E4 project : Maximin Affinity Learning of Image Segmentation Appendix

Josselin Lefèvre, Maxime Burchi, Maxime Schoenberger

July 2020

Contents

1 Introduction

In this appendix you will find the continuation of the work carried out for the first report. The main task was to adapt the algorithms and the architecture of the network to process 3D images. This in order to know if training with volumes allows obtaining higher scores than those obtained with training according to two dimensions.

Here, we will detail our implementation of the 3D architecture used in [1], some experiments and the various observations that we were able to make during the training of the neural network.
Our work also focused on the post processing proposed by the paper that we had achieved only a naive implementation.
Training is always carried out with the Cremi dataset and it should be noted that we did not have much training time and the affinities are not as good as the previous report.

2 Neural Network Architectures

2.1 A Deeper U-Net Mala

The first network we implemented is a U-Net Mala with a deeper architecture using zero padding for every convolution operation. We tried this experience because a deeper architecture is often associated with deeper meaning representation during learning. A padding for every convolution allows us to stack more layers adding a 5th level with 1024 filters without diminishing the size of the output gradient predictions along height and width. As previous experiences, the network takes a 256×256 input image and output the gradients along x and y axes for the whole image. Convolutional layers with 3×3 kernel size and Rectified Linear Unit activations are also used and max pooling operations stay the same. Bilinear upsampling layers are replaced by transposed convolutions as described in [1].

The network is trained for 230.000 iterations using an Adam optimizer with an initial learning rate $\alpha = 10^{-4}$, $\beta_1 = 0.95$, $\beta_2 = 0.99$ et $\epsilon = 10^{-8}$. During inference, a simple pixel wise mean followed by a threshold is applied to the two gradient volumes to create the final segmentation.

With this network, we were able to generate thinner segmentations for every frame and get better results on the Cremi test-set than previous experiences.

2.2 Multi Input Images

This architecture came with the idea to use 3 successive frames as input and compute the loss only on the center image in order to predict a better segmentation along the depth. As we know, Cremi is a volume dataset so a network linking a frame to its neighbors should better predict segmentations along the depth than previous networks.
This network shares the same architecture as U-Net Mala described in previous works, optimizer and other hyperparameters for training and inference remain the same.

2.3 3D U-Net Mala

One of our main objective was to implement the network as described in [1]. This part describes the implementation of a 3 dimensional U-Net like architecture for volume segmentation. Mathematical operations and training parameters are the same as described in the paper. The network takes $84 \times 268 \times 268$ (depth, height, width) patches as input and predict the gradient along x, y and z axes defined by $56 \times 56 \times 56$ outputs.

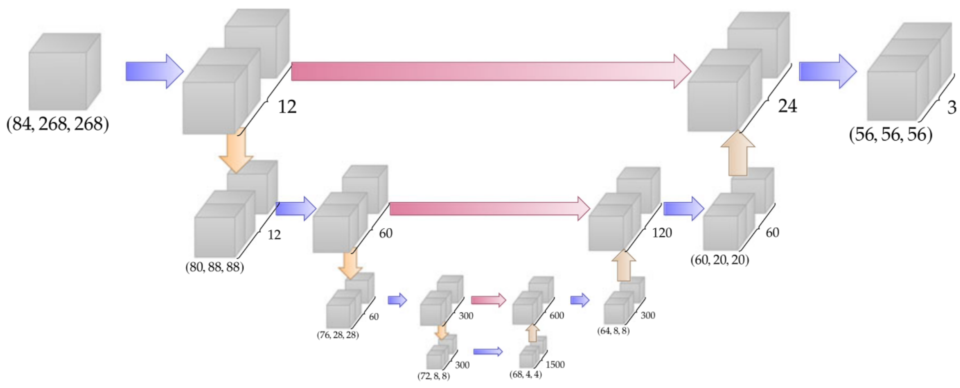


Figure 1: Overview of the U-Net architecture used for the Cremi dataset in [1]. The network is 4 levels deep and doesn't use padding for convolution operations. Transposed convolutions and max pooling layers use a stride of 3 along height and width but do not affect the input size along the depth. Relu activations are used after every convolution layer.

