SVETLANA: A SUPERVISED SEGMENTATION CLASSIFIER FOR NAPARI

Clément Cazorla, Renaud Morin *

IMACTIV-3D
1 place Pierre Potier, 31100 - Toulouse

Pierre Weiss †

CNRS & Université de Toulouse
31400 - Toulouse

ABSTRACT

We develop and present a Napari plugin called SVETLANA (SuperVised sEgmenTation cLAssifier for NapAri). It is dedicated to the manual or automatic classification of segmentation results, with a primary focus on bio-medical imaging. While many open-source software now make it possible to automatically segment complex 2D and 3D objects such as cells in biology, the subsequent analysis of the results is not yet accessible to non specialists. This plugin allows end-users to train and run efficient neural network classifiers such as residual networks. The resulting network can be used as a post-processing tool to improve the segmentation, or as a classifier for various tasks (e.g. separating different cells populations). We showcase its practicality through various real cell biology problems in 2D, 3D and multi-spectral imaging.

***Index Terms*—** Software, Segmentation, Classification, Convolutional Neural Networks, Bio-medical imaging, Image analysis

1. INTRODUCTION

The last decade has made automatic segmentation of bio-medical images much more accessible to users not familiar with signal processing. This is the result of progress in machine learning, to the creation of open training databases and to the development of ergonomic open-source software. Technologies such as neural networks provide unprecedented segmentation results. They make it possible to avoid setting hyperparameters which are often hard to tune and interpret. Examples of powerful and popular tools for segmentation in biology include Ilastik [1], CellPose [2], StarDist [3] or Deep-ImageJ [4].

1.1. Our motivation

Unfortunately, segmentation masks – as good as they are – are rarely directly exploitable to answer biological questions. In particular, it is often necessary to classify the detected objects in order to perform statistical analyses that give a concrete meaning to the results. While these excellent segmentation tools have solved an important problem, a difficult part of the analysis remains inaccessible to most users.

1.2. Our contribution

The goal of this work is to continue filling the gap between methodological advances and end-users, by providing a convenient software for the classification of segmentation results. We resort to the newborn Napari environment [5] which allows visualizing and analyzing complex multi-dimensional images (e.g. 2D, 3D, 3D+t, hyperspectral) in Python. Our software takes the form of a Napari plugin called SVETLANA. A screenshot of the plugin is displayed in Fig. 1.

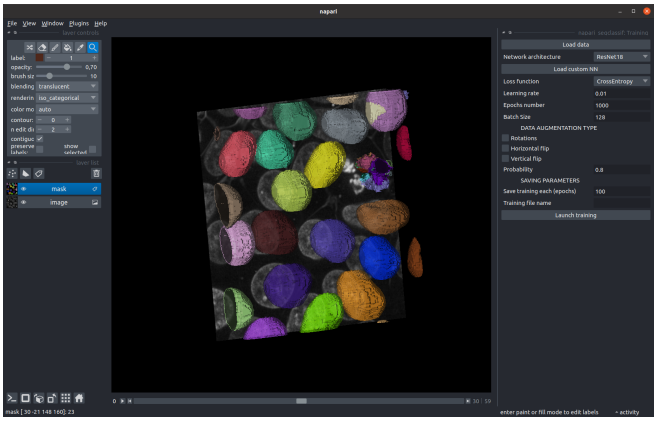


Fig. 1: The SVETLANA plugin under NAPARI.

It takes *an image to label and a segmentation mask* as an input. It is then separated in three different modules depicted in Fig. 2:

Annotation This module allows to label connected components of the segmentation masks. In our 2D experiments, it allows labeling about 1000 connected component in less than 30 minutes.

Training This module allows to pick an arbitrary Pytorch [6] neural network architecture (possibly pretrained) and to further train it with the annotations generated by the previous module.

*C. Cazorla is partially funded by ANR CIFRE 2020/0843. He acknowledges the image.sc community for their precious help.

