Federal Office for Topography Swisstopo. The labels are divided into 8 classes, which means that each pixel of the 200x200 label is the index, from 0 to 7, of the corresponding class. As for the images, they are of form 3x400x400, there are 3 channels, Red, Green, and Blue. The dataset is also already divided into training, testing, and validation sets of 60%, 30%, and 10% of the images.

We had to delete the image with the id '25595_11025' because we could not open it due to file corruption.

As it was said in the Task Description, we have noted that the dataset has a highly imbalanced frequency of class, with three classes being represented with an approximately 25% frequency, and the other classes are between 2% (for the Water and wetlands) to 6%.(see Figure 1).

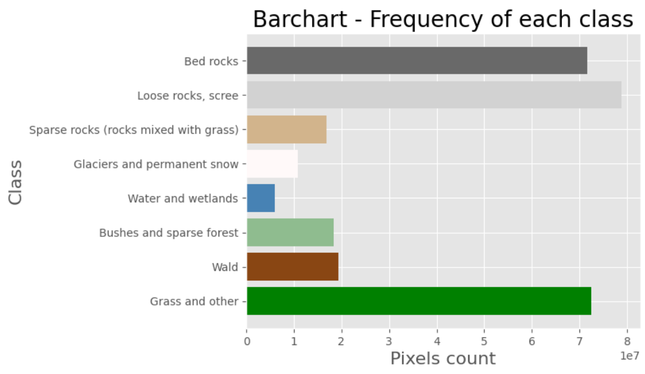

Figure 1. Frequency of each class in the training dataset.

II. Models and Methods

A. Image Preprocessing

We performed, at first, a sanity check on the training dataset and we observed that some all-black images had incoherent ground truths. (see Figure 2).

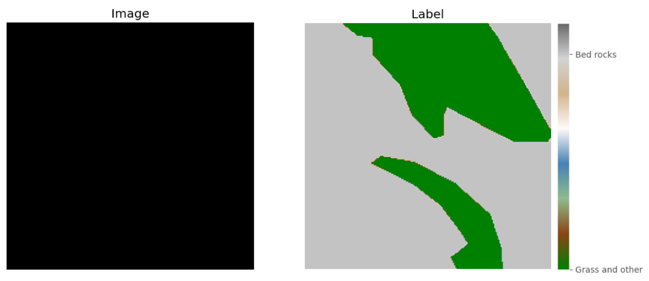

Figure 2. An all-black image that has an incoherent ground truth label.

In fact, we have found 11 images that were in this case and we, therefore, have chosen to remove them for the training phase.

We also computed the mean and the standard deviation per channel (R,G,B) for all the images on the training dataset, to

[1]https://enacshare.epfl.ch/drCz5HgLJyFPXifNBWad7

normalize them when loading them for the training phase. Normalization is a good practice in Deep Learning, it puts the features on the same scale as each other and it improves the performance and stability of the training. After the normalization, we recomputed the mean and the standard deviation to be sure that they were close to 0 for the mean, and close to 1 for the standard deviation, which is the desired result.
Since the training dataset is very imbalanced concerning classes, we performed some data augmentation to increase the frequency of underrepresented classes. We have performed 3 rotations of 90° on each image that contains the underrepresented labels, so a vertical flip, a horizontal flip, and a horizontal + vertical flip. (see Figure 3). We have chosen these operations because it doesn't affect the ground truth label, for example, if we had performed a rotation of 45°, then the label would have introduced some non-determined pixels in the corners of the label and the interpolation would also have brought errors.

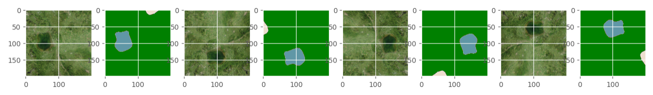

Figure 3. Example of the data augmentation with the different flips performed

We then computed the new repartition of the classes (see Figure 4). After the data augmentation, the repartition was better balanced, only the "grass and other" class still has a frequency of around 25%, but the frequency of the underrepresented classes is nearly 1.5x bigger than before. For example, the "Water and wetlands" went from 2% to 3% and the "Wald" from 6.6% to 10%.

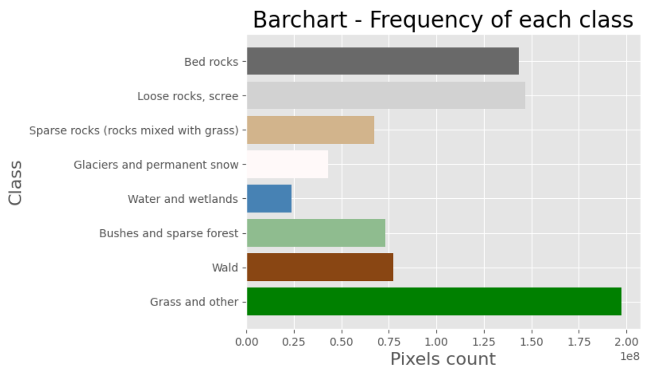

Figure 4. New frequency of each class in the training dataset after the data augmentation.

