Artificial neural networks and deep learning

Homework 2: Image segmentation

Motta Dennis Ostrovan Eugenio

December 27, 2020

1 Initial approach

The first step in approaching the homework was to analyze the data. Due to variety in the size and content of the images of the different teams we decided that we couldn't use all the images together to train a single model, so we focused on Bipbip/Haricot with the goal of finding a good solution to the problem and then apply it to the rest of the dataset. This has also the advantage of making the training of models faster allowing for less expensive experimentation.

2 Building a prototype

We built a first prototype, focusing on producing a valid submission. Initially we adapted the model from the practice session notebooks. We quickly implemented what we learned during the first homework: data augmentation, hold-out validation, Tensorboard visualization, checkpoint saving, early stopping and a callback for reducing the learning rate on the plateau.

3 First model

We understood that to have better performance we would need a more complex model so we built one using VGG16 as an encoder and created a decoder ourselves. Some shortcomings of this solution were that the model couldn't learn the shape of the plants very accurately (it predicted blobs instead of accurate plant shapes) and it was very insensitive to small patches of crop and especially weed. The latter problem was due to the fact that we were resizing the images in the data generator, losing much information in the process, especially small details. In this phase we achieved a test accuracy of 0.52 on the IoU Bipbip parameter.

4 U-Net

Since the model was not producing accurate results we tried to solve the problem by using a more complex model: we introduced U-Net that with its skip connections can preserve more information from the original image (the shape of the plants). Unfortunately the initial U-Net model had difficulty learning due to its very big size (30 million parameters) and the small size of the dataset (due to our resizing the images). To solve this issue we used a different U-Net implementation which uses VGG16 as the encoder. This allowed us also to use transfer learning on the encoder part by using the "imagenet" weights. In this phase we achieved a test accuracy of 0.70 on the IoU Bipbip parameter.

5 Combining Haricot and Mais images

While experimenting we noticed that any model that was trained on just the images of one crop of one team (Bipbip/Haricot for example) then had good performance on the photos of the other crop of the same team (Bipbip/Mais) without having been trained on them. We think training transfers well between different crops because the model learns small scale features that are common to different plant species. This is also helped by the images being very similar in scale, colour, perspective. For this reason we trained one model for each team on a dataset composed of images of both Haricot and Mais.

6 Image tiling

We implemented image tiling in order to take full advantage of the provided data. We cut the images of both the training set and the test set in 512x512 tiles (without overlap). We trained the model and made predictions on 512px tiles and then reassembled the full size test image prediction from which we can generate a submission file. In this phase we achieved a test accuracy of 0.72 on the Global IoU parameter, with 0.77 on IoU Bipbip, 0.60 on IoU Pead, 0.72 on IoU Roseau and 0.75 on IoU Weedelec.

7 Custom loss

We developed a custom loss function in order to give different weights to the weed and crop classes, because the dataset is unbalanced. Our custom loss is simply the sparse categorical cross-entropy which is weighted by the IoU metric in which we assign weights to classes. This helped better identify the weed class in models in which we resize the images, but very small details are still lost. Overall the use of custom loss did not lead to significant improvements, especially since tiling allows models to learn the weed class well. Our final models were trained with sparse categorical cross-entropy loss.

8 Additional analysis

We spent some time writing additional code that allowed us to gain better insights on what the various models predict: we display a random image from the validation set, its mask and the prediction the model gave. We noticed that with our final U-Net models the predictions were even better than the masks as can be seen in the image:

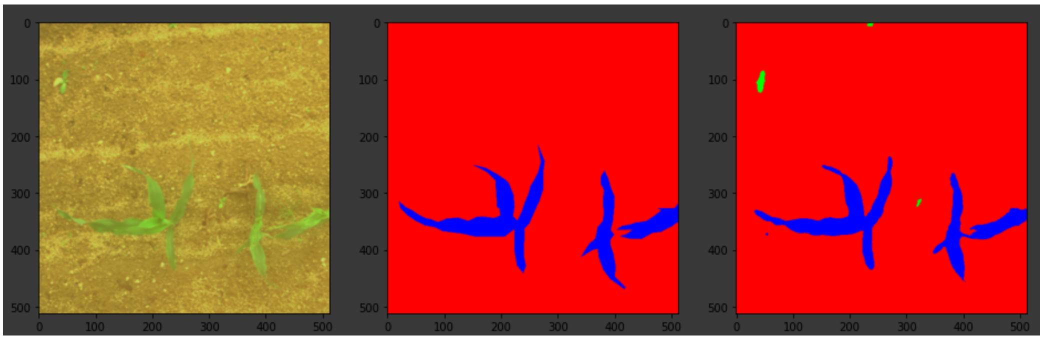

Figure 1: From left to right: image tile, image mask, model prediction

9 Conclusion

In the previous homework we learned the basic practical aspects related to deep learning, while in this homework we refined some concepts, learned how to manage cases where the model just doesn't seem to learn and finally we learned image segmentation in practice.