Multiclass Image Segmentation Project

Artificial Neural Network and Deep Learning

A Deep Learning model for multiclass crop segmentation

Carlo Ghiglione, Leonardo Perelli

12-12-2020

Contents

1 Introduction

We decided to develop our segmentation model only using the Bipbip dataset for haricot because we wanted to deal with data scarcity and see how this would affect our results.
We always used the *SparseCategoricalCrossEntropy* loss, the *Adam* optimizer, the $ReLU$ activation function and image augmentation.

2 Encoder-decoder model

2.1 Architecture

We started from a very simple architecture with an encoder-decoder structure made of five blocks. Each block contains:

1. a convolution and a max-pooling layer, doubling kernels number in the encoder;
2. an upsampling and a convolution layer, halving kernels number in the decoder.

The starting number of kernels is set at 16, the batch size at 4, the learning rate at 10^{-3}.
Rescaling the input image to size 256x256, we firstly obtained a validation mean IoU around 0.18 and some overfitting problems.

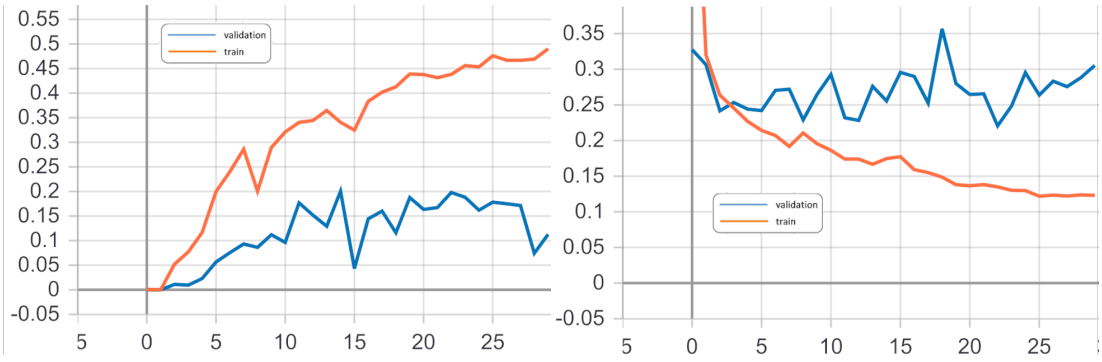

Figure 1: mean IoU (sx) and loss (dx) of the model

2.2 Increasing input size

In order to keep the level of details in input images as high as possible, we doubled their size up to 512x512. Keeping all the other hyper-parameters constant, the validation mean IoU highly increased up to 0.35, but overfitting problems still remained.

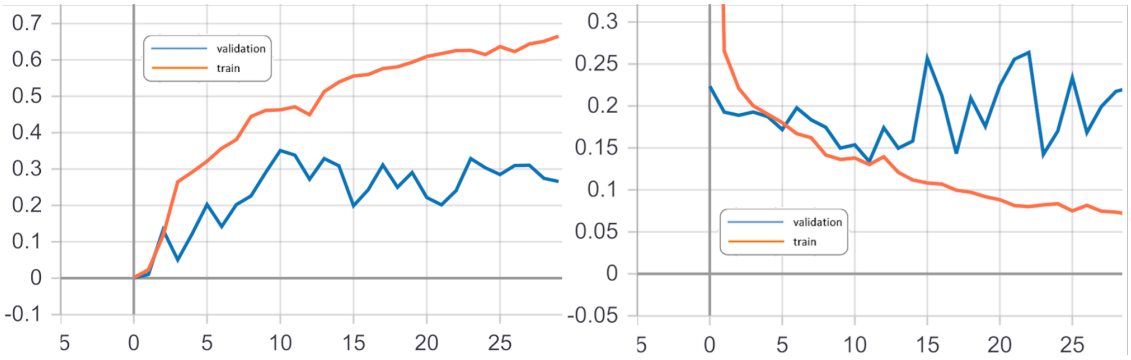

Figure 2: mean IoU (sx) and loss (dx) of the model

2.3 Adding regularizers and skip connections

To deal with overfitting problems, we introduced batch normalization between each convolution and relu activation function and, inspired by high performing architectures, we added skip connections for each block.
These features had a significant benefit: the overfitting problems are solved and the validation mean IoU arrived to 0.42.
The most evident problem is that for the majority of the training there isn't any improvement in the performances, probably because it is locked in a local minimum.