B. Deep Network Structure

The state of the art in semantic segmentation is the use of a U-Net. Thus, we used a U-Net with 5 layers: 64, 128, 256, 512, and 1024. For the implementation, we followed Figure 5 and we took help from the GitHub repository "U-Net: Semantic segmentation with PyTorch"[2].

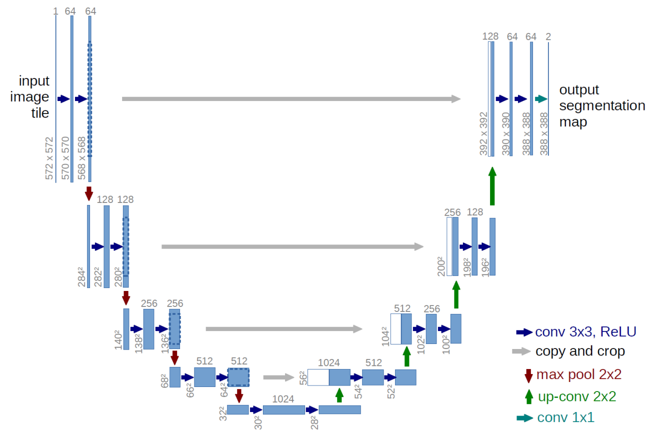

Figure 5. The architecture of the implemented U-Net[1]

At the optimizer level, we started with a Stochastic Gradient Descent and classical values: a learning rate of 1e-3 and a momentum of 0.9.

C. Loss Function

The main challenge of this project is the semantic segmentation of low frequency classes. Thus, after our improvements in data preprocessing for this problem, we decided to test several loss functions:

- **Cross Entropy Loss**: This loss function is the main one used in U-Net classification per pixel problem[2]. $l_n = \log \frac{exp(x_{x,y_n})}{\sum_{c=1}^{C} exp(x_{n,c})}$ This function is simple but can create errors with infrequent classes.

- **Weighted Cross Entropy Loss**: The idea of this function is to weight the loss by respecting the proportion of each class in the dataset[3]. The loss will be more important for the less frequent classes and will have less weight in the final calculation for the very represented classes. To calculate the weights we chose the ratio of the median of the class frequencies and the frequency of the class: $w_c = \frac{\sum_{c=1}^{C} F_c}{F_c}$

- **Focal Loss**: This loss function is a variation of the Cross-Entropy. It is especially good for imbalanced dataset[4], because its specificity is to down weight easy examples to focus more on learning the hard ones.
$FL(p_t) = -\alpha_t (1 - p_t)^\gamma log(p_t)$

[2]https://github.com/milesial/Pytorch-UNet

- **Intersection Over Union**: This function is defined by the ratio between the intersection of predictions and truth and their union. The intersection is defined by the sum of the correct predictions for an image and the union is the sum of the predicted labels and ground truth labels minus the intersection. We followed the implementation of this repository github[3].

III. Results

Unless otherwise specified, the models were run on 100 epochs with a batch size of 32 images and a Stochastic Gradient Descent optimizer with a learning rate of 10^{-3} and a momentum of 0.9.

- **Cross Entropy Loss**: This loss function is expected to perform the best when there is an equal distribution[4], which is not the case for us. We still tried training with it, but only on 75 epochs, because we saw that it was already overfitting the training dataset, as we can see in the Figure 6 with the validation loss increasing a lot, and the accuracy staying near 65% on Figure 7, so there won't be any improvement with more training. It took approximately 4 hours to train with this loss. It can be observed in Figure 8 that the per class accuracy is really good compared to other methods.

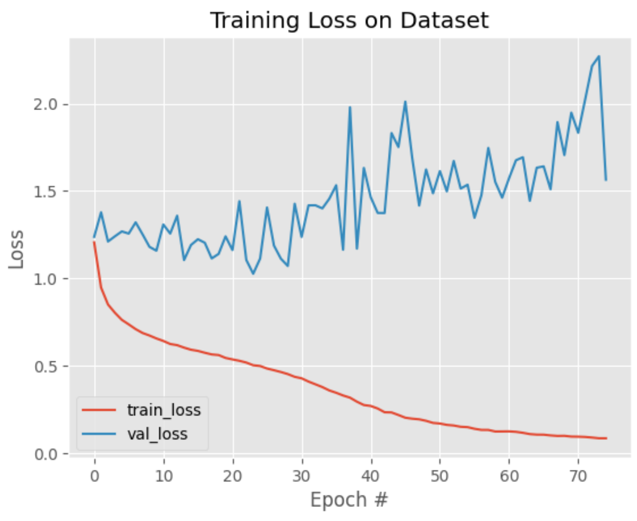

Figure 6. Training and validation Cross Entropy Loss over 75 epochs.

