Sketchpose: Learning to Segment Cells with Partial Annotations

Clément Cazorla, Nathanaël Munier, Renaud Morin, Pierre Weiss

Sketchpose: Learning to Segment Cells with Partial Annotations

Clément Cazorla*[1,2], Nathanaël Munier[2], Renaud Morin†[1], and Pierre Weiss‡[2]

[1]Imactiv-3D. Centre Pierre Potier, 1 place Pierre Potier, 31100 Toulouse, France
[2]Institut de Recherche en Informatique de Toulouse (IRIT), Institut de Mathématiques de Toulouse (IMT), Centre de Biologie Intégrative (CBI), Laboratoire de biologie Moléculaire, Cellulaire et du Développement (MCD) Université de Toulouse, CNRS, Université Toulouse III – Paul Sabatier, Toulouse, France

Abstract

A few neural networks in biological image segmentation rely on a prediction of a distance map. This principle is at the basis of popular software such as Stardist, Cellpose or Omnipose. It yields unprecedented accuracy but hinges on *fully annotated* datasets. This can be a serious limitation for generating training sets and performing transfer learning. In this paper, we show how to handle partial annotation, while still relying on the distance map. We design a variant of the Omnipose architecture embedded in a user-friendly Napari plugin. We evaluate the performance of the proposed approach in the contexts of frugal learning, transfer learning and regular learning on a large database. Our experiments show that the proposed approach can lead to substantial savings in time and resources without sacrificing segmentation quality.

Keywords MSC 41A05 – 41A10 – 65D05 – 65D17

Cellpose – Segmentation – Frugal learning – Napari – Deep learning – Distance map

Contents

*clement.cazorla31@gmail.com
†renaud.morin@imactiv-3d.com
‡pierre.armand.weiss@gmail.com

1 Introduction

Image segmentation plays a fundamental role in the analysis of biological images. It enables the extraction of quantitative information on diverse objects ranging from molecules, droplets, membranes, nuclei, cells, vessels or other structures. In modern biological research, accurate segmentation is often pivotal to better understand the mechanisms of life. The increasing availability of high-throughput imaging technologies has led to a surge in the quantity and complexity of image data, raising significant challenges and opportunities. Manual annotation of the resulting images is labor-intensive, time-consuming, and often impractical for large-scale datasets. Automated segmentation is therefore widely accepted as a critical step in biological research.

A simplified history of cell segmentation Image segmentation has long been dominated by handcrafted algorithms. The processing pipelines typically combine popular tools such as linear filtering, thresholding [15], morphological operations [21, 11], active contour models (Snake) [9] or watershed [23]. A significant issue with handcrafted approaches is that they are usually image-specific and rely on the manual tuning of a few complicated hyper-parameters. Although excellent performance can be achieved, it is often the work of a handful of talented people and these techniques are not broadly applicable.

The introduction of machine learning and especially random forests made image segmentation accessible to a much larger range of researchers. These techniques automatically combine and tune elementary image processing bricks. They are driven by a few easily interpretable user annotations. Embedded in well conceived software such as Ilastik [2] or Labkit [1], these techniques heavily contributed to democratize image segmentation and classification.

"Deep learning" and convolutional neural networks played an important role in improving the segmentation performance around 2015. For instance, the popular U-Net architecture [18] increased the accuracy on some cell segmentation challenges by more than 10%, which can be considered as a small revolution. This type of neural network architecture seems to be a good prior for segmenting "natural" images, as suggested by the so-called Deep Image Prior principle [12]. However, it can sometimes demonstrate limited effectiveness when it comes to separating nearby or touching objects. Many applications in biology involve densely packed objects (e.g. cells, nuclei) and a vanilla U-Net is often insufficient to perform a satisfactory analysis. To address this issue, new architectures coming from computer vision such as Mask R-CNN[8] have been developed and continued improving the performance.

Roughly at the same time, a few approaches (Deep Regression of the Distance Map [14], StarDist [20], Cellpose [22], Omnipose[7]) have been developed and generated results with an unprecedented quality. Despite certain differences, they all share a common underlying principle. The idea is to make a regression with respect to some *distance function*. Given a set of annotated objects, a distance function to the objects centers or boundaries is computed. A convolutional neural network is then trained to predict the distance function rather than a binary map of the objects. This principle created a new gap in the segmentation accuracy, especially for objects with touching boundaries. Indeed, the gradient of this distance function points in opposite directions on each side of the boundary, which makes it possible to determine them with much greater precision.

Finally, let us mention that a current trend consists in involving the user in the training procedure. This "human in the loop" principle was recently incorporated in CellPose 2.0 [16].

It would be hazardous to call these approaches the current "state-of-the-art", since this field is expanding extremely quickly. However – as of 2023 – we can safely claim that algorithms based on the distance map are at the basis of some of the most popular and efficient cell segmentation methods.

