Svetlana: a Supervised Segmentation Classifier for Napari

Clément Cazorla, Renaud Morin, Pierre Weiss

Svetlana: a Supervised Segmentation Classifier for Napari

Clément Cazorla[1, 2], Renaud Morin[2], and Pierre Weiss[1]

[1]Institut de Mathématiques de Toulouse (UMR 5219). University of Toulouse & CNRS. UPS, F-31062 Toulouse Cedex 09, France
[2]Imactiv-3D. Centre Pierre Potier, 1 place Pierre Potier, 31100 Toulouse, France

Abstract

We present Svetlana (SuperVised sEgmenTation cLAssifier for NapAri), an open-source Napari plugin dedicated to the manual or automatic classification of segmentation results. A few recent software have made it possible to automatically segment complex 2D and 3D objects such as cells in biology with unrivaled performance. However, the subsequent analysis of the results is oftentimes inaccessible to non-specialists. The Svetlana plugin aims at going one step further, by allowing end-users to label the segmented objects and to pick, train and run arbitrary neural network classifiers. The resulting network can then be used for the quantitative analysis of biophysical phenoma. We showcase its performance through challenging problems in 2D and 3D. Comparisons with random forest classifiers, which are the only easily available alternative to date, show significant advantages for the proposed approach.

Keywords– Software, Segmentation, Classification, Convolutional Neural Networks, Biomedical imaging, Image analysis, Microscopy, Frugal AI

1 Introduction

Recent years have witnessed spectacular progress in biological imaging. We can think of the improvement and accessibility of high-resolution microscopes, the explosion in storage and computing resources, and the advances in artificial intelligence. This offers exciting prospects for better understanding life. These advances however hinge on the ability to automatically analyze large volumes of data and, in particular, to segment and classify biological structures.

Specialized tools have emerged (e.g. the HoVer-Net [11]) and yield excellent performance for the quantification of histopathology images stained with a specific compound. Unfortunately, even though it effectively addresses an important and difficult issue, its adaptation to different datasets (imaging modality, staining, type of tissues, type of classification) is far from being obvious. To the best of our knowledge, there currently does not exist a general purpose classification software.

The situation is quite different for segmentation tasks. This is the result of concomitant facts including advances in machine learning, the creation of open training databases and 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 [4], CellPose [22], StarDist [9], ZeroCostDL4Miic [23] or Deep-ImageJ [10]. Their performance and ergonomics continue to improve at a fast pace.

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. Despite the significant benefits of these segmentation tools, a difficult part of the analysis therefore remains inaccessible to most users.

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 with a minimum amount of manual annotation. We designed a user-friendly plugin called Svetlana (see Fig. 1) within the newborn Napari environment [18].

2 Results

2.1 Workflow of Svetlana

The principle of Svetlana is displayed in Fig. 2. A tutorial video is available. It relies on an external segmentation module which outputs a segmentation mask. Svetlana then takes *the images to be labeled and the segmentation masks* as inputs (see Fig. 2 a)). It is then separated in three different modules depicted in Fig. 2 b):

Annotation This module allows the user to label some connected components of the segmentation masks. For simple classification tasks in 2D or 3D, we could label more than 1000 connected components in about 15 minutes.

Training This module allows the user to pick an arbitrary PyTorch [17] neural network architecture (possibly pre-trained) and to further train it with the annotations generated by the previous module.

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

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

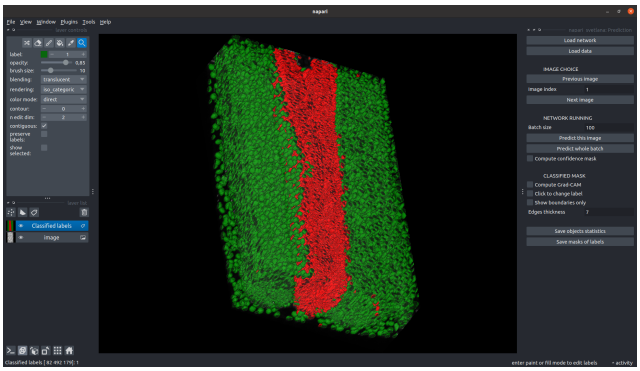

Figure 1: **The Svetlana plugin under Napari.** The image shows nuclei of a quail embryo classified by Svetlana (see Section 2.5).

2.2 Ilastik VS Svetlana

To the best of our knowledge, the software offering the closest functionality to Svetlana is Ilastik [4], and especially its *object classification* module. It is an excellent and powerful tool, which is widely adopted in biology laboratories. In this section, we discuss the differences, strengths and limitations of both applications.

The classifier Despite obvious similarities, the backbones of the two software are fundamentally different: Ilastik's classifier is currently based on random forests [6] while Svetlana's relies on neural networks.

Random forests require a *pre-defined set of features*, such as geometrical properties of the segmentation mask (volume, diameter, location, ...), or intensity properties inside and outside the segmented zone (mean intensity, variance, quantile, ...). These features are then assembled to construct a set of random decision trees. This process can be achieved through many different randomized techniques [5]. The final decision taken by the random forest classifier is based on a majority vote using the output of each tree, see Fig. 3 a).

Deep neural networks and especially the convolutional neural networks used in this paper work very differently. One of their main difference is that they are able to infer *automatically* the features, instead of having a fixed set of pre-defined image characteristics. The features (here, convolution filters and biases) are learned automatically from the labeled data using stochastic gradient descents.

Both approaches have their pros and cons. Random forests are typically easier to interpret than neural networks since their features have a clear meaning right from the start. If the features are discriminant, they can perform extremely well with few training examples and little risk of over-fitting, see Fig. 3 c).

These assets can also turn to disadvantages for more complex tasks. It is indeed possible that no feature enables discriminating different categories, see Fig. 3 e). On their side, neural networks are able to learn the features automatically and therefore perform more diverse and complex tasks than random forests. Overall neural networks tend to perform better, see for instance this comparison. However, this performance may require more training data, especially for large networks. This problem can be mitigated by choosing minimalist architectures when little training data is available or when the classification task is simple. This is why Svetlana offers the possibility to use networks with various number of layers and filter sizes (see Section 4.1).