[3]https://github.com/amirhosseinh77/UNet-AerialSegmentation/blob/main/losses.py

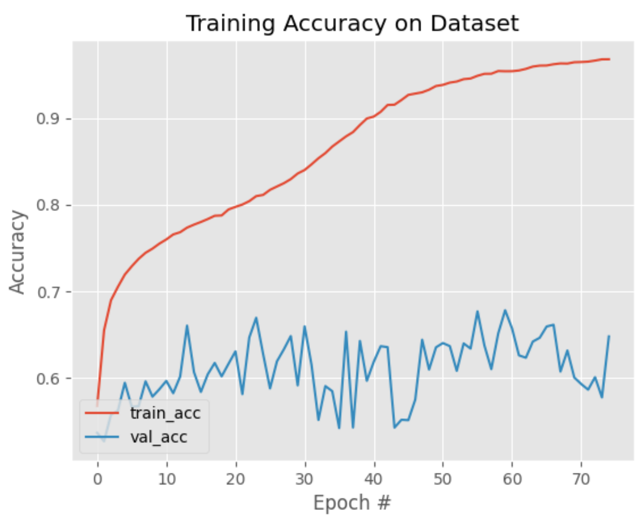

Figure 7. Training and validation accuracy over 75 epochs (CE).

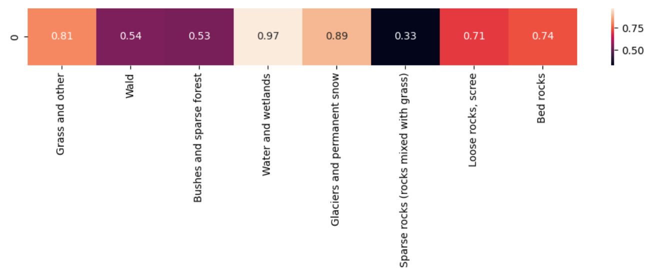

Figure 8. Per class accuracy on the test set. The U-Net model trained with Cross Entropy Loss.

- **Weighted Cross Entropy Loss**: With this loss function, the small classes are quite present in the predictions, Figure 9. The model tries to define many small regions as infrequent classes. However, the accuracy or loss of the validation set has a high variance (Figure 10 and 11), so the model may suffer from overfitting in the first 10 epochs. The accuracy by class is disappointing, only 45% for the most frequent class and less than 1% for the least frequent, Figure 12. The training time was approximately 7 hours for 100 epochs.

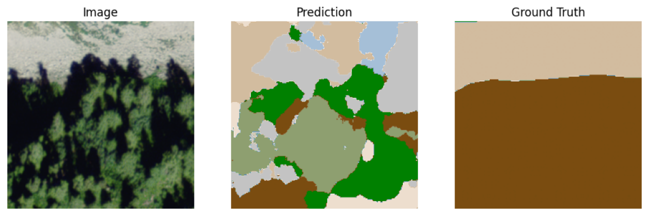

Figure 9. Example of prediction with the model trained over Weighted Cross Entropy Loss.

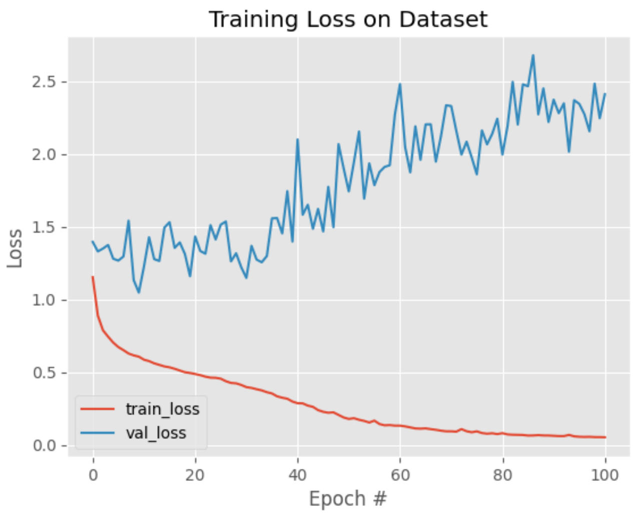

Figure 10. Training and validation Weighted Cross Entropy Loss over 100 epochs.