†P. Weiss acknowledges a support from ANR-3IA Artificial and Natural Intelligence Toulouse Institute and ANR Micro-Blind

Prediction This module uses the trained network to classify the connected components of the segmentation mask.

The outputs of the plugin are: *a set of manually annotated patches, a trained neural network and a prediction mask*. The results are stored in files widely accessible formats for the forth-coming analyzes. This plugin meets a need to further enhance the excellent results obtained with recent, wide purpose segmentation tools.

1.3. Related works

Different options can be adopted to segment and classify objects in images. Before 2010, most of the works relied on the following pipeline (see e.g. [7]): 1) Segment the image 2) Annotate the resulting masks 3) Extract features within the masks (e.g. edges, textures, ...) 4) Design a classifier based on the extracted features. Each of the above step was carried out with carefully hand-crafted methods. Supervised and unsupervised learning then progressively entered in the game. In many cases, they outperformed man-made routines by allowing to explore a wider range of decision routes.

An effort was then pursued to make these technologies available to the larger number. One remarkable example is Ilastik [1]. There, a few annotations by the user are usually enough to perform complex classification tasks with an arbitrary number of classes. Its backbone is a random forest classifier with a fixed number of features (convolutions with different filter types). It is widely praised for its ease of use. A few clicks are enough to solve many real-world problems. Unfortunately, this strength is also a limitation in certain cases: the performance of random forests falls short in comparison with the most advanced segmentation and classification routines trained with vast collection of carefully labeled data.

When precision is critical, the rapidly evolving state-of-the-art is rather based on neural networks and especially convolutional neural networks [8, 9]. The downside of these technologies is the need to create large data sets, which are usually just not accessible. Each biology laboratory explores a different organism, at a different scale with a different modality and focus. Each collected image can be costly both in terms of money, know-how and time. To address this issue, new initiatives emerge to collect large heterogeneous training databases. For instance the Data Science Bowl [10] allowed to train a single neural network, which is now capable of segmenting cells of nearly any type. This tool, embedded in neat graphical interfaces (e.g. CellPose [2] or StarDist [3]) is a huge asset for biology. Unfortunately, as of now, it does not provide classification tools.

The proposed ideas can be seen as a two-step procedure to classify complex objects. First, we use a general purpose segmentation routine trained with a huge dataset. This highly simplifies the classification task by having to focus only on the right objects. Then train a classifier with just a few annotations (e.g. 100-1000) to sort the resulting objects. As far as we are aware of, this principle is not yet made available as an open-source software.

2. PLUGIN DESCRIPTION

The objective of SVETLANA as a whole is to provide a user-friendly tool targeted to people not familiar with programming to label segmentation results, either manually or automatically.

2.1. The choice of Napari

The specifications were:

- Use highly parallel architectures for run-time efficiency.
- Easyness to integrate new classifiers.
- Embbed in an environment providing efficient segmentation tools.
- Embbed in an environment allowing to visualize the results efficiently.
- Easyness to use other analysis tools.

Those considerations led us to choose the Napari environment. It is developed in Python and already includes state-of-the-art segmentation tools such as Cellpose [2]. In addition, the development is currently very fast with new plugins being issued on a weekly basis. It allows to interact with Pytorch or TensorFlow in a natural way and therefore use the latest developments in machine learning.

2.2. The plugin

A general overview of the plugin is provided in Fig. 2.

2.2.1. The annotation mode

This first sub-plugin takes two images as an input: the image itself and its segmentation mask. The user can also choose the number of different labels he wants.

The connected components of the segmentation mask are then extracted using scikit-image [11]. Then, there are two possible ways to annotate:

- The user clicks on the desired connected component and labels it.
- SVETLANA randomly picks a connected component, displays it with its neighborhood and the user labels it. This mode can help avoiding a user bias in the choice of the connected components.

Once the user feels enough annotations have been proposed, he can save the results in a binary file.