Contributions This work stems from a practical observation: methods which rely on a regression to the distance function currently require exhaustive annotations. As the distance function is a global geometrical property, adding just a point in the annotation set can dramatically change it everywhere in the image domain. Such an unstability is

illustrated in Figure 1. Hence, it is *a priori* unclear how partial annotations can be used in this framework. It is however much easier to draw a few strokes and boundaries, than to segment *all* the objects in a complex image.

In this paper, we introduce an idea that allows us to use the distance function even with partial annotations. To assess its potential, we develop a Napari plugin [6] named Sketchpose, that relies on a modified version of the Omnipose [7] algorithm. After drawing just a few regions and boundaries, the user can train a task-aware neural network. This approach capitalizes on the generalization capacity of neural networks, reducing the overall annotation effort without sacrificing accuracy. We explore the performance of the proposed architecture in 3 different settings:

- *Frugal learning:* starting from random weights, we show that just a few annotations are already enough to quickly realize complex cell segmentation analyses. This is interesting when faced with a problem for which no close pre-trained model exists.
- *Transfer learning:* starting from Omnipose's optimized weights, we show that just a few clicks at locations where the segmentation is inaccurate lead to improved weights and fast adaptation to out-of-distribution images.
- *Large databases:* finally, we show that large, but partially annotated sets can also be used to optimize neural networks in a robust way.

This comprehensive evaluation on both small and large-scale datasets, overall showcases the advantages of our approach in terms of time and resource savings.

2 Methodology

2.1 Preliminary definitions and notation

In all the paper $\mathcal{X}$ refers to the image domain, which can be understood as a discrete set of coordinates, or as a continuous domain depending on the context. In the discrete setting, we let $|\mathcal{X}|$ denote the number of pixels of $\mathcal{X}$.

Definition 1. *For an arbitrary set $\mathcal{S} \subset \mathcal{X}$, we let $\partial\mathcal{S}$ denote its boundary. We use the 4-connectivity in the discrete setting.*

Definition 2 (Point to set distance)**.** *The distance from a point $\mathbf{x} \in \mathcal{X}$ to a set $\mathcal{S} \subseteq \mathcal{X}$ is defined by*

$$\operatorname{dist}(\mathbf{x}, \mathcal{S}) \stackrel{\text{def}}{=} \inf_{\mathbf{x}' \in \mathcal{S}} \|\mathbf{x} - \mathbf{x}'\|_2. \tag{1}$$

2.2 Omnipose

Our work is based on the Omnipose cell segmentation architecture [7]. In this section, we justify this choice, explain its founding principles and then demonstrate how they can be adapted to deal with partial annotations.

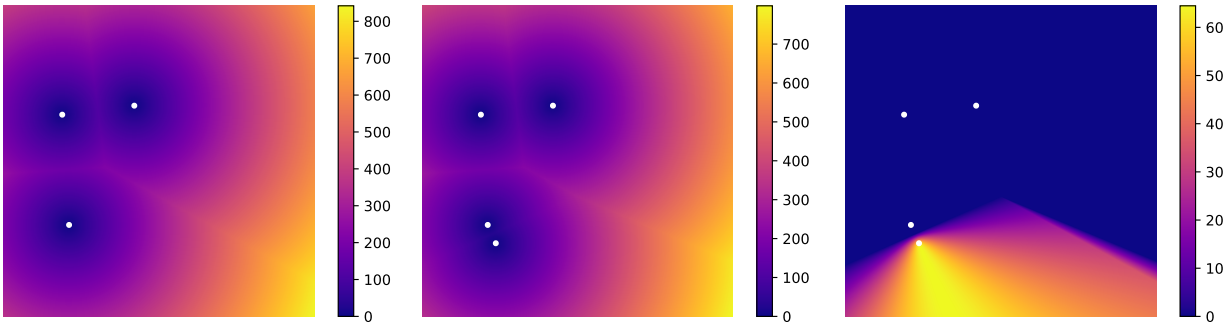
2.2.1 Why Omnipose

Cellpose[22] has now become a standard in cell segmentation. Its excellent perfomance, processing speed, and ergonomic graphical interface make it a near ideal tool for every day cell biology image analysis. However, it occasionally fails in scenarios involving complex and elongated objects. In such cases, it tends to produce over-segmentation, where neighboring objects are split in smaller fragments.

The Omnipose[7] algorithm was initially conceived in order to address this limitation but is currently not as popular as Cellpose. Among the explanations for this phenomenon, we may think of the following facts i) Cellpose was developed earlier, ii) Cellpose is trained continuously with new data coming on a daily basis, while Omnipose is trained on a much smaller and fixed dataset, iii) as a consequence, Omnipose currently produces less accurate results in our experience, except for elongated bacteria datasets.