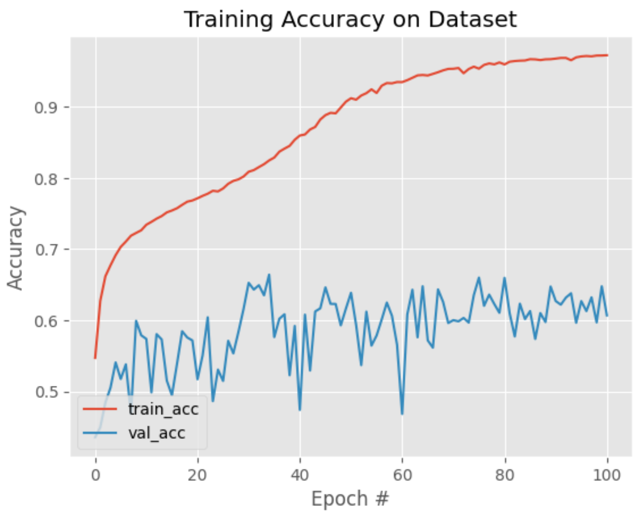

Figure 11. Training and validation accuracy over 100 epochs (CEW).

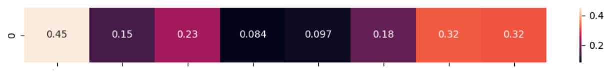

Figure 12. Per class accuracy on the test set, from left to right: Grass..., Wald, Bushes..., Water..., Glaciers..., Sparse rocks, Loose rocks, Bed rocks.. Model trained with Weighted Cross Entropy Loss.

- **Focal Loss**: We trained our model with a focal loss so that the model would better learn the hardest class. We used $\gamma = 2$ and no α parameter. The training was performed for 100 epochs and it took between 4 to 5 hours. The training curves obtained were similar to the ones from Weighted Cross Entropy, with the increase of the loss after ~ 40 epochs, Figure 13, and an accuracy stagnating near 65% on the validation dataset (as observed on Figure 14). The accuracy per class obtained with this loss method is similar, but a little bit worse than the one from Weighted Cross Entropy Loss, Figure 15.

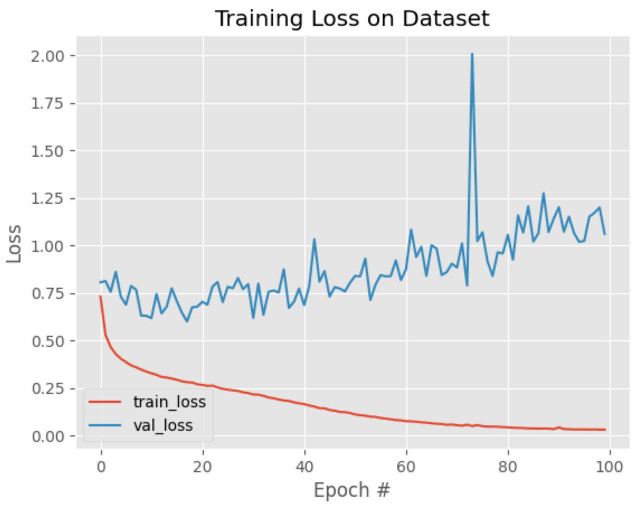

Figure 13. Training and validation Focal Loss over 100 epochs.

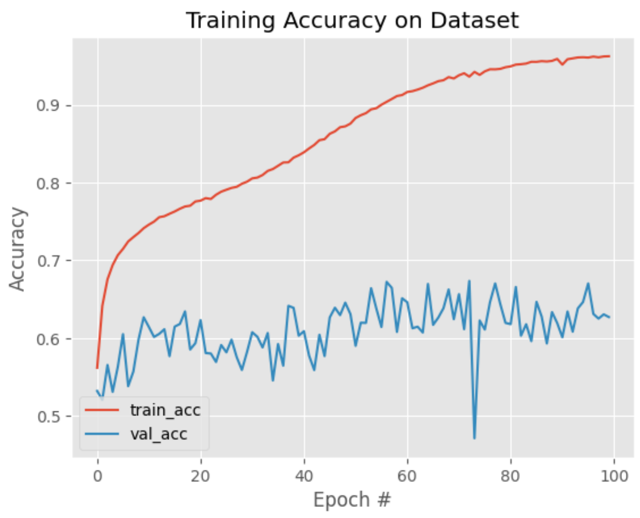

Figure 14. Training and validation accuracy over 100 epochs (Focal Loss).