To implement the network, we created a 3D version of the processing algorithms for calculating edge weights and decompose the dataset into patches. For inference, every patch of the volume is fed into the network, then gradients volumes are reconstructed for the 3 axes.

3 Training observations

3.1 Collapsing

During the training we saw the network collapse several times. The gradients took on a uniform value and the network was unable to get out of this situation. After comparing the losses for each time this happened we were unable to find a pattern. The number of iterations was also quite different: 50K, 60K or 70K iterations.

We also checked the predictions before this collapse but we did not see anything that could have alerted us. The only common point is that each time the "grids" which we will talk about in the next part had already appeared.

3.2 Bad predictions along Z

As described in section 6.3 of the previous report, we encountered a problem of dissimilarity between the predictions in x and in y. We can note that this inconsistency in the predictions has completely disappeared. However, we were able to see a "grid" appear in the zones corresponding to the darkest zones of the input image such as the nuclei.

This phenomenon only occurred after a certain number of iterations and never disappeared.

We believe that this is due to the anisotropy of the volumes. In the Cremi dataset the resolution along the z axis is ten times greater than that in x and y. It would be interesting to interpolate the frames according in order to reduce this difference and observe if this phenomenon still occurs. Another simpler solution to verify the source of this problem would be to train the network on isotropic volumes like Fib-25.

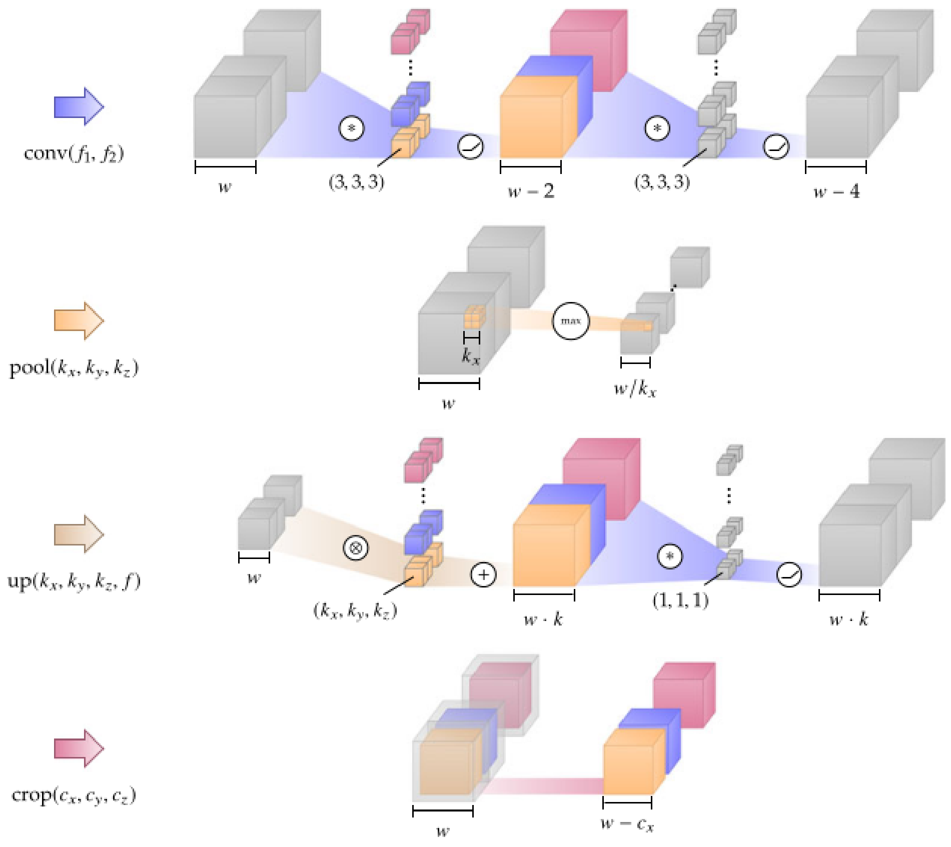
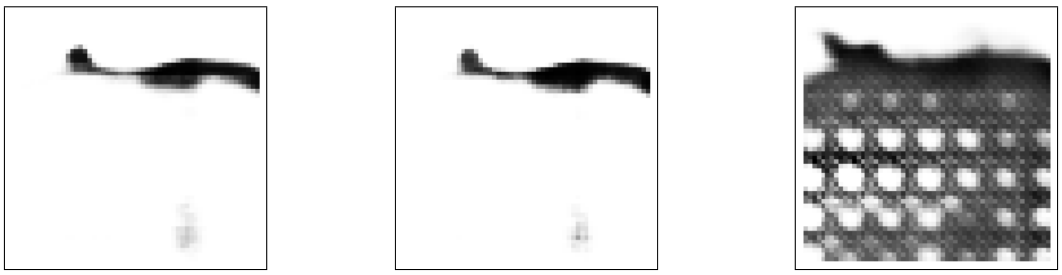