Omnipose can be seen as an evolution of Cellpose, as some features of the latter architecture have been upgraded. For instance, the distance map is defined as the distance to the cell boundaries in Omnipose, while it is defined as a distance to an arbitrary cell centroid in Cellpose. The problem is that there is no canonical choice to define this center, hence Omnipose's choice seems preferable.

With more developers, training data and further methodological advances, the features of Omnipose might therefore turn it into a serious competitor. This explains our decision to choose and base our work on its architecture.



(a) Distance map to three points (b) With an extra point (c) Difference

Figure 1: Example of unstability of the distance map

2.2.2 The main principles

Fig. 2 summarizes the main ideas behind the Omnipose architecture and its training. Omnipose is based on a regular convolutional neural network (CNN), with a U-Net like architecture [18]. Given an input 2D image with N pixels, the CNN can be seen as a mapping N_w of the form

$$\begin{array}{rccccccc} N_w & : & \mathbb{R}^N & \to & \mathbb{R}^N & \times & \mathbb{R}^N & \times & \mathbb{R}^{2N} \\ & & u & \mapsto & (N_w^b(u) & , & N_w^d(u) & , & N_w^{\mathbf{v}}(u)) \end{array} . \tag{2}$$

It depends on weights w that should be optimized during a training stage. It returns 3 different outputs (illustrated on the top of Fig. 2):

- $N_w^b(u) \equiv$ *boundary probability*: at every pixel, the value of this image can be interpreted as a probability of being a boundary between the objects to segment.
- $N_w^d(u) \equiv$ *distance map*: at a given pixel, the value of this map is equal to:
  - The distance of the pixel to the closest object boundary, if the pixel is inside an object.
  - 0 (or a fixed negative value) elsewhere.
- $N_w^{\mathbf{v}}(u) \equiv$ *flow field*: can be interpreted as the gradient of the distance map. It is an essential feature of the Cellpose and Omnipose architectures. Ultimately, the flow is used through a procedure called Euler integration to generate a segmentation mask. This is illustrated on the top right of Fig. 2.

2.2.3 The training procedure

The original training stage involves a collection of $K \in \mathbb{N}$ images $(u_k)_{1 \le k \le K}$ together with their *exhaustive* segmentation masks. For every image u_k in the dataset, an algorithm creates the gold standard boundary probability $b_k^\star$, distance map $d_k^\star$ and flow field $\mathbf{v}_k^\star$. This is illustrated on the bottom of Fig. 2.

The weights w of the neural network are then optimized so as to minimize a loss function that compares the output of the CNN with the gold standard:

$$\inf_w \text{loss(w)} \stackrel{\text{def}}{=} \frac{1}{K} \sum_{k=1}^{K} \ell_{\mathcal{B}}(b_k, b_k^\star) + \ell_{\mathcal{D}}(d_k, d_k^\star) + \ell_{\mathcal{V}}(\mathbf{v}_k, \mathbf{v}_k^\star), \tag{3}$$

where $b_k = N_w^b(u_k)$, $d_k = N_w^d(u_k)$, $\mathbf{v}_k = N_w^{\mathbf{v}}(u_k)$. The optimization is performed using standard stochastic gradient procedures with the RAdam algorithm [13]. The different losses are defined as follows:

- **Boundary loss $\ell_{\mathcal{B}}$**:

  This loss simply compares the predictions b to $b^\star$ using a binary cross-entropy loss. It is defined as

  $$\ell_{\mathcal{B}}(b, b^\star) \stackrel{\text{def}}{=} \frac{\lambda_{\mathcal{B}}}{|\mathcal{X}|} \sum_{\mathbf{x} \in \mathcal{X}} g(b[\mathbf{x}], b^\star[\mathbf{x}]), \tag{4}$$

  where $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ combines a sigmoid and a binary cross entropy loss. It corresponds to the function BCEWithLogitsLoss in the package torch.nn.

- **Distance loss $\ell_{\mathcal{D}}$**:

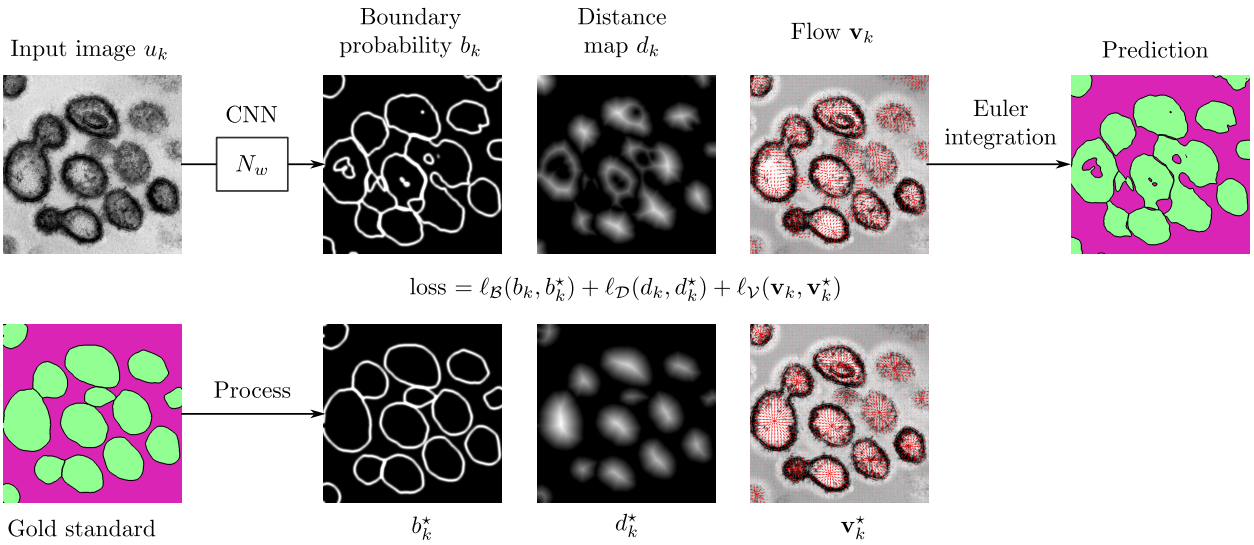


Figure 2: A sketch of the architecture and training procedure in Omnipose

This loss calculates a weighted mean squared error between the predicted distance fields and the ground truth distance fields. It is defined as

$$\ell_{\mathcal{D}}(d, d^\star) \stackrel{\text{def}}{=} \frac{\lambda_{\mathcal{D}}}{|\mathcal{X}|} \sum_{\mathbf{x}\in\mathcal{X}} (d[\mathbf{x}] - d^\star[\mathbf{x}])^2 \cdot \rho[\mathbf{x}],$$

where $\rho \in \mathbb{R}^N$ is a weight image with higher values around the gold standard boundaries. We will use $\rho = \beta + \exp(-d^\star/\alpha)$ in our numerical experiments. The parameter $\alpha > 0$ can be interpreted as a characteristic distance, measuring how far from the boundary we want d to match $d^\star$ closely. The parameter β allows the used to balance the relative importance of the boundaries. We pick $\alpha = 5$ and $\beta = 2$ in our implementation.

- **Flow loss** $\ell_{\mathcal{V}}$:

  This loss is defined as a weighted sum of three losses $\ell_{\mathcal{V}} = \ell^1_{\mathcal{V}} + \ell^2_{\mathcal{V}} + \ell^3_{\mathcal{V}}$. The first one is a mean squared error loss:

$$\ell^1_{\mathcal{V}}(\mathbf{v}, \mathbf{v}^\star) \stackrel{\text{def}}{=} \frac{\lambda_{\mathcal{V},1}}{|\mathcal{X}|} \sum_{\mathbf{x}\in\mathcal{X}} \left\|\mathbf{v}[\mathbf{x}] - \mathbf{v}^\star[\mathbf{x}]\right\|_2^2 \cdot \rho[\mathbf{x}]. \quad (5)$$

The second one compares the norms of the vector fields:

$$\ell^2_{\mathcal{V}}(\mathbf{v}, \mathbf{v}^\star) \stackrel{\text{def}}{=} \frac{\lambda_{\mathcal{V},2}}{|\mathcal{X}|} \sum_{\mathbf{x}\in\mathcal{X}} (\|\mathbf{v}[\mathbf{x}]\|_2 - \|\mathbf{v}^\star[\mathbf{x}]\|)^2 \cdot \rho[\mathbf{x}]. \quad (6)$$

The third one aims to minimize the distance between trajectories generated through the ground truth and predicted flows. Trajectories starting from an initial point $\mathbf{z}$ can be generated by simple explicit Euler discretization:

$$\mathbf{x}_0(\mathbf{z}) \stackrel{\text{def}}{=} \mathbf{z}$$
$$\mathbf{x}_{l+1}(\mathbf{z}) \stackrel{\text{def}}{=} \mathbf{x}_l(\mathbf{z}) + \Delta t \cdot \mathbf{v}[\mathbf{x}_l(\mathbf{z})]$$

$$\mathbf{x}^\star_0(\mathbf{z}) \stackrel{\text{def}}{=} \mathbf{z}$$
$$\mathbf{x}^\star_{l+1}(\mathbf{z}) \stackrel{\text{def}}{=} \mathbf{x}^\star_l(\mathbf{z}) + \Delta t \cdot \mathbf{v}^\star[\mathbf{x}^\star_l(\mathbf{z})].$$

where Δt is a step-size. Letting $L \in \mathbb{N}$ denote an integration time, the "Euler" loss then becomes:

$$\ell^3_{\mathcal{V}}(\mathbf{v}, \mathbf{v}^\star) \stackrel{\text{def}}{=} \frac{\lambda_{\mathcal{V},3}}{|\mathcal{X}|} \sum_{\mathbf{z}\in\mathcal{X}} \sum_{l=1}^{L} \|\mathbf{x}_l(\mathbf{z}) - \mathbf{x}^\star_l(\mathbf{z})\|_2^2. \quad (7)$$