Finally, we illustrate a problem of *confounding variable* in Fig. 3 g). In this experiment, we attempt to distinguish neural tube nuclei from somites nuclei in a developing quail embryo [3]. In Fig. 3 g), the random forest classifier learned to distinguish the neural tube nuclei using their position in space rather than morphological or radiometric features. Hence, if the image is rotated, the classification becomes catastrophic. Similar issues may happen with convolutional neural networks, but they can be mitigated using the data augmentation mechanism [7] available in Svetlana. In Fig. 3 h), we added random rotations during the training phase to obtain a near rotation invariant classifier.

The interface differences Svetlana was developed under Napari, which offers many advantages for multi-dimensional image processing. First, the environment is supported by a large team and is fast evolving. It is developed in Python and already includes state-of-the-art segmentation tools such as Cellpose of Stardist. It allows using parallel programming (multi-CPU and GPU) in a stable and seamless way. It also makes interactions with PyTorch or TensorFlow easy and therefore benefits from the latest developments in machine learning.

A valuable feature of Napari is its ability to handle large 2D or 3D images. The pyramid image representation indeed enables to interact with them smoothly, while nearly all competitors we tried failed to treat such large datasets. The overall visualization experience under Napari is already excellent for 2D and 3D image processing and can be expected to continue improving.

Svetlana shares many of Ilastik object classification

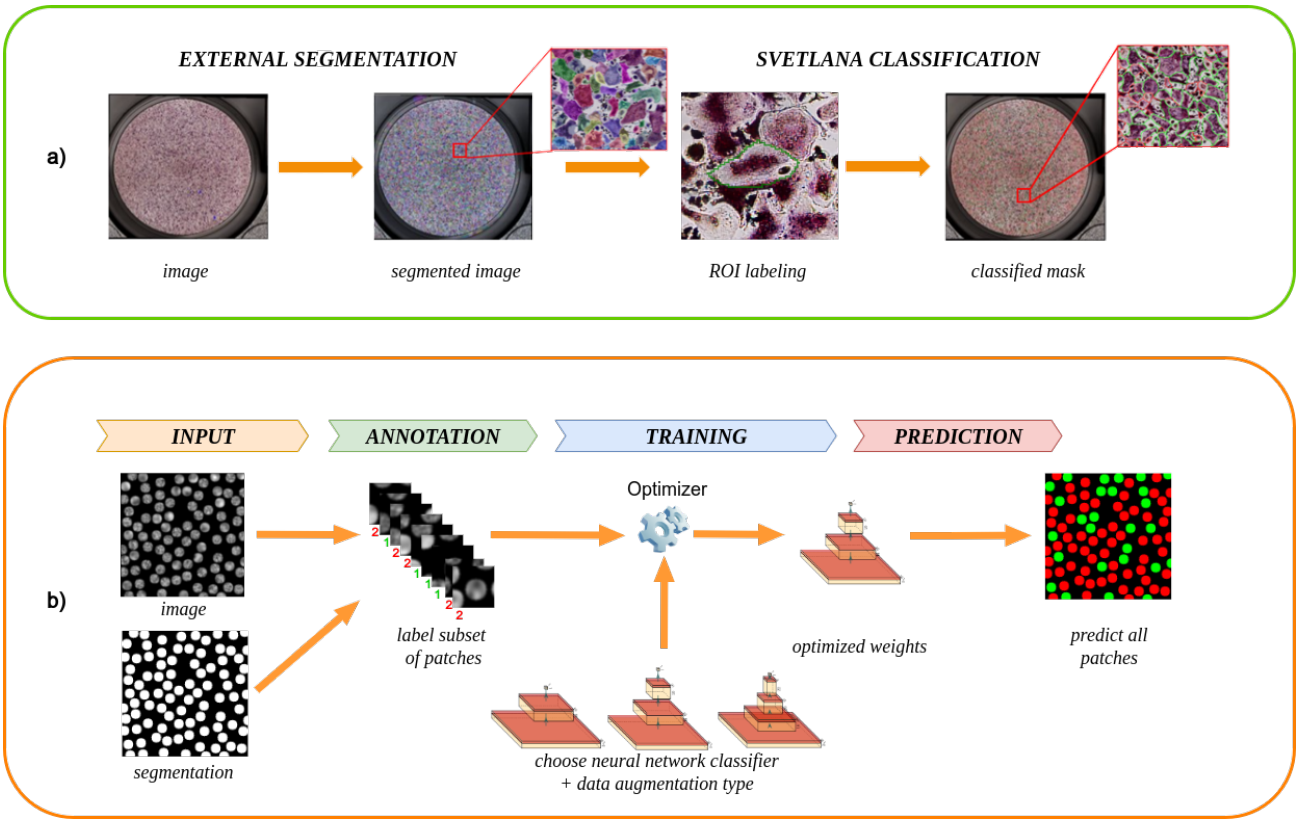

Figure 2: **A schematic overview of the Svetlana plugin. (a)** First, an external segmentation software is used (e.g. Cellpose, Stardist), then Svetlana classifies the resulting masks. **(b)** Overview of the Svetlana's three-step pipeline: Annotation, Training and Prediction. Given pairs of images and segmentation masks, the user labels a few connected components. This first allows training set a neural network classifier and then predicting batches of images.

features. Both interfaces have the same input: an image (or a batch of images) and the associated segmentation mask(s). The trained classifiers can be applied on an individual image or on a batch of images. For the annotation, the user can click on the connected components of the mask or let Svetlana draw connected components uniformly at random. Svetlana does not offer the "live-update" mechanism provided by Ilastik. The main reason is a higher computational cost for training the neural networks.

2.3 Requirements

Once Napari has been set up, Svetlana should be easy to install by following the instructions on the installation page, . It is then available in the plugin manager or through the command line:

```
pip install napari_svetlana
```

Ideally, it should be used with a computer equipped with an Nvidia GPU (Graphical Processing Unit) for faster processing. Remarkably, Svetlana requires little training data (as illustrated in Fig. 3), but also computational power and time resources for the training step. Table 1 summarizes this fact using three different datasets. The energy footprint given below is just indicative. It is computed using the thermal design power (TDP) given by the manufacturer.