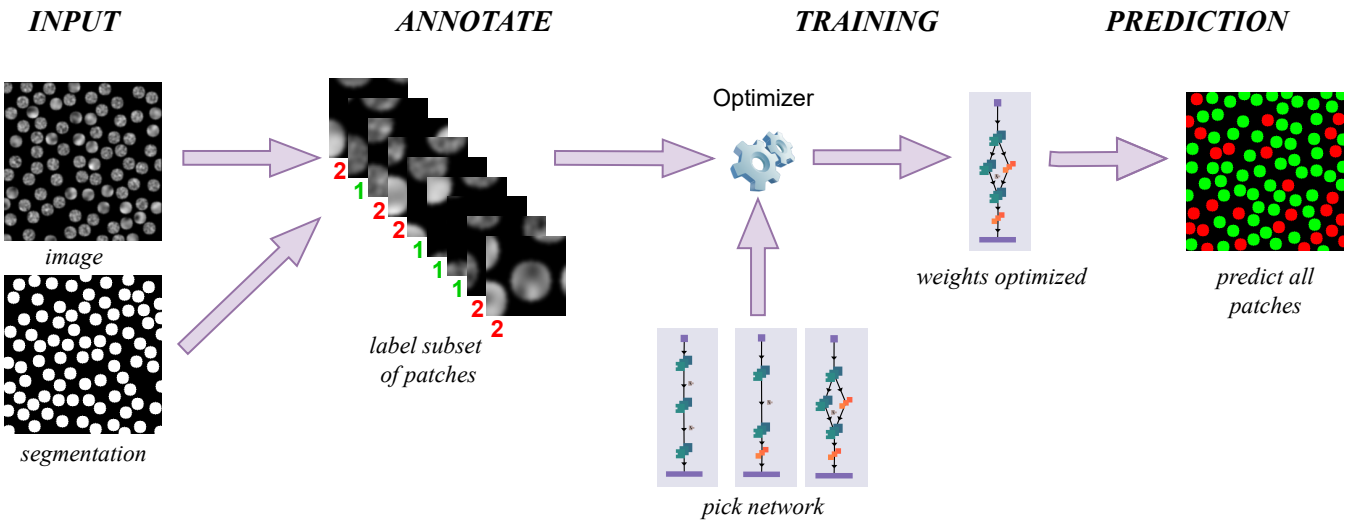


Fig. 2: The SVETLANA plugin pipeline.

2.2.2. *The training mode*

Interface In this mode, the user can:

- choose a loss function (cross-entropy, binary cross-entropy, MSE,...)
- pick a neural network from a list of pre-defined architectures (e.g. ResNet with various depths [12]) or his own pre-trained network in a .pth file.
- adapt the parameters of the Adam optimization routine.
- choose the batch-size for training, depending on the memory resource available.
- choose the type of data augmentation he wants (none, flip, rotation, . . .)
- train the network with the previous parameters.

Why could it work? Training a classifier with just a few annotations (10-1000) goes against conventional wisdom. Indeed, complex neural network architectures as ResNet are usually trained with huge dataset. It is therefore legitimate to question why the training could yield a good classifier. We do not have a theoretical answer for this beyond numerical experiments. It turns out that this approach provides fairly good classifiers (see section 3).

Let us mention that a few recent works point out that this is a rich research avenue. An example is Deep Image Prior [13]. There, it was shown that a large collection of linear inverse problems could be solved by using a single image as an input without any training. One way to interpret this is that the training phase could act as Occam razor's principle and favor the 'simplest' answer given the observations. The network architecture plays a critical there and we observed experimentally that simple networks with few weights were often preferable to more complicated ones. In all forthcoming experiments, we used the simple 2-layers neural net depicted in Fig. 3. It contains 3017 parameters.