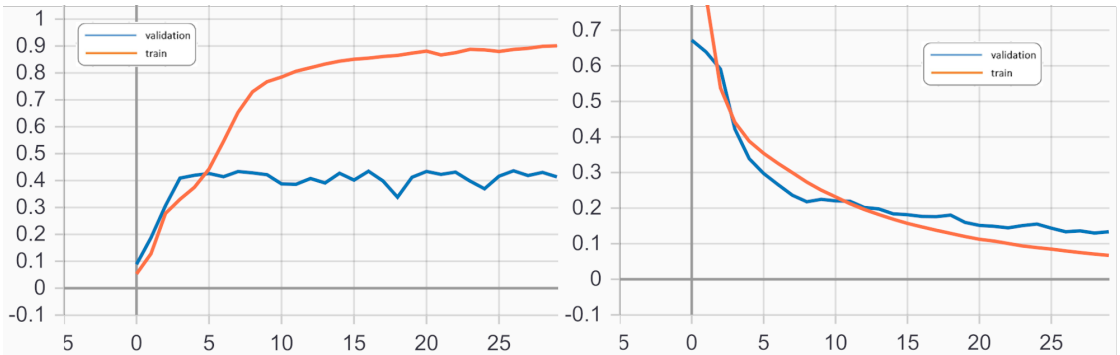

Figure 3: mean IoU (sx) and loss (dx) of the model

2.4 Implementing K-Fold cross-validation and adaptive learning rate

Due to data scarcity, we split the dataset in five folds and we trained the model on 80 percent of the data for 50 epochs, changing the validation fold every 10 epochs. Moreover, in order to avoid local minima, we increased the learning rate during the training every 10 epochs by 10^{-3}.
Considering the simplicity of the architechture, the results are relatively good, being the test mean IoU score 0.53 on average for the Bipbip haricot dataset and 0.32 for the global dataset.

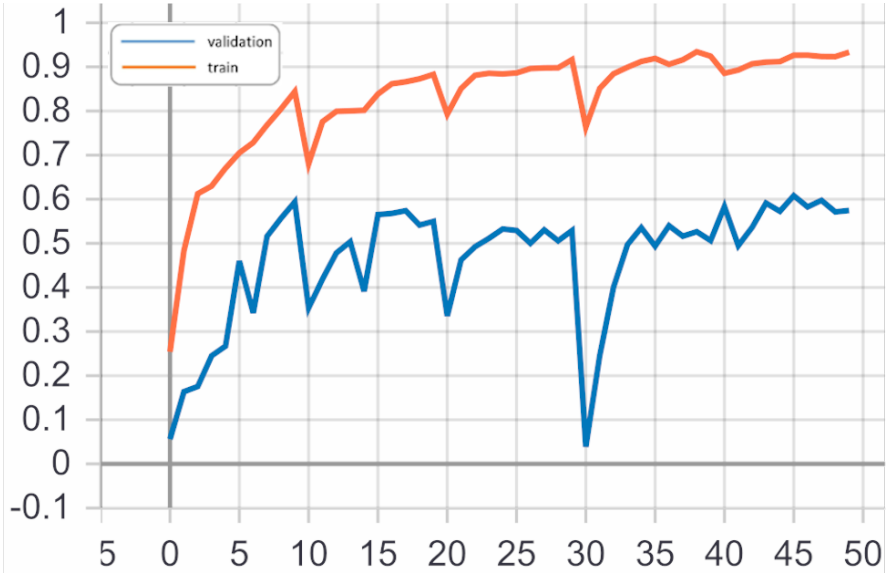

Figure 4: mean IoU (sx) and loss (dx) of the model

epoch	val_IoU	test_IoU	glob_IoU
46	0.6080	0.5502	0.3575
48	0.5975	0.5461	0.3317
44	0.5913	0.4882	0.3068
50	0.5746	0.5262	0.3640
41	0.5835	0.5488	0.2426
average	0.5910	0.5319	0.3205

3 VGG-16 as encoder

3.1 Model with VGG-16

Our next attempt was to use the VGG-16 as encoder. The decoder is made up by 5 blocks, with each one consisting of a up-sampling, a convolution, a batch-normalization and a *ReLU* layer.
The model was able to reach a peak validation mean IoU of 0.5, however the decoder lacks precision when reconstructing the full size images. There is overfitting quite early, maybe for the learning rate not being optimal.

