Good crop, bad weeds segmentation

Davide Brescia
Matteo Ilari
Francesco Montanaro

1. Data loading

The first approach for data loading was to take as dataset all the images and their relative masks of each team and each vegetable (haricot/mais) without any distinction, as shown in the *alternative_data_loading.py* script.
After noticing that performances on the global meanIoU were not so relevant, another approach focused only on specific teams and specific vegetables has been chosen, as shown in *crop_weed_segmentation.ipynb*. After an accurate phase of tuning on which ones of the teams and crops could bring us to better performances, a dataset with Haricot as vegetable, excluding the team 'Pead' (since it shows very different images with respect to the other datasets), has been chosen.

Tiling

The first approach used was **resizing images**. Despite it brings good results on validation dataset, very low performances are obtained on test submission as was expected, indeed during testing, an upscale of the predicted mask is needed, losing many details.
The solution was using fully convolutional NN or using **tiling**. After having done a fast trial with FCNN, a second solution has been chosen: writing all the functions to perform the task from scratch.
During the training phase, images from the training dataset are randomly cropped and augmented (with their correspondent masks). An alternative was to create another big dataset composed of all the cropped images. It has been preferred the first approach because it seemed better from the 'overfitting' point of view.
Validation images are obtained in the same way, through *valid_size_multiplier* parameter the model can be evaluated with many random crops of each image from the validation dataset, and not just one.
During the testing phase, each image from the test dataset is cropped (with the possibility of overlap) and processed by the NN, then the original size mask is reconstructed just doing the sum of each predicted mask class layer and finally the argmax.
Usually an overlap = 10 (pixels) to mitigate border glitches has been set.
Since images were not perfectly divisible by the crop dimension (considering also overlap), the right - bottom sides of the image present larger overlaps. A function has been written to compute the proper overlap value not to have this asymmetry, it has been rarely used because overlap value couldn't be too large (increasing computing time), as shown in the *dynamic_overlap.py* script.
All the models shown in the notebook are trained with tiling.

2. Experiments

General approach

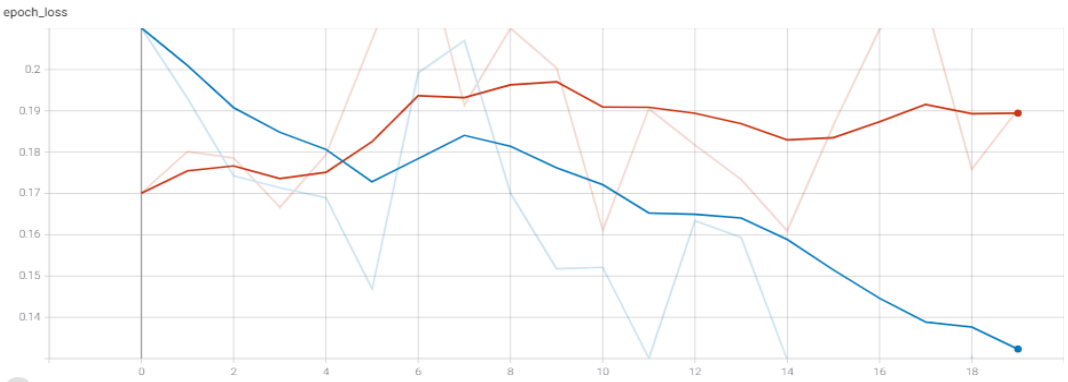
The models shown are the ones which have been considered the most promising. The final hyperparameters (shown in the notebook) have been chosen after a long process of trials and errors.
At the beginning, to solve **class unbalance** issue, cropped images containing just background were discarded. Unluckily, this slightly decreased the performance, so it hasn't been adopted.

Experiment 1

In the first experiments, it has been tried to build a model from scratch by using a classical "encoder-decoder" architecture.

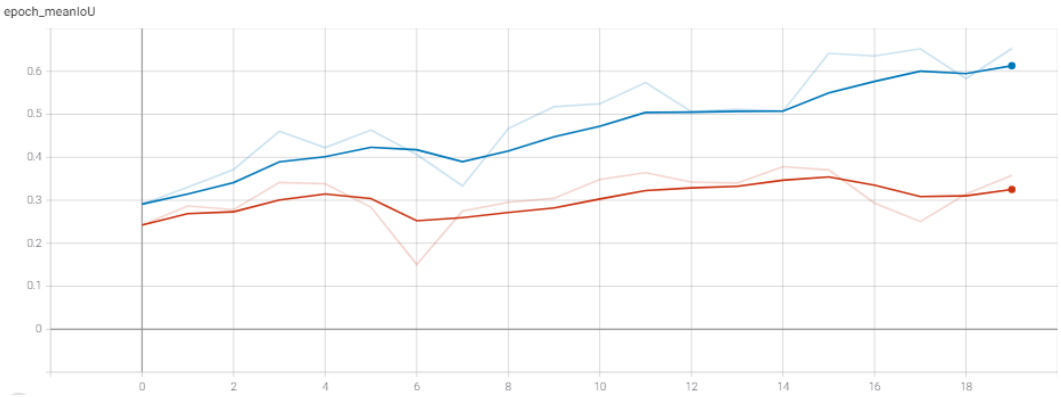
Here's the final training trends of the best hyperparameters we found:

loss:



Early stopping, monitoring loss, interrupted the training at this point.

meanIoU:



Light blue: training
Red: validation

Performance achieved is:

- global IoU ***0.3345***
- Weedelec Haricot IoU ***0.4756***

By inspecting single predictions it was clear that the segmentation was a little bit coarse, indeed this model doesn't include skip connections, which could increase the details in the output mask; nevertheless, this is the best model on the test dataset.