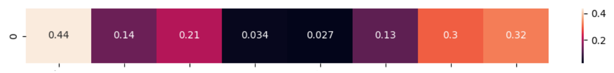

Figure 15. Per class accuracy on the test set, from left to right: Grass..., Wald, Bushes..., Water..., Glaciers..., Sparse rocks, Loose rocks, Bed rocks.. Model trained with Focal Loss.

- **Intersection Over Union**: IoU loss gives similar results to weighted cross-entropy loss or focal loss in terms of accuracy per class, Figure 18. Compared to Figure 10, It is less clear in Figure 16, if the model suffers from overfitting. The loss converges to 0.88 but does not increase again after the first 20 epochs. The model was trained for 6 hours for the 100 epochs.

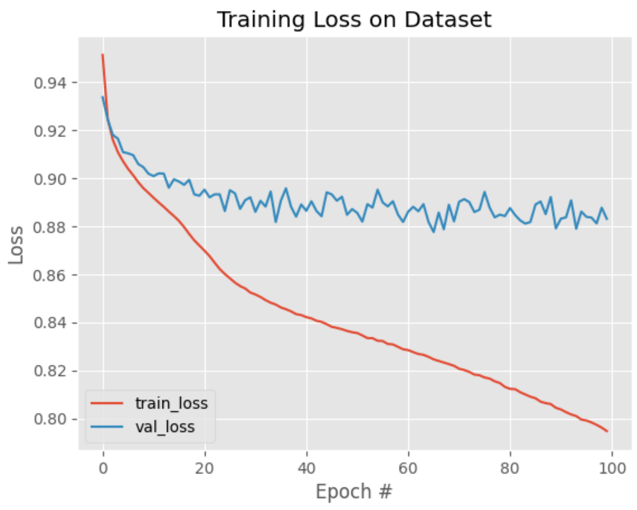

Figure 16. Training and validation IoU Loss over 100 epochs.

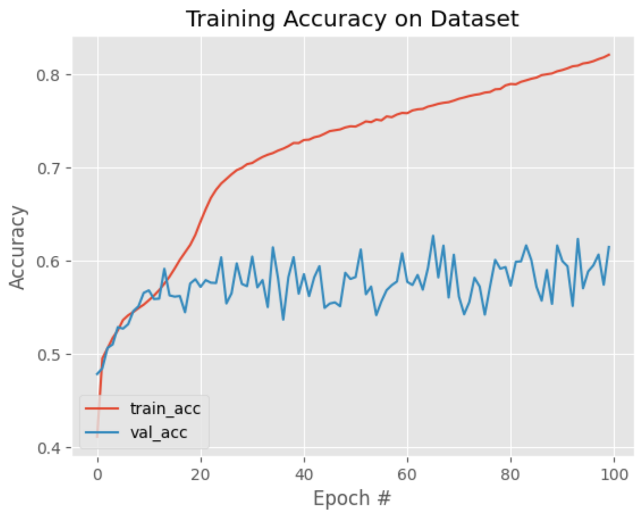

Figure 17. Training and validation accuracy over 100 epochs (IoU).

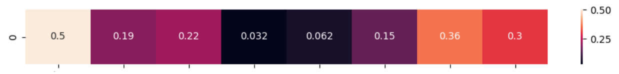

Figure 18. Per class accuracy on the test set, from left to right: Grass..., Wald, Bushes..., Water..., Glaciers..., Sparse rocks, Loose rocks, Bed rocks.. Model trained with IoU Loss.

As our results were not satisfactory, we decided to try two new methods:

- **DeepLabV3**: Training a model from scratch can be tedious to get good accuracy. So we tried to fine-tune a pre-trained model. We have chosen DeepLabV3[5]. It is a semantic segmentation model designed and open-sourced by Google. It seems to be much more complex than our UNet, and it brought good results in other projects of land cover semantic segmentation[6]. To use this model we just change the last layer of classification so that it has the right number of classes. We have performed transfer learning and tried to fine-tune the pre-trained weights of the model by training it for 50 epochs with a learning rate of 10^{-3} for the first 25 epochs and then 10^{-4} to avoid overfitting for the last 25 epochs. It was way more time consuming to train this model : $\sim$ 10 min for 1 epoch when training on GPU on SCITAS. Thus the training lasted for a bit more than 8 hours. We obtained pretty similar result as with the other methods during the training, but it was more stable and the loss was still slowly decreasing, see on Figure 19 and Figure 20. The predictions seem more accurate than on Figure 9, as we can observe on Figure 21. This method gave pretty good accuracy for each class, see Figure 22. Even the underrepresented classes achieve a good accuracy, it can be explained because the model knows already well the features to extract from the image.