- CPU is a 1.9 GHz Intel Xeon in a desktop (TDP: 165W).
- GPU1 is a Quadro T2000 with 4 GB RAM in a laptop (TDP: 40W).
- GPU2 is an RTX4000 with 8 GB RAM in a desktop (TDP: 160W).
- Dataset 1 is the oriented textures presented in Fig. 3 e) (45 labels).
- Dataset 2 is the 3D neural tube image presented in Fig. 5 (169 labels).
- Dataset 3 is the osteoclast dataset presented in Fig. 4 (5400 labels).

	Dataset 1	Dataset 2	Dataset 3
CPU	10 s – 0.5 Wh	6h15 – 1 kWh	5.8 days – 23 kWh
GPU1	8.5 s – 0.1 Wh	6h – 0.24 kWh	26h – 1 kWh
GPU2	4.3 s – 0.2 Wh	45' – 0.1 kWh	18h – 2.9 kWh

Table 1: Computation time and energy consumption in watt-hour (Wh) or kilowatt-hour (KWh) for different hardware configurations.

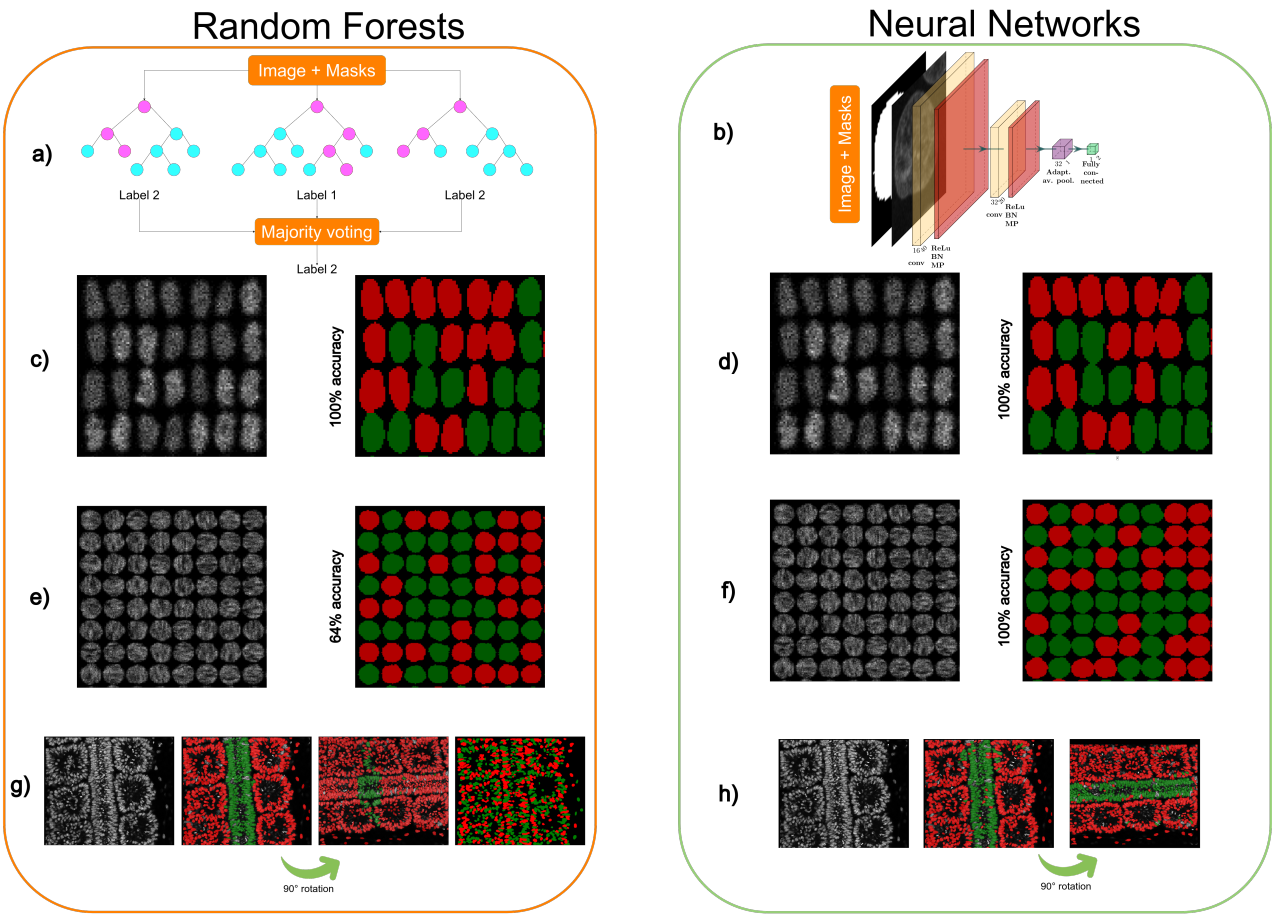

Figure 3: **Random forests (e.g. Ilastik) vs neural networks (Svetlana)** - **(a)** The principle of random forest classifiers: a few (10-1000) random decision trees are constructed and a majority voting allows making a prediction. **(b)** A convolutional neural network classifier makes a prediction by a sequence of convolutions, nonlinear activation functions and pooling. **(c)-(h)** Various classification tasks performed by each classifier. In order to facilitate the visualization, only small portions of the images are displayed. **(c)-(d)** Synthetic cell classification with differences of average gray level. Both Ilastik and a simple neural network yields 100% accuracy with as little as 10 labels. **(e)-(f)** Anisotropic texture classification. The two textures possess the same mean and variance, but different orientations. The random forest yields unsatisfactory results (64%) whatever the number of labels. The neural network yields 100% accuracy with only 45 labels. The reason for the failure of the random forest is that no pre-defined feature allows discriminating the texture orientations. On their side, neural networks are able to learn the right features with just a few annotations. **(g)-(h)** 2D slice of a two-photon confocal microscope image of a quail embryo [3] (courtesy of B. Bénazéraf). It shows a neural tube surrounded by somites. **(g)** From left to right: the slice containing 854 nuclei – the classification result using 35 labels. Thanks to the use of spatialization parameters, Ilastik provides excellent results for this task. However, if the classifier is applied to the rotated image, it yields unsatisfactory results since the spatialization changed. If the spatialization features are removed to construct the decision trees, Ilastik fails to classify the cells. **(h)** Svetlana result with 169 labels: it leads to a few mis-classifications on the original image. Nevertheless, it remains effective when the image is rotated thanks to the use of data augmentation during the training. This experiment overall shows the advantages of learning the classification features and to use data augmentation to add desirable properties such as rotation invariance.