It measures how two trajectories generated by Euler integration using the ground truth and predicted vector

fields deviate. This loss is implemented in the torchVF library by [17]. For more information, we refer the reader to the related report.

An inspection of the code reveals that the different weights have been set empirically as: $\lambda_{\mathcal{B}} = 10$, $\lambda_{\mathcal{D}} = 2$, $\lambda_{\mathcal{V},1} = 2$, $\lambda_{\mathcal{V},2} = 2$, $\lambda_{\mathcal{V},3} = 1$.

Remark 1. *The different losses probably have been combined by trial and error to produce the best possible results. It seems however that some simplifications might be good to consider. For instance, the flow field can be obtained from the distance map directly. Hence, the network could likely be simplified to return the boundary probability and flow only. The loss $\ell^1_{\mathcal{V}}$ is an upper-bound of $\ell^2_{\mathcal{V}}$. The loss $\ell^1_{\mathcal{V}}$ is also present already in the first term of $\ell^3_{\mathcal{V}}$. We decided to stick to these choices to perform some numerical comparisons.*

2.3 Adapting to partial annotations

All the principles described above heavily depend on an exhaustive segmentation of the cells. Indeed, the distance functions and gradient flows – which are instrumental to define the loss functions – are global properties which do change heavily if the objects boundaries are incomplete. In this section, we describe the main methodological contribution of this paper, which will allow us to handle partial boundaries.

2.3.1 The gold standard

We assume that the domain $\mathcal{X} = \mathcal{X}_0 \sqcup \mathcal{X}_1$ is partitioned with the background set $\mathcal{X}_0$ and the foreground set $\mathcal{X}_1$. A difficulty in instance segmentation is that multiple objects may exist within the connected components of a region $\mathcal{X}_i$. To differentiate them, we let $(\mathcal{X}_{i,j})_{1 \leq j \leq J_i}$ denote a partition of the set $\mathcal{X}_i$ as different objects within a similar class. For instance in Fig. 3a, the foreground set $\mathcal{X}_1$ is split in 13 components. A connected component of $\mathcal{X}_1$ can be split as $\mathcal{X}_{1,2} \cup \mathcal{X}_{1,3}$. The background set $\mathcal{X}_0$ is split in a single component $\mathcal{X}_{0,1}$.

We let

$$\mathcal{E} \overset{\text{def}}{=} \bigcup_{i \in \{0,1\}} \bigcup_{j=1}^{J_i} \partial \mathcal{X}_{i,j} \tag{8}$$

denote the set of all *edges* (or object boundaries) within the image. It is depicted in red in Fig. 3a.

2.3.2 The annotation set

The input of our neural network is a set of "strokes" drawn by the user. We let $\mathcal{S}_0$ and $\mathcal{S}_1$ denote those strokes describing the background and foreground respectively. They are depicted in brown and blue respectively in Fig. 3b. The intersection of the brown and blue strokes define natural boundaries. We can indeed construct the *touching boundaries* $\mathcal{B}_{0,1}$ between different strokes as

$$\mathcal{B}_{0,1} \overset{\text{def}}{=} \overline{\mathcal{S}_0} \cap \overline{\mathcal{S}_1},$$

where $\overline{\mathcal{X}}$ is the closure of $\mathcal{X}$ in the continuous setting and the interface between neighboring pixels in the discrete setting.