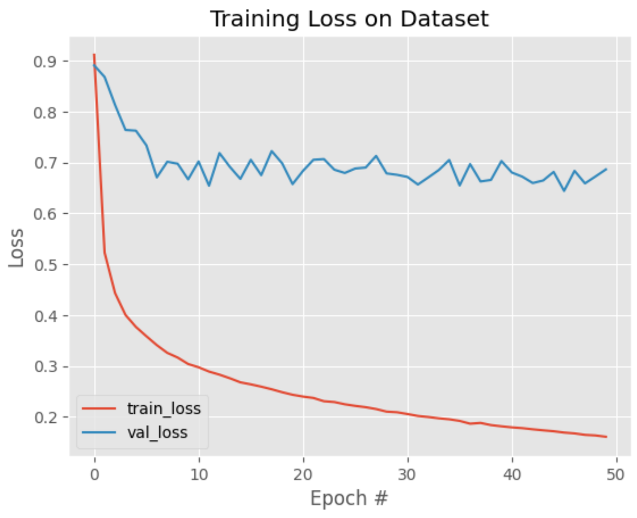

Figure 19. Training and validation Focal Loss for DeepLabv3 over 50 epochs.

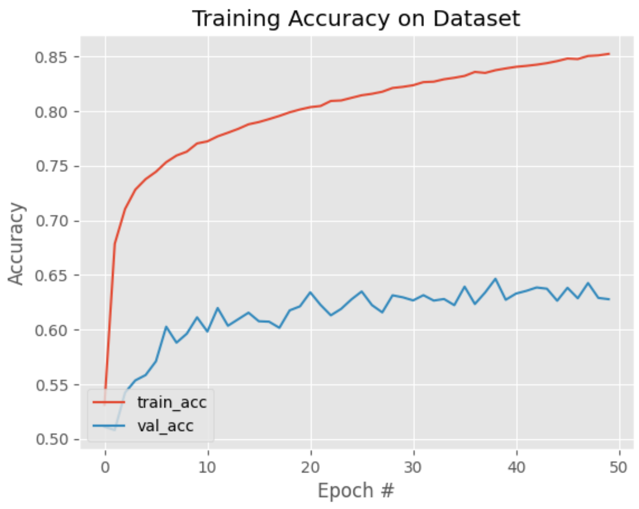

Figure 20. Training and validation accuracy for DeepLabv3 over 50 epochs.

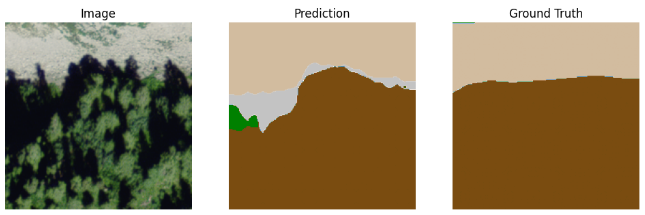

Figure 21. Example of prediction with the DeepLabv3 model trained with Focal Loss.

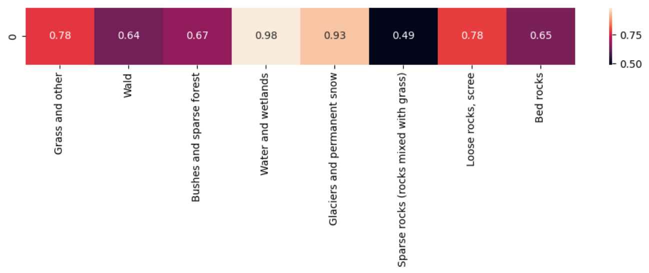

Figure 22. Per class accuracy on the test set. DeepLabv3 model trained with Focal Loss.

- **Smaller Model**: Our model may overfit the train set, so we tried to generalize the model. To do this, we have reduced the complexity of the model so that there are fewer parameters (weights and biases), so that it will not be as accurate for the train set. We have simplified the layers of the UNet: the encoders now have a number of channels of 32, 64, 128, and 256, instead of 64, 128, 256, 512, and 1024. In the Figure 25, we can see that results are better than the full model with Weighted Cross Entropy Loss or Focal Loss. But the validation accuracy, Figure Figure 24, has a much higher variance over 100 epochs than our other training like Figure 14 or 19. So maybe we just got lucky on this run. The duration of the training was about 5 hours and a half.