2.4 2D bright-field microscopy

Segmentation in bright-field microscopy can be challenging due to low contrasts, out-of-focus blur and differences in staining across different samples for example. To illustrate that Svetlana overcomes these issues, we solved a cell classification problem used in medium throughput screening on osteoclasts. Osteoclasts are bone cells that go through different stages of differentiation. This gives rise to a wide diversity of cell mor-

phologies as shown in Fig. 4 c. In this experiment, patient osteoclasts are isolated and grown in culture. They are then distributed in wells and subjected to various molecules.

The impact of each treatment is assessed by the activation of osteoclasts, also called osteoclastogenesis. Activated cells are differentiated from others as they are typically large, with a dark violet color and more spots (nuclei) on the inside than the others cells. A few activated osteoclasts are shown in Fig. 4 f), while osteoclasts in different states are shown in Fig. 4 g).

A well contains about 10,000 cells. The classification task is performed by specialists at Atlantic Bone Screen (ABS) and includes a manual counting of activated osteoclasts. This expert approach is accurate, but time consuming for high cell density wells and may require the intervention of several experts to multiply the counts, in order to reduce inter-operator variability, and guarantee a high accuracy. Automating the classification process allows contract research organizations to save time on their projects, and analyze more data while reproducing their selection criteria.

The automatic classification task is really challenging due to a wide variety of image appearances, see Fig. 4 c). These differences can be explained by the biological nature of the sample, but also by the diversity of patients, staining protocols, drug types and concentrations. To address this problem, we first improved the Cyto2 neural network available in CellPose2 [21] using its human-in-the-loop feature. This provided satisfactory segmentation results. We then trained our classifier on 5400 patches extracted from diverse images ($\approx$ 1 hour of manual labeling). We then turned to the prediction mode and applied the neural network classifier to the whole batch. The prediction on 63 images of size 8000×8000 pixels took about 1.5h, i.e. 1.4 minute per image. This has to be compared to the 20 minutes taken by a specialist to process a portion of a single image by hand.

Fig. 4 g) shows that our classifier reaches coefficient of determination R^2 of 96%, between the values measured manually and those produced by Svetlana classifier. For this application, Svetlana is a powerful ally for medium throughput screening. It makes it possible to quantify the effects of various drugs with a good accuracy and with a remarkably small need for human interactions.

2.5 3D fluorescence microscopy

In this paragraph, we aim at illustrating that Svetlana trains efficient classifiers in 3D fluorescence microscopy as well. To this end, we use a $624 \times 158 \times 232$ voxels crop of a large quail embryo image. It was taken with a confocal two-photon microscope with a resolution of $0.68 \times 0.68 \times 1\mu m$. The cropped part contains two structures of interest:

- The neural tube, the axial tissue that will form the spinal cord.
- The somites, the round structures located on either side of the neural tube that will form skeletal muscles and vertebrae.

The classification task we are interested in is to distinguish between nuclei within the neural tube and those within the mesoderm. It is a relatively easy segmentation task since the neural tube is clearly identified by its position in space. This feature allows generating a gold standard by hand to assess Svetlana's accuracy. The segmentation task can arise from two different biological problems:

- Problem a): can we get a rapid and accurate classification of both nuclear types to further quantify biological traits?
- Problem b): can we distinguish both cell types using only the image contents within the immediate neighborhood of the nuclei? This would indicate that nuclei possess photometric or morphometric features allowing to distinguish their class.

The second problem is obviously significantly harder. A quick human inspection at the different nuclei types leads to a negative answer. However, it may have far-reaching consequences. One can think of better understanding the cell migration in development biology or the detection of circulating cancer cells in oncology.

In this work, we used the same approach to tackle both problems:

1. segment the whole volume using the model Cyto2 in CellPose.
2. annotate the 3D image using Svetlana (169 annotations in about 5 minutes).
3. train a two-layer 3D convolutional network ($\approx$ 7 minutes).
4. predict all the nuclei types in the image ($\approx$ 10 minutes).

The only difference lies in the way the patches are treated before training. For Problem a), we directly crop patches of size $45 \times 45 \times 45$ voxels from the complete 3D image. For Problem b), we use the same patches, but multiply them by a dilated mask of the nuclei. This is illustrated on the right of Fig. 5, where a dilation radius of 6 voxels was used. Svetlana offers this pre-processing feature.

The classification accuracy for Problem a) is more than 94% on both nuclear types. Obtaining such a result using only 169 annotations out of $26,214$ is nothing short of remarkable. The average classification rate for Problem b) is about 79%. A visual inspection at the result on Fig. 5 b) might be disappointing. However, this has to be qualified by the fact that this problem seems hopeless at first sight. Notice that, for this example, the image could not be processed by the other software we tested (e.g. Ilastik), since the loading latency is considerable on such a large 3D sample.