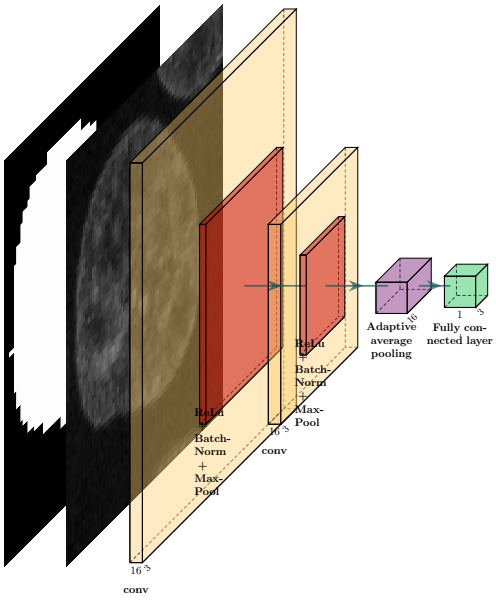


Fig. 3: A minimalist 2-layers convolutional neural network.

2.2.3. *The prediction mode*

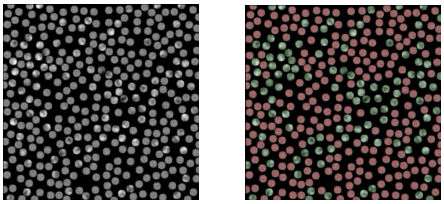
This plugin allows to load the trained network and to launch the prediction by choosing the batch size. It then saves the results in a binary file.

3. NUMERICAL EXPERIMENTS

In all experiments, we use the minimalist CNN from Fig. 3. For the data augmentation, we simply flipped the patches in the 4 arbitrary positions. All calculations were performed on an GPU Nvidia RTX5000 with 16Go memory.

3.1. Experiment 1: artificial texture classification

For this first experiment, we generated an image composed of 345 circular objects with two different textures (see Fig. 4). For the training, we label 100 objects in about 3 minutes. After a training 500 epochs (about 20 seconds), we get a misclassification rate of 0.8%.

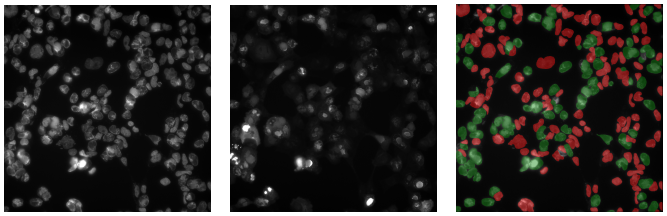


(a) Original image (b) Classification

Fig. 4: A first result on texture classification.

3.2. Experiment 2: Infected cells detection in virology

The two images on the left of Fig. 5 depict two different channels of human embryonic kidney (HEK) cells. Some of them are infected with an adenovirus. The first channel (left) displays a Hoechst staining of the nuclei. In the second channel, the sites of viral proliferation have been marked using the ANCHOR system [14]. The cells containing bright nuclear spots in the second channel are more likely to be infected. We annotated 100 cells over 245 (2 minutes) to train the 2-layer model and got a rate of 95.5% of well-classified cells.

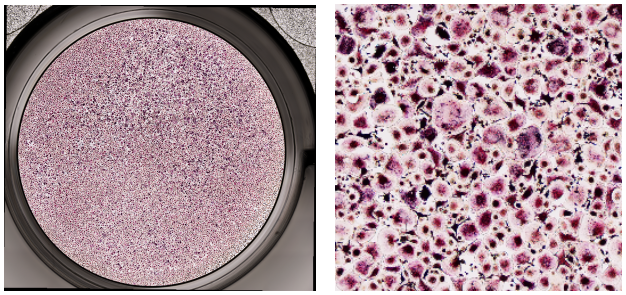


(a) Hoechst staining (b) ANCHOR channel (c) classification result

Fig. 5: Result of the infected cells classification.

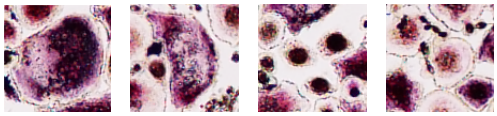
3.3. Third experiment: ABS

4591 kept over 16671 after classification.