Experiment 2

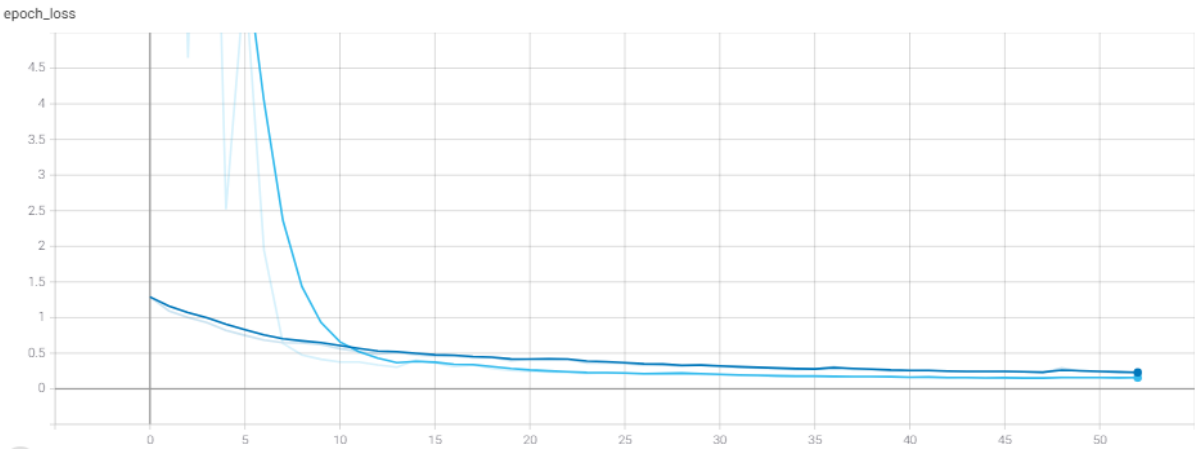
This experiment was conducted by using the transfer learning approach on a classical "encoder-decoder" architecture. In particular, a VGG16 model, pre-trained on the "imagenet" dataset, has been chosen for the features extraction process (encoder), so a decoder has been added at the end of the chain.

Achieved global meanIOU on test set: ***0.3135***.

Experiment 3

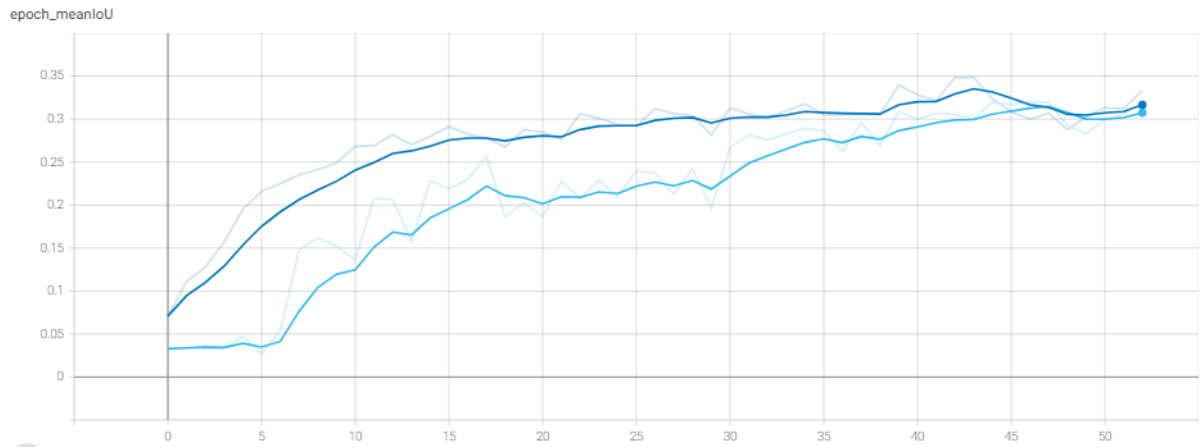
In the last experiment a custom U-Net architecture has been implemented and trained from scratch. However, the obtained results were no better than those of the previous models, with a global meanIOU equal to ***0.2776***.

loss:



Light blue: validation
Blue: training

meanIoU:



Light blue: validation
Blue: training

Second Phase

In the second phase, the best performing model has been used, i.e. the one described in the “Experiment 1” section. The only differences with respect to the old notebook are on the Data Loading part in which an adjustment of the code to properly load

the data from both the old training dataset and the new Test_Dev dataset, has been done.