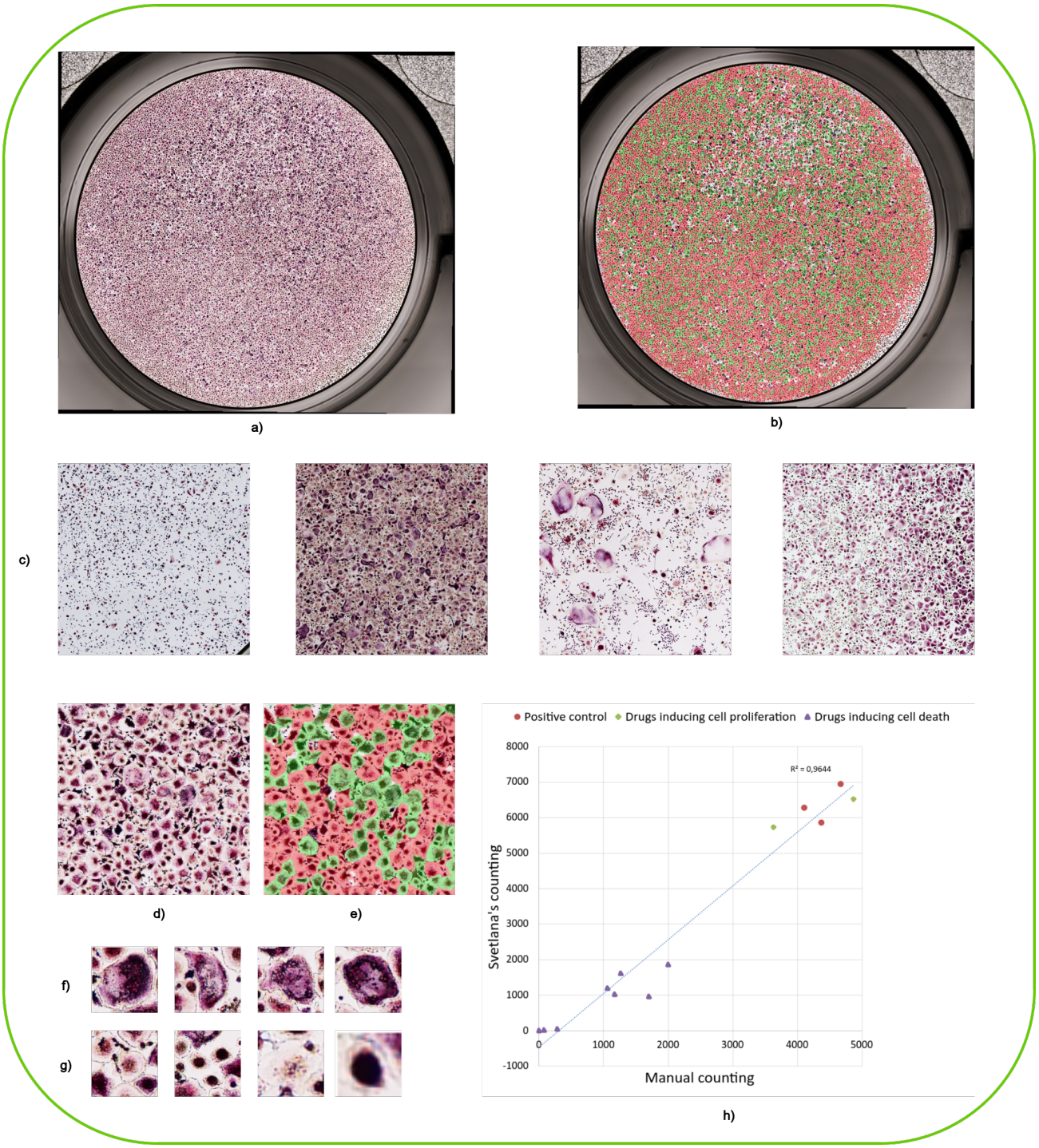

Figure 4: **The osteoclasts classification** - **(a-b)** An example of an entire 8000 × 8000 pixels image and a classification result using Svetlana. The green spots correspond to activated osteoclasts. The image was provided by Atlantic Bone Screen company(ABS) **(c)** 2000 × 2000 pixels crops of four different images illustrating the diversity of the dataset. **(d-e)** 750 × 750 pixels crop and its classification result after training a neural network with 600 annotations (out of 16,671). **(f)** Example of activated osteoclats. **(g)** Example of non-activated osteoclasts. **(h)** This graph shows the excellent correlation (96% coefficient of determination) between Svetlana's counting and a human counting. Various conditions were tested. For the red dots, no drug was applied. For the green diamonds, drugs inducing cell proliferation were tested. For the purple triangles, drugs inducing cell death were introduced.