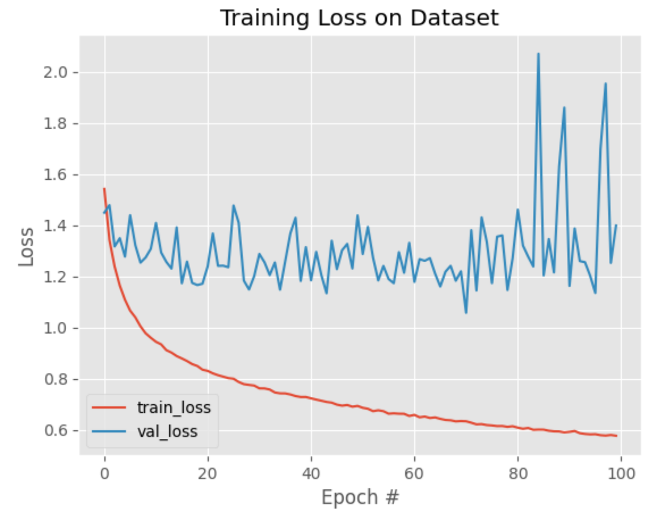

Figure 23. Training and validation loss with the reduced model with Cross Entropy Weighted over 100 epochs .

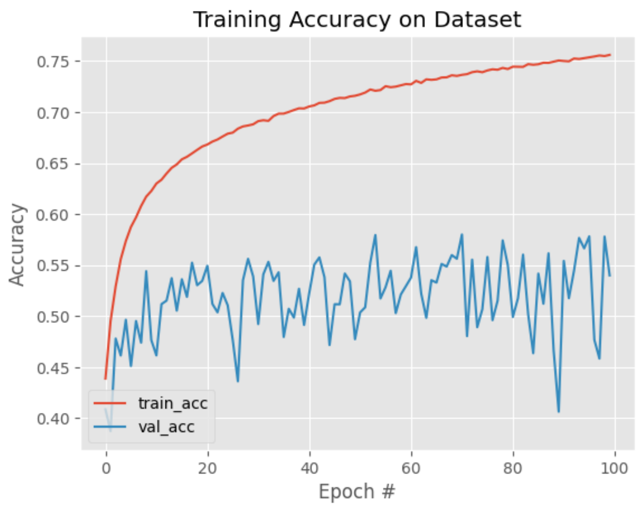

Figure 24. Training and validation accuracy with the reduced model with Cross Entropy Weighted over 100 epochs .

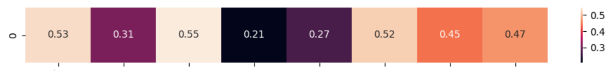

Figure 25. Per class accuracy on the test set. Reduced model trained with Cross Entropy Weighted Loss.

- **Restricted Dataset**: So far we have tried to solve the problem of unbalanced labels by the model architecture, the loss function, or the hyperparameters without much success. So we decided to try to restrict the dataset to make the classes fairer. So for each labeled image, we calculate the percentage of each class present in the image. Then if one of the 3 most frequent classes (Grass, Loose rocks, and Bed rocks) is represented by more than 50% in the image, we do not take into account the data in the training set. This greatly reduces the training set but we rely on data augmentation to keep the set large enough. The training dataset is composed of about 12’000 images, with the increase in data augmentation we have about 19’000 images but we fall to 8’751 images with this restriction. The model has difficulty in converging to a good accuracy, it remains stuck around 0.5 in validation, Figure 27. Maybe the threshold is too high and it is better to keep as many images as possible even with unbalanced classes. This training was the fastest, only 2 hours for 60 epochs.

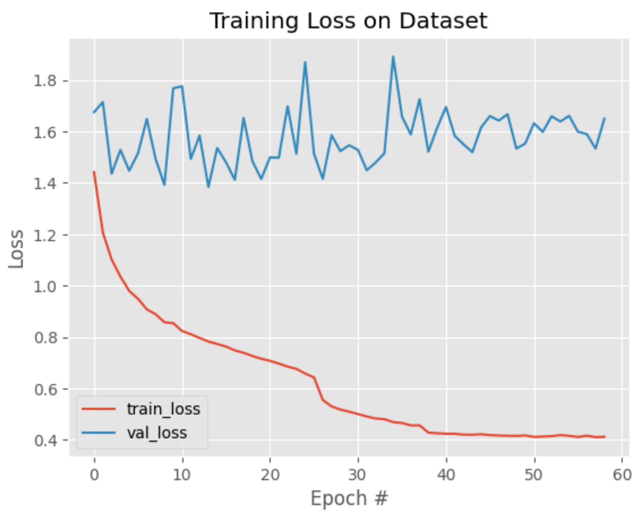

Figure 26. Training and validation accuracy over the restricted training dataset with Cross Entropy over 100 epochs .