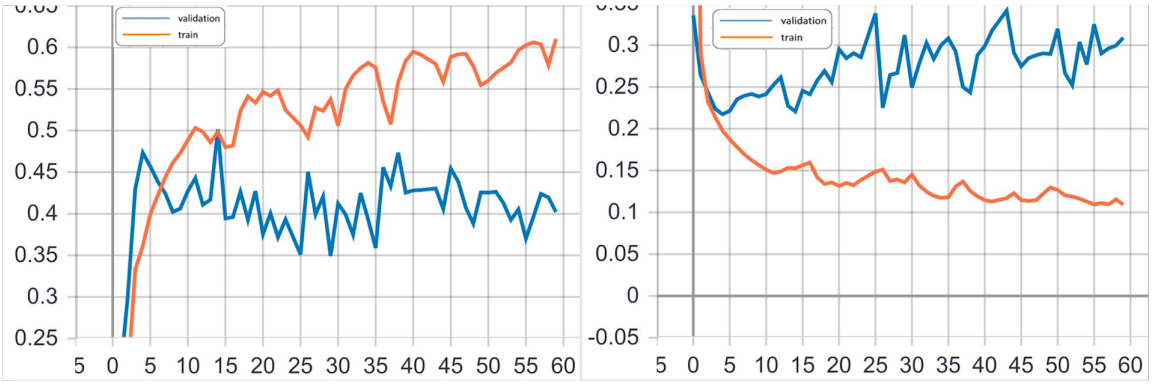

Figure 5: mean IoU (sx) and loss (dx) of the model

4 U-Net model

4.1 Architecture

In order to reach higher precision in the decoder, we added skip connections, taking inspiration by the U-Net model and building one from scratch.
It consists of a specular encoder - decoder structure with two types of blocks:

1. two convolutions followed by a max-pooling;
2. transpose convolution and concatentation with the corresponding output of the encoder side.

The starting number of filters was set at 12 and it was doubled in each down-sampling block and halved in the upsampling ones.

4.2 Training

We also developed a U-Net with batch-normalization layers, but the training was much worse in terms of stability, as we can see.
Trying to boost the performances, we adjusted the learning during the training by

increasing it when overfitting arised and reducing it when a plateau was reached. The effect of this can be seen for example around epoch 30.
The peak validation MeanIoU reached 0.70, which is relatively good considering the low complexity of 4 million parameters.