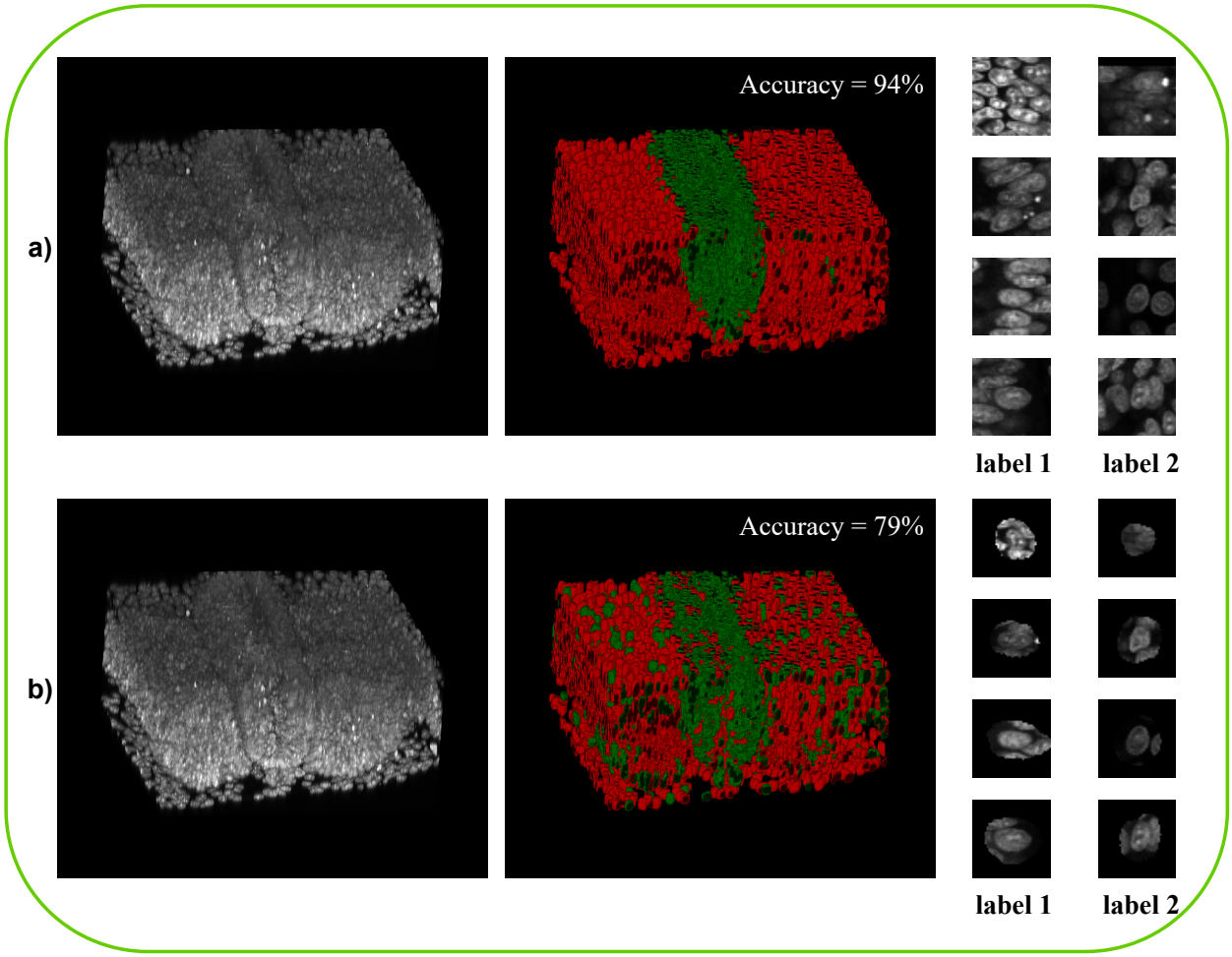

Figure 5: **Classification of 3D nuclei in a quail embryo - (a)** In this experiment, a $45 \times 45 \times 45$ voxels patch with the whole contextual information is extracted around each segmented nuclei. The first column on the right shows 4 nuclei belonging to the neural tube and the right column 4 nuclei from the somites. The image on the left is a 3D volume rendering of the raw image. The classification result is shown on the right. The neural tube is represented in green, and the somites in red. An accuracy of more than 94% is reached for each nuclear type with only 169 annotations in ≈ 7 minutes. **(b)** This experiment is the same as the first one except the patches are pre-processed by multiplying them with a dilated segmentation mask. The contextual information is therefore removed to a large extent. Here, we wish to evaluate to what extent the information contained in a single nucleus and its immediate surrounding is enough to classify it. This challenging problem is still solved with an accuracy of 79%, suggesting that nuclei contain subtle morphometric or optical differences not accessible to the human eye, but distinguishable by a trained computer.

3 Discussion

We presented a new plugin called Svetlana for the classification of segmentation results within the Napari environment. We showed through two applications that Svetlana is a handful tool for challenging cell classification tasks. Neural networks are at the core of the software and tackle problems that would not be solvable with more elementary artificial intelligence tools such as random forests. Svetlana is open-source with a modular architecture allowing to integrate further features, following the most promising technological progress and user feedback. Through this paper, we wish to advertise this tool which should prove valuable to the biomedical community and beyond.

Training with scarce data Training a neural network classifier with just a few annotations (10-1000) in a few seconds goes against conventional wisdom. Indeed, complex architectures are usually trained for days or weeks with huge datasets. For instance, the data science bowl used to train the CellPose models contains 37,333 segmented nuclei in 841 2D images from more than 30 different imaging modalities. It is therefore legitimate to question whether the training phase could yield a good classifier. In practice, it turns out that this approach provides remarkable results with an accuracy sometimes higher than 90% for complex tasks. This may not be on par with the best possible results, but still sufficient for many quantitative analyses in biology.

Let us mention that a few recent works point out that training with a minimal amount of data and *over-parameterized* networks is a rich research avenue [2]. This is exactly the setting explored in Svetlana. As far as we know, complete theoretical explanations are still lacking. One possible way to interpret this is Ockham razor's principle. The neural network architecture together with the training algorithm favor the "simplest" answer capable of explaining the observations. In our experiments, elementary networks tend to perform better than more complicated ones with many parameters. This looks quite natural, since a neural network with few parameters limits the expressiveness of the classifier and acts as a regularizer for the problem.

When high precision is critical, it is very likely that neural networks trained on large datasets would perform better. Unfortunately, such datasets are usually just not accessible. Each biology laboratory explores different tissues, at a different scale with a different modality and focus. Each collected image and labeling 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 [8] allowed training 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 [22] or StarDist [9]) is a huge asset for biology. Unfortunately, as of now, it does not provide classification tools. Hence, Svetlana covers a crucial need which seems yet unmet. In addition, if general purpose classifiers appears in the future, they could be easily integrated and retrained within Svetlana.

Results interpretability Aside from the classification results, another output of Svetlana is a trained neural network. Interpreting this algorithm, i.e., understanding how the decision process was made is a complicated task with many open research avenues. Answering recurrent questions such as: "What makes these populations different? Is it a difference of intensity, volume, ellipticity, or other things I may have missed?" is not directly accessible. Without further analysis, the neural network can therefore be considered as a black-box model, which is unquestionably a limitation of the approach. However, it is important to note that many alternative artificial intelligence models (including linear models or random forests) are often not easier to interpret [16]. In addition, the sole fact to know the existence of discriminating features significantly eases their determination.

That is why the development of interpretation tools has been an active area of research for some years now [1]. Algorithms Grad-CAM [19] or guided Grad-CAM [20] give the user an insight into the areas of the image that are prevalent in the classifier decision. We chose to integrate them in the prediction module of Svetlana, based on their popularity. New routines might be added in the future. Fig. 6 illustrates how Grad-CAM allows the user interpreting how the network might take its decisions on a cancer cell problem.

In addition to Grad-CAM, Svetlana offers the possibility to explore simple hypotheses by using data augmentation. Assessing whether the cell volume plays a role can be determined by adding dilations or not at the training time, and comparing the classification scores. The same experiment can be conducted using different types of intensity normalization, to assess whether intensities and contrasts are discriminant. This is a simple approach to post hoc interpretation.

4 Methods

4.1 The training options

Svetlana provides many options in the training mode. Many of them can be set using the interface, but the most advanced ones should be set using a JSON configuration file. A default version of this file is created when launching the plugin. It can then be modified using any text editor to use more advanced settings. Let us describe the most important ones below.