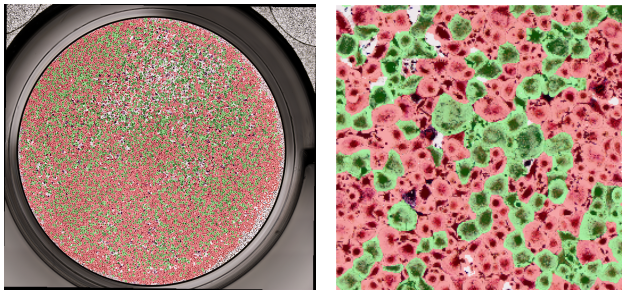
(a) ABS image (b) Zoomed crop in the image showing the cells variety

Fig. 6: ABS image to segment and classify



(a) Good cell (b) Good cell (c) Bad cell (d) Bad cell

Fig. 7: ABS image to segment and classify



(a) Overview of the classification result (b) Zoomed crop of the classification result

Fig. 8: Classification result

4. CONCLUSION

We showcased through various applications that SVETLANA is already a handful tool for analyzing segmentation results. It is yet at an early stage of development. By publishing this paper, we wished to have a first document that could be cited in the case that the plugin was used successfully. We plan to progressively add new features suggested by users or our own experience, to make this tools as helpful as possible to the community.

- Add the ability to label multiple images.
- Add the ability to directly label or correcting the labels by clicking on the connected components.
- Add the ability to predict on multiple images.

- Add a layer with already annotated features.

5. REFERENCES

[1] Stuart Berg, Dominik Kutra, Thorben Kroeger, Christoph N Straehle, Bernhard X Kausler, Carsten Haubold, Martin Schiegg, Janez Ales, Thorsten Beier, Markus Rudy, et al., "Ilastik: interactive machine learning for (bio) image analysis," *Nature Methods*, vol. 16, no. 12, pp. 1226–1232, 2019.

[2] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *Nature methods*, vol. 18, no. 1, pp. 100–106, 2021.

[3] Elnaz Fazeli, Nathan H Roy, Gautier Follain, Romain F Laine, Lucas von Chamier, Pekka E Hänninen, John E Eriksson, Jean-Yves Tinevez, and Guillaume Jacquemet, "Automated cell tracking using stardist and trackmate," *F1000Research*, vol. 9, 2020.

[4] Estibaliz Gómez-de Mariscal, Carlos García-López-de Haro, Wei Ouyang, Laurène Donati, Emma Lundberg, Michael Unser, Arrate Muñoz-Barrutia, and Daniel Sage, "Deepimagej: A user-friendly environment to run deep learning models in imagej," *Nature Methods*, vol. 18, no. 10, pp. 1192–1195, 2021.

[5] Jeffrey M Perkel et al., "Python power-up: new image tool visualizes complex data," *Nature*, vol. 600, no. 7888, pp. 347–348, 2021.

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[7] Humayun Irshad, Antoine Veillard, Ludovic Roux, and Daniel Racoceanu, "Methods for nuclei detection, segmentation, and classification in digital histopathology: a review—current status and future potential," *IEEE reviews in biomedical engineering*, vol. 7, pp. 97–114, 2013.

[8] Anamika Dhillon and Gyanendra K Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[10] Juan C Caicedo, Allen Goodman, Kyle W Karhohs, Beth A Cimini, Jeanelle Ackerman, Marzieh Haghighi,

CherKeng Heng, Tim Becker, Minh Doan, Claire McQuin, et al., "Nucleus segmentation across imaging experiments: the 2018 data science bowl," *Nature methods*, vol. 16, no. 12, pp. 1247–1253, 2019.

[11] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, pp. e453, 2014.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[13] Victor Lempitsky, Andrea Vedaldi, and Dmitry Ulyanov, "Deep image prior," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 9446–9454.

[14] Bernard Mariamé, Sandrine Kappler-Gratias, Martin Kappler, Stéphanie Balor, Franck Gallardo, and Kerstin Bystricky, "Real-time visualization and quantification of human cytomegalovirus replication in living cells using the anchor dna labeling technology," *Journal of Virology*, vol. 92, no. 18, pp. e00571–18, 2018.