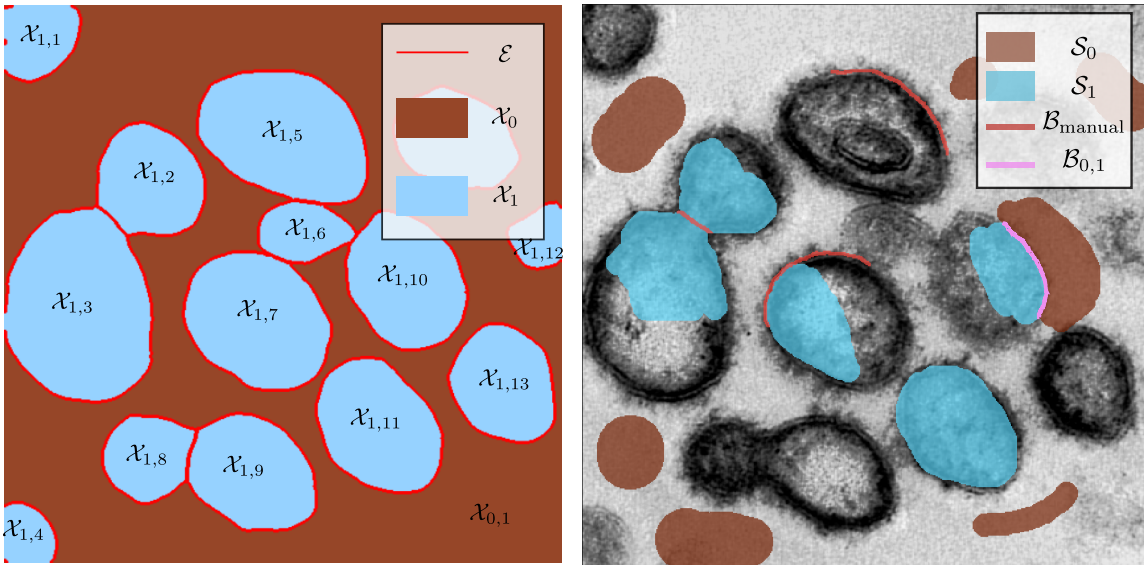
In addition, the user can delineate other boundaries, denoted $\mathcal{B}_{\text{manual}}$, to separate touching objects within a class. We can concatenate all the boundaries to obtain a complete boundary set $\mathcal{B}$ defined as

$$\mathcal{B} = \mathcal{B}_{\text{manual}} \bigcup \mathcal{B}_{0,1}. \tag{9}$$

For the algorithm to work properly, we require the following set of assumptions.

Assumption 1 (Assumptions on the strokes)**.**

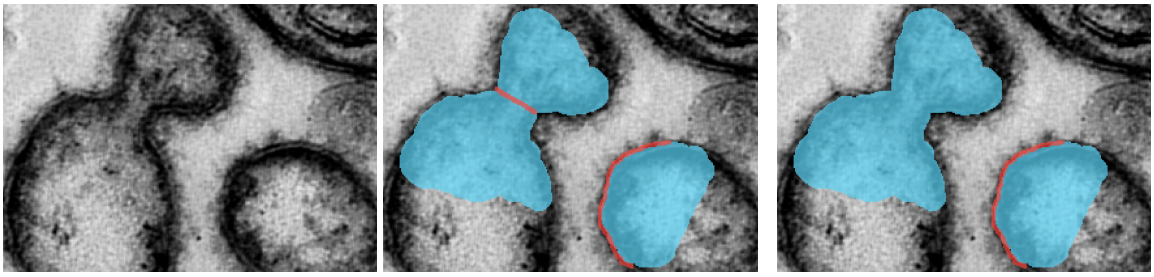
- *The strokes correctly separate the background and foreground: $\mathcal{S}_0 \subseteq \mathcal{X}_0$ and $\mathcal{S}_1 \subseteq \mathcal{X}_1$.*
- *The strokes do not overlap: $\mathcal{S}_0 \cap \mathcal{S}_1 = \emptyset$. This is actually forced by our Napari interface.*
- *The boundaries $\mathcal{B}$ are a subset of the exact boundaries $\mathcal{E}$, that is:*
$$\mathcal{B} \subseteq \mathcal{E}. \tag{10}$$
- *If the stroke $\mathcal{S}_i$ contains multiple objects, then the boundaries between the objects* need to be completely drawn *with $\mathcal{B}_{manual}$ (see Fig. 4). Letting $\mathcal{S} \overset{\text{def}}{=} \mathcal{S}_0 \cup \mathcal{S}_1$ denote the complete stroke set drawn by the user, this condition reads:*
$$\mathcal{E} \cap \mathcal{S} \subseteq \mathcal{B}. \tag{11}$$



(a) Gold-standard & corresponding definitions

(b) An admissible ground-truth annotation

Figure 3: Ground-truth and an associated admissible annotation set



(a) Image

(b) Admissible

(c) Not admissible

Figure 4: If a stroke contains multiple objects, the object boundaries have to be drawn. In this example, two nuclei are present under the blue bow-tie-shaped region. Therefore, a manual boundary has to be added in the center

2.3.3 The main observation

The main result we will use to define and certify our algorithm is summarized in the following theorem.