Available neural network architectures PyTorch provides a large number of popular pre-defined architectures (ResNet [12], DenseNet [13], AlexNet [15] of various depths). They are all available within Svetlana. However, these networks are only implemented in

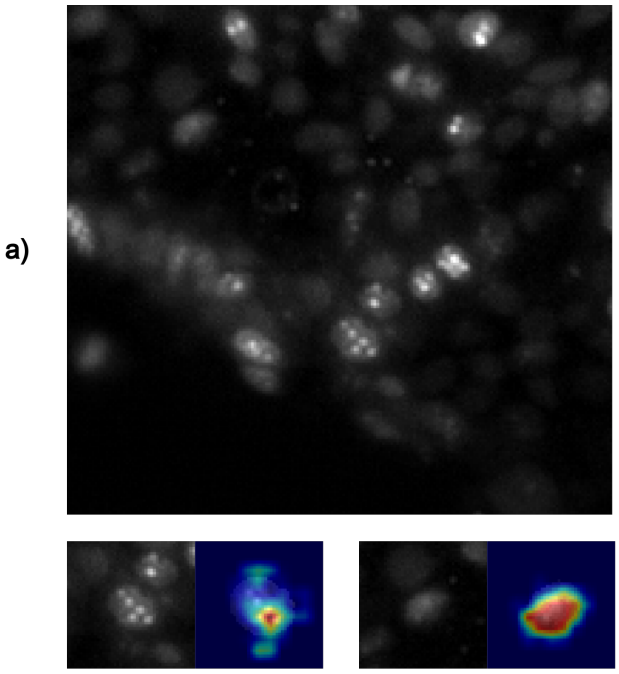

Figure 6: **Computation of Grad-CAM on a MCF-7 cells' image acquired by a light-sheet microscope** - **(a)** The image shows MCF-7 cells (breast cancer cell line) stained with Ki67 (nuclear protein associated with cell proliferation). The proliferating cells have bright spots, while the other ones are darker and more uniform. **(b)** The result of Grad-CAM computation on a proliferating cell: the network focuses on the bright spots. **(c)** The result of Grad-CAM computation on a negative cell. As there are no spots, the network did not focus on any particular region of the cell.

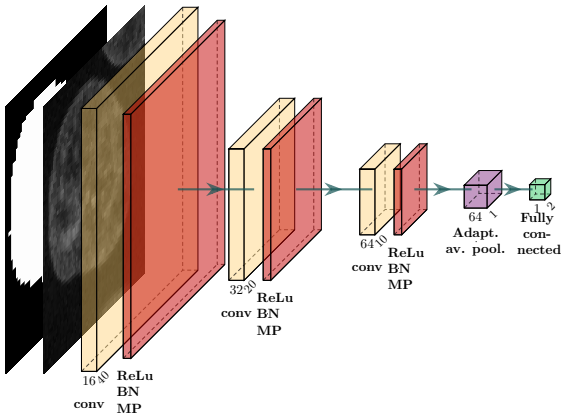

Figure 7: **A light-weight architecture proposed in Svetlana** - This networks contains 3 blocks (depth = 3) and has a width parameter of 16 since the initial image is first duplicated in 16 using the convolutional layer.

2D. Furthermore, they contain millions of parameters, which seems excessive for small datasets.

We therefore designed light-weight 2D and 3D encoder architectures defined through their width and depth. See Fig. 7. They all possess the same elementary blocks of the form Convolution → ReLU activation function → Batch normalization (BN) → Max pooling (MP). Every block reduces the patch size by a factor two with the max-pooling layer, while the width is multiplied by two. After these steps, the output is transformed as a vector of fixed size using an adaptive average pooling layer. To obtain a result of size n_{labels}, we use a fully connected layer followed by a SoftMax. The output can be interpreted as the probability for a patch to belong to each of the classes.

Finally, it is possible to load custom pre-trained PyTorch models and to adapt them to specific datasets using transfer learning.

Optimization routine Currently, Svetlana only provides Adam as an optimization algorithm [14]. It is widely accepted as one of the most versatile method for training neural networks. Its parameters (learning rate and momentum) can be tuned as well as the number of epochs and the batch size.

Our experiments revealed that it is critical to use a step decay during the optimization process. It stabilizes the optimization process significantly. Thus, by default, the models are trained using a decreasing learning rate. It is multiplied by $\gamma = 0.1$ every 100 epochs. Those default parameters can be changed in the configuration file.

Patch pre-processing Svetlana offers various preprocessing steps. The first one is intensity normalization. Different modes are available within the interface:

- No normalization
- Min-max scaling:

$$x = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- Division by the maximum value:

$$x = \frac{x}{\max(x)}$$

Depending whether the pixel intensity value is meaningful or not to discriminate the classes, the user should pick "no normalization" or one of the other two modes.

By default, the input of the network is the image as well as the mask. As illustrated in Fig. 5 b), it might me meaningful to use a different strategy and reduce the impact of contextual information around the object to classify. We therefore added the possibility to multiply the patch by a dilated version of the segmentation mask. This possibility is available only through the configuration file.

Data augmentation A quite unique and useful feature of Svetlana is the possibility to use data augmentation during the training step. This has significant assets. First, it allows a consequent enrichment of the set labeled by the user. Second, it offers desirable properties. For instance, using random patch rotations ensures an approximately rotation invariant response of the network.

To avoid overloading the graphical interface, the augmentation possibilities are limited to vertical and horizontal flips and random rotations. It is however possible to use the much richer set of transformations available within the Albumentations library [7] by setting the configuration file. The transformations are specified in the form of a dictionary, as indicated in the documentation.

Loss functions Svetlana provides a number of loss functions (cross entropy, binary cross entropy, L1 smooth, L1, mean squared error). In practice, for all the experiments presented above, we used a cross entropy loss.

4.2 The influence of the neural network architecture

Choosing the best neural network architecture is a time consuming art, not accessible to non-experienced users. A method that would provide completely different scores depending on the architecture is therefore not acceptable. Hopefully, it turns out that all the minimalist architectures we designed provide fairly similar classification scores. To illustrate this fact, we studied the influence of the architecture in terms of the depth and width parameter for the 3D experiment on the developing quail embryo. Table 2 displays the accuracy scores for every architecture. As can be seen here, all of them provide a fairly similar accuracy. Svetlana makes it easy to test a few different architectures, and visually compare the results. Overall, this experiment reveals that working with a single architecture (e.g. depth 3, width 32) provides consistently good classification results.