Figure 2: Details of convolution operations (blue), pooling (orange), transpose convolution (brown) and crops (red). Every transpose convolution layer is followed by a concatenation operation between outputs and crops from the same level. A $1 \times 1 \times 1$ convolution follows the concatenation to reduce the filter dimension.



(a) Affinity along the X axis (b) Affinity along the Y axis (c) Affinity along the Z axis

Figure 3: Bad affinity prediction along the Z axis

3.3 Deterioration of the agglomeration

We worked on the implementation of the agglomeration method proposed in [1] that the first group had carried out in order to optimize it and compare it with that provided by the authors of the paper (a library called waterz). By computing the region adjacency graph of the regions only once, and not at every step as previously, we have succeeded in reducing the execution time and we have obtained results quite comparable to those obtained with waterz. The differences can be explained by the discretization of the initial costs into k-bins which we have not implemented. But waterz is still much faster than our python implementation

As described earlier we saw a "grid" appear in areas corresponding to the the darkest regions of the input. We wondered if this could cause the agglomeration of regions along the z axis to deteriorate. We then implemented the agglomeration algorithm on different affinities. First those which do not present this grid and then those where it is present. To carry out our experiment, we preferred to take the ground truth as a fragment, taking care to decouple the x-y frames along the z axis by adding to each value of an x-y frame its index.

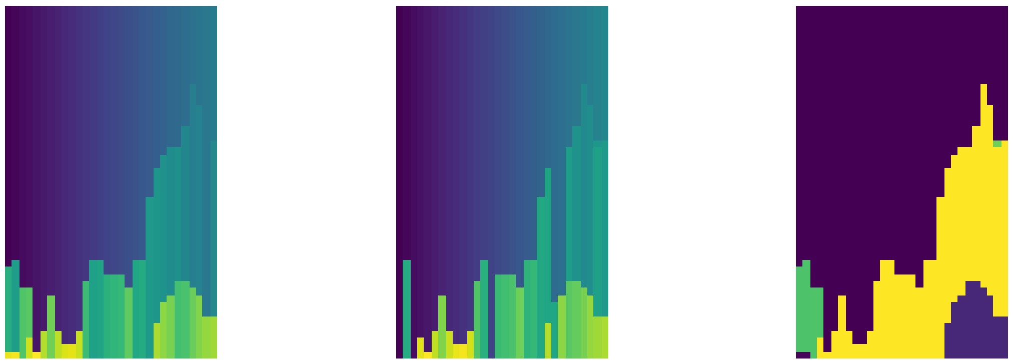


Figure 4: On the left the x-z cut from the result of agglomeration on affinities without the grid, at the center x-z cut from the result of agglomeration on affinities with the grid and on the right the ground truth.

After taking several x-z cuts we could see that the agglomeration according to z suffered from the presence of this grid as we can see in figure 4.