Theorem 1 (The valid distance set)**.** *Let* $C\mathcal{B} \stackrel{\text{def}}{=} \partial\mathcal{S}_0 \cup \partial\mathcal{S}_1 \cup \mathcal{B}$ *denote the complete set of annotation boundaries and define the* valid distance set $\mathcal{D}$ *as*

$$\mathcal{D} \stackrel{\text{def}}{=} \{x \in \mathcal{S}, \operatorname{dist}(x, \mathcal{B}) \leq \operatorname{dist}(x, C\mathcal{B})\}. \tag{12}$$

The following relationships hold:

$$\begin{aligned} \textit{For all} \quad & x \in \mathcal{D}, \quad \operatorname{dist}(x, \mathcal{E}) = \operatorname{dist}(x, \mathcal{B}). \\ \textit{For all} \quad & x \in \mathcal{S}_0 \cup \mathcal{S}_1, \quad \operatorname{dist}(x, C\mathcal{B}) \leq \operatorname{dist}(x, \mathcal{E}). \end{aligned}$$

The proof of this theorem is given in Appendix 5. This theorem should be understood as follows. The first identity informs us that we can compute the *exact distance map* $\operatorname{dist}(x, \mathcal{E})$ to the set of exact boundaries $\mathcal{E}$ on the valid set $\mathcal{D}$. This set can be computed using only the

partial annotations of the boundaries $\mathcal{B} \subseteq \mathcal{E}$ and the different semantic regions $\mathcal{S}_i \subseteq \mathcal{X}_i$. The second inequality tells us that we have an information everywhere on the strokes $\mathcal{S}_0$ and $\mathcal{S}_1$. Moreover, in the case of total annotations, we get $\mathcal{D} = \mathcal{X}$ and the proposed idea will lead to a training equivalent to the one in Omnipose. Figure 5 schematically summarizes Theorem 1.

2.3.4 Adapting the training

Different summation sets Equipped with the valid distance set $\mathcal{D}$, we are ready to adapt the losses to cope with partial annotation. In Omnipose, the losses $\ell_{\mathcal{B}}$, $\ell_{\mathcal{D}}$, $\ell^1_{\mathcal{V}}$ and $\ell^2_{\mathcal{V}}$ are defined by summation over the set $\mathcal{X}$ (see paragraph 2.2.3). With partial annotation, the gold standard is not properly defined on this set and we therefore need to change the summation sets.

First, we propose to modify the weight vector w by defining it as:

$$\rho = \exp(-\alpha d_{C\mathcal{B}}) + \beta, \tag{13}$$

where $d_{C\mathcal{B}}$ is the distance to the complete boundary set.

For the loss $\ell_{\mathcal{B}}$, we know the partial boundaries $\mathcal{B}$ and that there should be no boundaries in the strokes $\mathcal{S}_i$, hence we can sum over $\mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{B}$:

$$\ell^{\text{partial}}_{\mathcal{B}}(b, b^\star) = \frac{\lambda_{\mathcal{B}}}{|\mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{B}|} \sum_{\mathbf{x} \in \mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{B}} g(b[\mathbf{x}], b^\star[\mathbf{x}]). \tag{14}$$

Based on Theorem 1, the losses related to the distance set and to the flows become:

$$\ell^{\text{partial}}_{\mathcal{D}}(d, d^\star) \overset{\text{def}}{=} \frac{\lambda_{\mathcal{D}}}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} (d[\mathbf{x}] - d^\star[\mathbf{x}])^2 \cdot \rho[\mathbf{x}],$$

$$\ell^{1,\text{partial}}_{\mathcal{V}}(\mathbf{v}, \mathbf{v}^\star) \overset{\text{def}}{=} \frac{\lambda_{\mathcal{V},1}}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \left\|\mathbf{v}[\mathbf{x}] - \mathbf{v}^\star[\mathbf{x}]\right\|_2^2 \cdot \rho[\mathbf{x}],$$

$$\ell^{2,\text{partial}}_{\mathcal{V}}(\mathbf{v}, \mathbf{v}^\star) \overset{\text{def}}{=} \frac{\lambda_{\mathcal{V},2}}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} (\|\mathbf{v}[\mathbf{x}]\|_2 - \|\mathbf{v}^\star[\mathbf{x}]\|_2)^2 \cdot \rho[\mathbf{x}].$$

We simply replaced $\sum_{\mathbf{x} \in \mathcal{X}}$ by $\sum_{\mathbf{x} \in \mathcal{D}}$ in section 2.2.3. This means that we compare the ground truth and prediction only where it makes sense to do so. Since $\mathcal{X} = \mathcal{D}$ in the case of exhaustive annotation, the training of Omnipose and ours coincide and we can see our setting as a generalization of Omnipose.