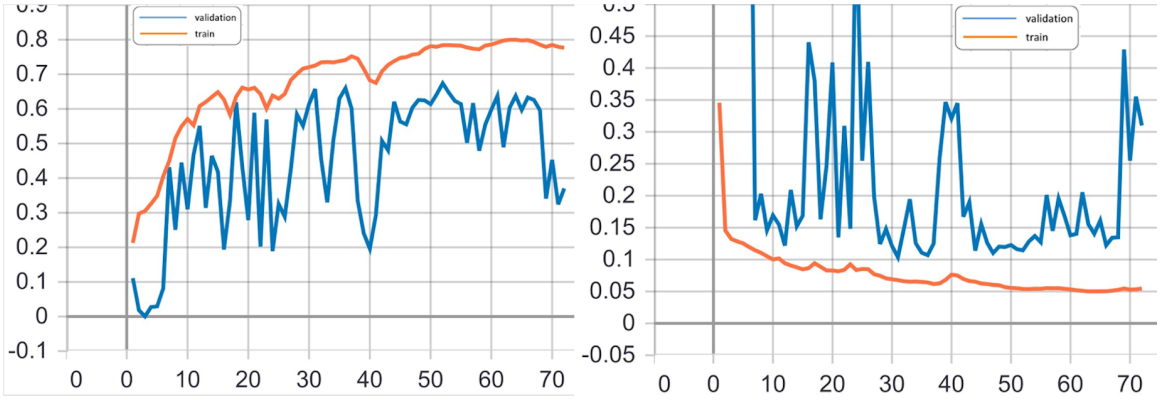

Figure 6: mean IoU (sx) and loss (dx) of the model with batch-normalization

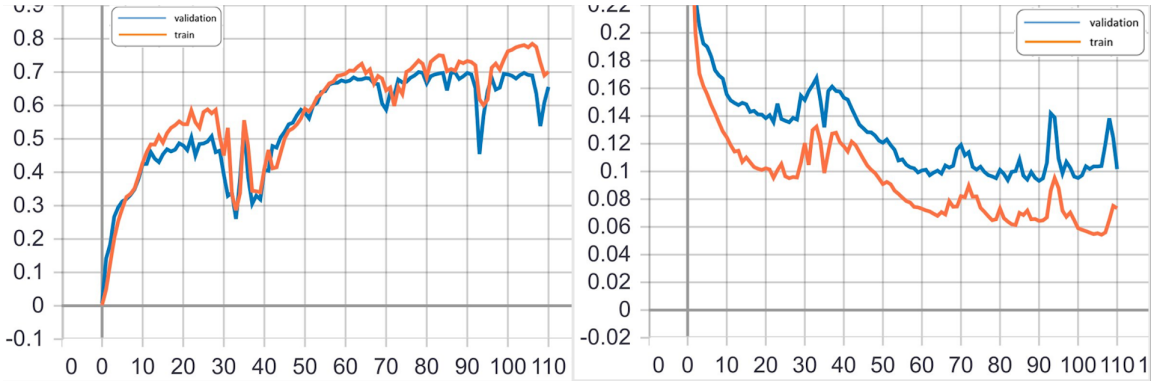

Figure 7: mean IoU (sx) and loss (dx) of the model without batch-normalization

4.3 Input size tuning

All our models were trained on 512x512 images, so we had to rescale the output masks to the original 1536x2048 dimension. However, it was also possible to directly feed the model with original size images, thanks to the characteristics of the U-Net layers. The best result was obtained in the first way, with a test mean IoU of 0.64 for the BipBip Haricot dataset and of 0.31 for the global one.

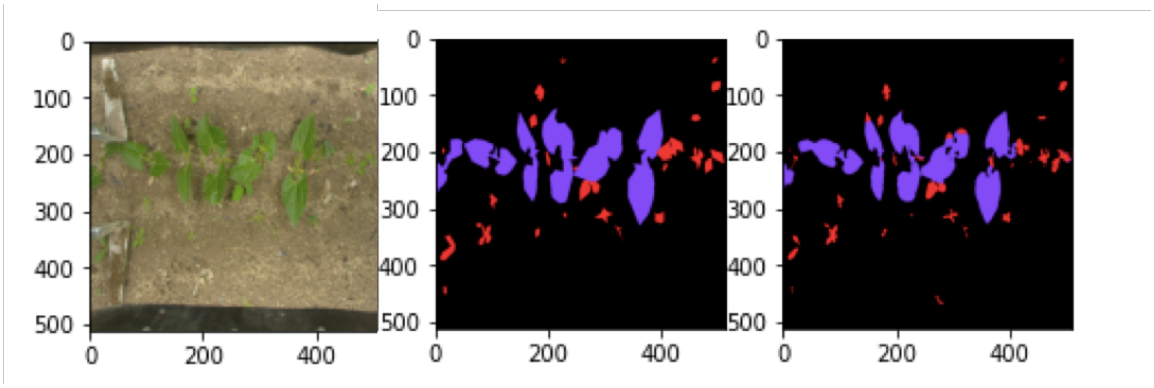

Figure 8: input image (sx), target mask (center) and predicted mask (dx)

5 Conclusions

Although we used basic architectures, implementing adaptive learning rate and skip connections we were able to reach high level of segmentation accuracy.
As further developments, we would have liked to:

1. tiling input images instead of rescaling them to maintain high precision;
2. developing a custom loss function to deal with class imbalance;
3. training the models with the other datasets.

We are satisfied because we learnt to deal with a real-life segmentation problem using deep learning techniques.