4 Results

As for the previous report, we focused our evaluation on the Cremi dataset. We didn't have time to implement data augmentation like 2D training.
We have selected a good iteration for the Unet 3D but the number of iterations is well below that of the other networks. Indeed, the training time exploded with the use of volumes and we could only train briefly. The time losses are not due to the calculation of the loss which remains fast but to the data transfers between the GPU, for the neural network and the CPU for the calculation of the loss with Higra.

In the following tables "U-net MALA" refers to the network implemented by the previous group and "U-net MALA 3 images" refers to the network taking as input 3 successive frames.

	Rand index	VOI merge (lower is better)	VOI split (lower is better)	CREMI score (lower is better)
U-net MALA	0.83	0.46	0.54	0.42
U-net MALA padding	**0.84**	**0.39**	**0.48**	**0.36**
U-net MALA 3 images	**0.84**	0.40	0.52	0.37
U-net MALA 3D	0.80	0.69	0.47	0.48

Table 1: Results on the training set of the CREMI dataset

	Rand index	VOI merge (lower is better)	VOI split (lower is better)	CREMI score (lower is better)
U-net MALA	0.80	0.50	0.57	0.46
U-net MALA padding	**0.82**	**0.43**	**0.51**	**0.40**
U-net MALA 3 images	0.80	0.45	0.56	0.44
U-net MALA 3D	0.75	0.53	0.77	0.56
SOTA (in 3D)	*0.89*	*0.115*	*0.339*	*0.221*

Table 2: Results on the test set of the CREMI dataset

We can see that the experience with padding is conclusive and provides better results than previously as much on the training set as on the train set. The most interesting improvement is at the VOI level. By calculating the loss on a larger patch we have significantly improved in performance. However, we are still far from the state of the art regarding the VOI split and merge.

The results of the 3D U-net are not yet up to par but given the little iteration this is quite encouraging. The differences between the numerical results of the train set and the test set clearly show that the network is not able to generalise well and lacks training. It will be necessary to optimize the code for the transition from GPU to CPU to save time and of course work on the inconveniences which we have discussed in the previous parts.

5 Teamwork

5.1 Team organization

The context of this project is somewhat special because with quarantine we have never seen each other face to face. Knowing that Maxime B. and Maxime S. had no knowledge of the project, we had to make many calls so that Josselin explained the content of the papers as well as the work carried out. The recording of the presentation as well as the previous report were good supports to facilitate the handling of the project.

The organization is the same as that of the previous group: we have set personal goals listed in a document, and we have a meeting per week with our supervisor.

We finally had only a short time to carry out our objectives because they took time for new entrants to be comfortable with the project and of course the training time is very long with volumes.

5.2 Task distribution

Josselin	Maxime B.	Maxime S.
Graph generation from output of neural network	Learn PyTorch	Learn PyTorch
Training	Implement 3D Unet	
Review of the previous implementation of post-processing and comparison with that of the authors	Training and Evaluation	
	Image reconstruction with inference	

Table 3: List of the main tasks and their repartition

6 Conclusion

As we have seen, we have been able to obtain promising results by working on the architecture of the neuron network. However the results of our implementation of the 3D architecture is quite disappointing due to the problems we encountered and the short training time that we were able to do.

There is still a lot of experiments to do as much on training as on post-processing. The question of the importance of the agglomeration of regions in terms of numerical results remains pending. Moreover the previous group was able to apply MALIS to another type of image with BSDS500 using the implementation of [2] but we did not have time to do it with the improved method.

We would like to thank Laurent Najman for supervising us during this project.

References

[1] J. Funke, F. Tschopp, W. Grisaitis, A. Sheridan, C. Singh, S. Saalfeld, and S. C. Turaga, "Large Scale Image Segmentation with Structured Loss Based Deep Learning for Connectome Reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 1669–1680, July 2019.

[2] S. C. Turaga, K. L. Briggman, M. Helmstaedter, W. Denk, and H. S. Seung, "Maximin affinity learning of image segmentation," *arXiv:0911.5372 [cs]*, Nov. 2009. arXiv: 0911.5372.