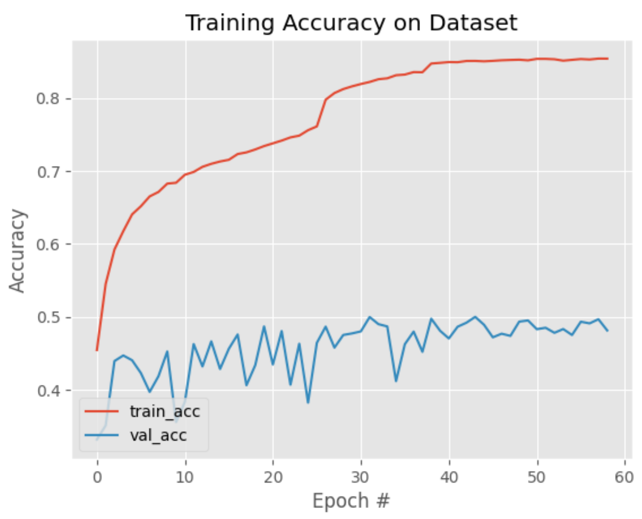

Figure 27. Training and validation accuracy over the restricted training dataset with Cross Entropy over 100 epochs .

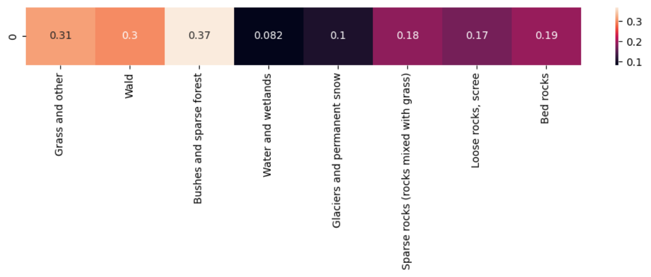

Figure 28. Per class accuracy on the test set. Model trained over the restricted dataset with CE loss.

IV. Discussion

The biggest problem encountered during the development of our model is the inequality of the class frequencies. Either our model can learn correctly the 3 most frequent classes (Grass, Loose rocks and Bed rocks) or it tries to compensate for this unbalanced dataset but the model suffers from overfitting, it predicts a bit of all class for each image without consistency. Another problem that we have encountered is that some ground truths seem inaccurate and our model predicts the right segmentation but is still penalized because of the wrong ground truth, see Figure 29.

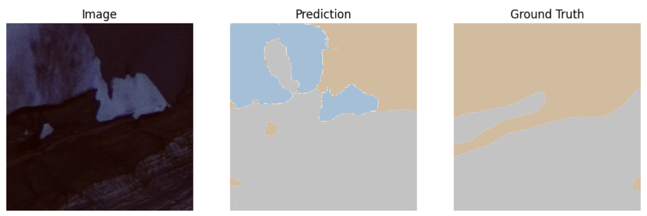

Figure 29. Example of prediction with the model trained over Cross Entropy Loss. There is clearly snow on the image and the model classified it correctly but the ground truth is inaccurate.

We think that a good approach would be to train the DeepLabv3 with other losses and compare them to achieve the highest accuracy possible for each classes. We also think that one could fine-tune the 'loss' parameters to achieve better results, for example, we have only tried 1 value of γ for the Focal Loss. These parameters could be determined by cross-validation. Also we could try other optimizers, like Adam and also add a learning rate scheduler. Finally, we have found that DeepLabv3 can be promising with transfer-learning for this semantic segmentation task, at least this method gave us the best result.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[2] D. Tuia, D. Marcos, K. Schindler, and B. L. Saux, "Deep learning-based semantic segmentation in remote sensing," pp. 46–66, Aug. 2021. [Online]. Available: https://doi.org/10.1002/9781119646181.ch5

[3] T. Sugino, T. Kawase, S. Onogi, T. Kin, N. Saito, and Y. Nakajima, "Loss weightings for improving imbalanced brain structure segmentation using fully convolutional networks," *Healthcare*, vol. 9, no. 8, p. 938, Jul. 2021. [Online]. Available: https://doi.org/10.3390/healthcare9080938

[4] S. Jadon, "A survey of loss functions for semantic segmentation," 2020. [Online]. Available: https://arxiv.org/abs/2006.14822

[5] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017. [Online]. Available: https://arxiv.org/abs/1706.05587

[6] S. Manzanarez, V. Manian, and M. Santos, "Land use land cover labeling of GLOBE images using a deep learning fusion model," *Sensors*, vol. 22, no. 18, p. 6895, Sep. 2022. [Online]. Available: https://doi.org/10.3390/s22186895