Challenge 2

Multiclass Segmentation

Matteo Pacciani (Codice Persona 10525096 - Matricola 945104)
Oscar Pindaro (Codice Persona 10568511 - Matricola 946671)
Francesco Piro (Codice Persona 10534719 - Matricola 946450)

1 Problem Approach

We decided to start this challenge adapting the notebooks seen during lectures to our task. We used a k-fold cross validation approach, exploiting KFold from Scikit-learn. This module provides a ".split()" method, which randomly creates all the specified folds, based on a seed. Our first architecture was a simple encoder made of convolutions and maxpooling, followed by a simple decoder, with upsampling layers and convolutions. Since it had very poor performances, we replaced the encoder with the VGG16 architecture, without the dense layers. We discovered that squared pictures of 256x256 pixels brought to a huge loss of information when resizing, thus we progressively increased the size up to 1152x1536 and we obtained 0.56 meanIoU on the whole dataset. Still, this size was too arbitrary to account for all the different sizes, therefore we cropped all the images into 512x512 tiles, without overlapping. With this architecture, tiles didn't improve our score.

2 Models Architectures

We then decided to implement a U-Net like architecture, with VGG16 as the encoder. This implementation involved the concatenation of layers from the encoder into the decoder, followed by a merge through a convolution. We also added batch normalization layers and dropout in the decoder. This obtained a .64 global IoU on the test set, and all the Pead images were excluded during training. We decided to try the same architecture without tiles but with a weighted categorical cross entropy loss, and it obtained .65 but with quite different team scores from the previous try. For example, the "Roseau" team went above .6 IoU (and it was at about .4 before).

At this point, we decided to try to train the worst performing teams singularly. Having less data, we chose lighter architectures, still U-net shaped. This was able to obtain higher scores on every team, excluding Roseau, but just for haricot crop. We were able to improve the worst score, that was on Pead Maize, thanks to an even lighter architecture, with less than a million parameters (the previously described U-Net had instead 4 millions while VGG U-Net had 11 millions trainable). In these U-Nets, we had to use "same" padding, because we wanted to avoid resizing at the end, to obtain the original size of the images. We tried using simple upsampling2D layers with nearest interpolation, but transpose convolution turned out to get higher scores.

3 Weighted Loss

For what concerns the weighted loss, we thought it was necessary because all the architectures were able to understand what was not background, but struggled when deciding if a pixel was crop or weed. Thus we initially tried setting the weights according to the inverse frequency of occurrence for all of our three classes, but this led to highly unbalanced weights and worse performances. So we made them slightly different from each other: 0.5 for background class, 1 for crop class, 2 or even 1.5 for weed class, and this improved all the architectures scores, especially on weed IoU.

We tried an alternative to this balancing method, filtering the tiles according to a threshold of "interesting" pixels, that were those belonging to crop or weed class. Setting this threshold to 6000 pixels (tiles were 512x512), and thus removing tiles with less than 6000 "interesting" pixels, led to a speed up in training and a slight improvement of performances. Filtering tiles even more instead gave us lower scores, because a lot of tiles with small groups of weed were deleted in this way.

4 Conclusions

Finally, we tried a learning rate scheduler, that forced the Adam optimizer (that anyway should automatically adjust the learning rate) to stay under a threshold, decided by us. This threshold simply decayed of 0.1 every 7 epochs. This improved our performance. At the end, since at prediction time we knew exactly the team and the crop of each image, for example "Bipbip Haricot", we chose the best model for the prediction of that particular pair, and of course this gave us a better global IoU.

5 Reproducibility

To reproduce our results, you need to first train all the models with the notebooks `unet_ADD_SKIP`, `UNET_PEAD_MAIS` and `UNET_VGG`, and then generate the json file with the `build_best_submission` notebook. Alternatively, you can download all the checkpoints from this link and load them in `build_best_submission` notebook: https://drive.google.com/drive/folders/1T7Xy-Zm4LimCX3a8l6zFbZK2Co-_sZCN?usp=sharing