	depth	width	acc. 1	acc. 2	avg acc.
	3	32	0,946	0,938	0,9419
	3	64	0,921	0,959	0,9402
	2	16	0,938	0,937	0,9374
	3	16	0,916	0,948	0,9318
	2	64	0,917	0,94	0,9304
a)	2	8	0,906	0,951	0,928
	2	32	0,914	0,936	0,9253
	3	8	0,90	0,935	0,9170
	3	4	0,865	0,961	0,9131
	2	4	0,884	0,936	0,9100
	3	2	0,894	0,91	0,9010
	2	2	0,834	0,925	0,8795
	2	4	0,809	0,766	0,7877
	3	32	0,774	0,778	0,776
	3	8	0,786	0,763	0,7743
	3	4	0,709	0,84	0,7738
	3	64	0,71	0,835	0,7709
b)	3	2	0,691	0,834	0,7624
	3	16	0,706	0,817	0,7614
	2	16	0,7	0,839	0,750
	2	32	0,710	0,776	0,7430
	2	64	0,673	0,783	0,7282
	2	2	0,686	0,747	0,7162
	2	8	0,634	0,698	0,6658

Table 2: Classification accuracy for Problem a) and Problem b) depending on the neural network architecture. In this experiment, we varied the depth and width of the proposed custom 3D convolutional network. All the models have been trained for 600 epochs. The columns acc. 1 and acc. 2 represent the classification accuracy for the neural tube and somites nuclei respectively. They are sorted from the most to the least accurate. As can be seen, the classification rate depends only mildly on the architecture defined through its width and depth.

4.3 The choice of Napari

The objective of Svetlana as a whole is to provide a user-friendly tool to label and classify segmentation results, either manually or automatically. The specifications are:

- Use highly parallel architectures for run-time efficiency.
- Easiness to integrate new classifiers.
- Embed in an environment providing efficient segmentation tools.
- Embed in an environment allowing to visualize the results efficiently.
- Easiness 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 [22]. In addition, the development is currently very fast with new plugins being issued on a weekly basis. It allows interactions with PyTorch or TensorFlow in a limpid manner, and therefore use the latest developments in machine learning.

4.4 The batch mode

Svetlana allows the treatment of a single image or of an entire folder of images for the annotation, training and prediction plugins. It is therefore possible to build a training dataset using several images, possibly from different imaging modalities. Furthermore, it is possible to predict only one image, or to process a whole folder directly. It has been shown on the osteoclasts example (see Section 2.4) that Svetlana has the ability to be trained only on a few images and to generalize

to other ones quite well despite the image acquisition variability.

4.5 The human-in-the-loop feature

The annotation interface offers the possibility to correct the predictions after training. The previous prediction mask is loaded as an overlay and the user can correct or complete his annotation, before restarting a training (online learning). This iterative approach makes it possible – as experienced in Cellpose 2.0 [21] – to quickly improve the classification scores.

Code availability

The source code is provided on BitBucket (`https://bitbucket.org/koopa31/napari_svetlana/src/main/`).

A documentation is available at : `https://svetlana-documentation.readthedocs.io/en/latest/`.

References

[1] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[2] M. Belkin. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica*, 30:203–248, 2021.

[3] B. Bénazéraf, M. Beaupeux, M. Tchernookov, A. Wallingford, T. Salisbury, A. Shirtz, A. Shirtz, D. Huss, O. Pourquié, P. François, and R. Lansford. Multiscale quantification of tissue behavior during amniote embryo axis elongation. *Development*, Jan. 2017.

[4] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, et al. Ilastik: interactive machine learning for (bio) image analysis. *Nature Methods*, 16(12):1226–1232, 2019.

[5] G. Biau and E. Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[6] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[7] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.

[8] J. C. Caicedo, A. Goodman, K. W. Karhohs, B. A. Cimini, J. Ackerman, M. Haghighi, C. Heng, T. Becker, M. Doan, C. McQuin, et al. Nucleus segmentation across imaging experiments: the 2018 data science bowl. *Nature methods*, 16(12):1247–1253, 2019.

[9] E. Fazeli, N. H. Roy, G. Follain, R. F. Laine, L. von Chamier, P. E. Hänninen, J. E. Eriksson, J.-Y. Tinevez, and G. Jacquemet. Automated cell tracking using StarDist and TrackMate. *F1000Research*, 9, 2020.

[10] E. Gómez-de Mariscal, C. García-López-de Haro, W. Ouyang, L. Donati, E. Lundberg, M. Unser, A. Muñoz-Barrutia, and D. Sage. DeepImageJ: A user-friendly environment to run deep learning models in ImageJ. *Nature Methods*, 18(10):1192–1195, 2021.

[11] S. Graham, Q. D. Vu, S. E. A. Raza, A. Azam, Y. W. Tsang, J. T. Kwak, and N. Rajpoot. Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical Image Analysis*, 58:101563, 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[16] Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[18] J. M. Perkel et al. Python power-up: new image tool visualizes complex data. *Nature*, 600(7888):347–348, 2021.

[19] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[20] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[21] C. Stringer and M. Pachitariu. Cellpose 2.0: how to train your own model. *bioRxiv*, 2022.

[22] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods*, 18(1):100–106, 2021.

[23] L. von Chamier, R. F. Laine, J. Jukkala, C. Spahn, D. Krentzel, E. Nehme, M. Lerche, S. Hernández-Pérez, P. K. Mattila, E. Karinou, et al. Democratising deep learning for microscopy with zerocostdl4mic. *Nature communications*, 12(1):1–18, 2021.

5 Acknowledgements

This work was supported by Institut de Mathématiques de Toulouse (IMT), Paul Sabatier University (Toulouse III). C.Cazorla was a recipient of ANRT (Agence Nationale pour la Recherche et la Technologie) in the context of the CIFRE PhD program (N°2020/0843) with Imactiv-3D and Institut de Mathématiques de Toulouse (IMT). P. Weiss acknowledges a support from ANR-3IA Artificial and Natural Intelligence Toulouse Institute ANR-19-PI3A-0004 and from the ANR Micro-Blind ANR-21-CE48-0008. The authors gratefully acknowledge the image.sc community for their precious help, and Atlantic Bone Screen and B. Bénazéraf, for providing some experimental data and valuable scientific discussions, as well as for the constructive feedback on the evaluation of the methodology. They also acknowledge V. Lobjois and L. Guillaume for their precious advice.

6 Author contributions

C.C developed the plugin and the documentation. C.C, P.W. and R.M. conceived the experiments and C.C. realized them. C.C. and P.W. wrote the paper and implemented the classification methods.

7 Competing interests

The authors declare no competing interests