POLITECNICO DI MILANO

MSc in Computer Science and Engineering

Artificial Neural Networks and Deep Learning

A.Y. 2020/2021

CHALLENGE 2
IMAGE SEGMENTATION

Team: *Spatial Extent*

Nicola Rosetti 940435

Alessio Russo 945781

Vittorio Torri 945208

We started using an “handmade” encoder-decoder network and feeding the entire images to our network. As for the datasets, we started with only BipBip. Performance with encoder-decoder was really bad, since we had few images and a model to be trained from scratch, clearly this was not the solution.
So we tried with the Multiclass Segmentation Model seen in lab, the one using vgg16 as a backbone for feature extraction, fine-tuning the last layers, with a learning rate=10^-3.
This model was more promising, (0.62 val_meanIoU on BipBip). We also added the other datasets to the training, but in this way the problem of managing images with different dimensions arose: we decided to resize all the images to the dimension of Bipbip images.
Upsampling layers were substituted by transpose convolution ones.
The performance obtained was clearly worse, but not so bad, achieving 0.46 as global IoU on Codalab.
We tried then to drastically reduce the images dimensions to speed up training, but the performance was terrible.
To solve the unbalanced classes problem, we tried to use weights with the focal loss; performance didn’t improve. Even changing backbone (from vgg to MobileNet) resulted in no improvement (0.40 on test).
Then we added skip connections: the number of parameters was probably too high and performance worsened. All of this is in [Notebook 1].
Meanwhile we were also trying a kind of U-Net model with residual connections, trained from scratch, with ~ 3 millions parameters. We weren’t achieving good performances, feeding the entire image to the network and resizing every image to one fixed size, since the model had too many parameters and few images were available: overfitting was constant (max 0.25 val_meanIoU), even with different learning rates, the addition of dropout layers and of more convolutional layers to increase spatial extent.
We also tried to preprocess the images coloring in black everything that was not green, hoping to simplify the learning and in particular the boundaries of crops/weeds, but it didn’t improve the results, they got slightly worse, probably because in certain images the background contains portions of green which belongs to the background class. We also tried to add a 4th additional channel in input, with the “excessive-green” representation, but it wasn’t helpful.
At this point we decided to crop images in tiles.
The first attempt consisted in splitting every image in 2 big tiles except for Weedelec images that were split in 4 because of their high dimension; all tiles were resized to the same dimension. The same was done on the test set. This performed 0.55 in val_meanIoU, with a global IoU on CodaLab of 0.56. This tiling preprocessing was tried also on the U-net model, with almost no improvement as a result.
Then we tried smaller tiles (224x224) with an overlapping of 24px. Here two different approaches were used for training:

- resizing the images in order to be correctly cropped
- adapting cropping to the size of the image

Using the first approach, at testing time the images were resized a little, divided into tiles of 224x224 and then combined and resized to match initial image size. It was successful with the vgg model, reaching a 0.52 on CodaLab, while the second one didn’t achieve good performance.
We tried the first approach on the Unet model and good performances were achieved on validation (0.6) but on test was really bad (0.2 on CodaLab). The problem was (even in the vgg model, but there it was less trivial to see) the fact that we splitted crops of the same

image in validation and training. This resulted in overfitting the validation set. We then removed the splitting of the crops of the same image in different sets, but performances didn't improve so much.
The second idea was to keep the size of the original images and crop them with an overlap of 20 pixels. To cope with different dimensions, the last tiles have a different overlap. Using the new generated training set, bs=32, an initial learning rate of 10^{-3} and a validation split of 0.1, the U-net model has been trained for 12 epochs progressively lowering the learning rate to reach 10^{-5}. These are the final values:

```
accuracy: 0.9835 - meanIoU: 0.7436 - val_loss: 0.0673 - val_accuracy: 0.9771 - val_meanIoU: 0.6260
```

On the test set the tiling is the same, ties are broken choosing the background class or, in case, the crop class.
This finally achieved 0.68 on Codalab.
We also tried the classical U-Net architecture and we arrived near but not over the previous one. [Notebook 2] contains the final version of all this.