Artificial Neural Network and Deep Learning

Homework 2 (A.A. 2020/2021) – Image Segmentation

Diego Savoia, Matr. 944508 (10535515) – Francesco Emanuele Stradi, Matr. 944616 (10538326)

For this Image Segmentation problem, we decided to set up the architecture seen during the practical session of the course; this includes a batch size equal to 32, the image rescaling to 256x256 and the Adam optimizer with a learning rate of 1e-3. For simplicity, we started by considering only the images/masks from the Bipbip team and using data augmentation; then, we decided to create two CustomDataset objects (one for training and one for validation) which take the images from a single folder containing all the samples from all the different teams; the masks are taken from another directory, and they are converted into integer masks using the given script → Validation_MeanIoU ≈ 0.27.

Our work focuses on three different models:

- F-CNN (Model 1)
- U-NET (Model 2)
- VGG-16 for Segmentation (Model 3) – Final Model

F-CNN

The model structure is composed by an encoder part, which contains sequences of convolution, ReLU and MaxPooling, doubling the number of filters after each sequence. Then, after the bottleneck composed by a convolution and a ReLU, there is the decoder part, which contains sequences of upsampling, convolution and ReLU, halving the number of filters after each sequence. Finally, the prediction is given by the usual convolution layer with filter size 1x1 and a SoftMax.

We started by setting the image size to 512x512 and by tuning the batch size to 4, which showed an improvement → *CodaLab Score = 0.016*.

At this point we realized that during the prediction on the Test_Dev, every output was resized to the original size of the Bipbip images; instead, upscaling the results to the true original dimensions of the images (according to the team), we achieved the correct performance also on CodaLab → *Validation_MeanIoU = 0.36*.

U-NET

The model structure is composed by an encoder part, which contains a sequence of convolution and MaxPooling, followed by a decoder part, which upsamples the image using skip-connections.

We started with an image size of 256x256 and a batch size of 2 → *CodaLab Score = 0.053*. To improve the result, we tried to apply the VGG pre-processing to the inputs → *CodaLab Score = 0.062;* → *BipBip_Global_Score = 0.40*.

Changing the batch size to 4 and using an image size of 384x384 the results were slightly better → *CodaLab Score = 0.068;* → *Bipbip_Global_Score = 0.45*.

At this point we realized that during the prediction on the Test_Dev, every output was resized to the original size of the Bipbip images; instead, upscaling the results to the true original dimensions of the images (according to the team), we achieved the correct performance also on CodaLab → *CodaLab Score = 0.44 (with 512x512 image size);* → *Validation_MeanIoU = 0.44 (with 768x768 image size)*.

VGG-16 – Final Model

This is a Transfer Learning approach, importing VGG-16 from the Keras library without the top part, and applying a custom decoder part, which contains sequences of upsampling, convolution and ReLU, halving the number of filters after each sequence; finally, the prediction is given by the usual convolution layer with filter size 1x1 and a SoftMax.

Since this model is lighter than U-NET, we used an image size of 512x512 and we tuned the batch size to 4. Then, we started working on the fine-tuning value, obtaining best results freezing the first 13 layers → *CodaLab Score = 0.085*.

At this point we realized that during the prediction on the Test_Dev, every output was resized to the original size of the Bipbip images; instead, upscaling the results to the true original dimensions of the images (according to the team), we achieved the correct performance also on CodaLab → *CodaLab Score = 0.52*.

Training the network using an image size of 768x768 there was a visible improvement → *CodaLab Score = 0.59*. Better results were obtained using an image size of 1024x1024 → *Validation_MeanIoU = 0.61;* → *CodaLab Score = 0.64*. Finally, using an image size of 2048x2048 → *CodaLab Score = 0.67* (best score).