The Euler loss A specific care needs to be taken for the Euler integration loss $\ell^3_{\mathcal{V}}$. In that case, we need to replace the summation over $\mathcal{X}$ by a summation over $\mathcal{D}$, but this is insufficient since trajectories might escape the valid distance set during integration. To avoid this, we mask the ground truth $\mathbf{v}^\star$ and predicted $\mathbf{v}$ vector fields outside of the valid set $\mathcal{D}$. Therefore, the trajectories generated from points in $\mathcal{D}$ by Euler integration will at stop at the boundary of $\mathcal{D}$. In equations, this yields:

$$\mathbf{x}_0(\mathbf{z}) = \mathbf{z}$$

$$\mathbf{x}_{l+1}(\mathbf{z}) \overset{\text{def}}{=} \begin{cases} \mathbf{x}_l(\mathbf{z}) + \Delta t \cdot \mathbf{v}[\mathbf{x}_l(\mathbf{z})] & \text{if } \mathbf{x}_l(\mathbf{z}) \in \mathcal{D} \\ \mathbf{x}_l(\mathbf{z}) & \text{otherwise} \end{cases}$$

with a similar modification for $\mathbf{x}^\star_l$. The loss then becomes

$$\ell^{3,\text{partial}}_{\mathcal{V}}(\mathbf{v}, \mathbf{v}^\star) \overset{\text{def}}{=} \frac{\lambda_{\mathcal{V},3}}{|\mathcal{D}|} \sum_{\mathbf{z} \in \mathcal{D}} \sum_{l=1}^{L} \|\mathbf{x}_l(\mathbf{z}) - \mathbf{x}^\star_l(\mathbf{z})\|_2^2. \tag{15}$$

Dealing with inequalities Until now, we just used the first identity in Theorem 1, but the second inequality brings some additional information. We propose to integrate it in the training through the additional asymmetric loss:

$$\ell^{\text{ineq}}_{\mathcal{D}}(d, d^\star) \overset{\text{def}}{=} \frac{\lambda_{\mathcal{D},\text{ineq}}}{|\mathcal{S}_1 \cup \mathcal{S}_2|} \sum_{\mathbf{z} \in \mathcal{S}_1 \cup \mathcal{S}_2} \text{ReLU}^2\left(d[\mathbf{z}] - d^\star[\mathbf{z}]\right) \rho[\mathbf{z}]. \tag{16}$$

2.4 The Sketchpose plugin

A significant part of this work lies in the development of a user-friendly graphical interface to train and use the neural network. It is integrated in Napari[6], which is well suited to embedding the Python/PyTorch codes at the core of our approach.

Sketchpose can be easily installed through either the pip package manager or Napari's[6] built-in interface. A detailed documentation can be accessed at: `https://sketchpose-doc.readthedocs.io/en/latest/index.html`. It offers step-by-step instructions illustrated by short videos, to assist users in effectively testing all the capabilities of the plugin.

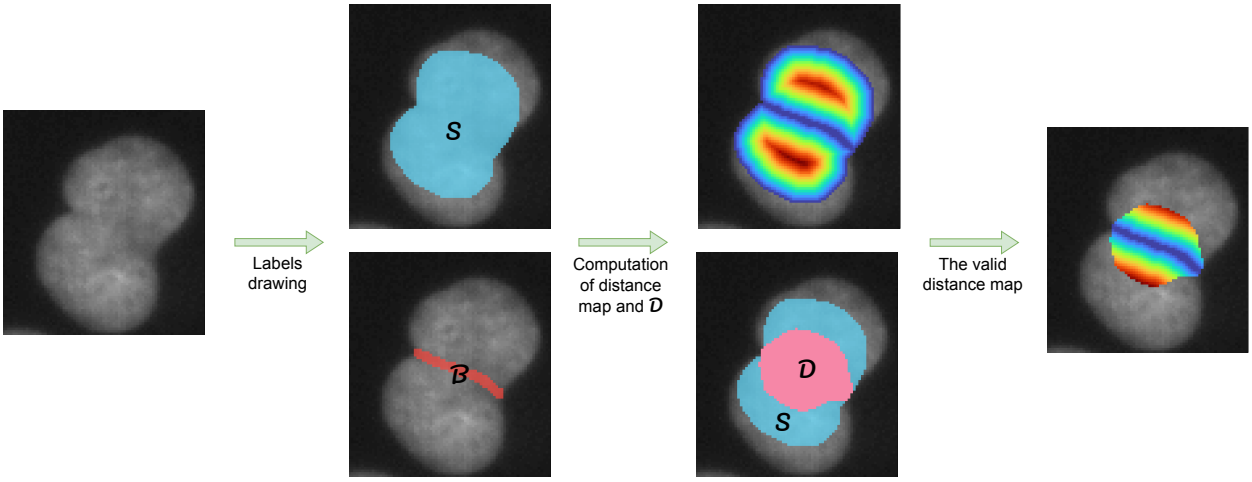


Figure 5: Illustration of the valid distance set theorem. All the pixels in $\mathcal{D}$ are closer to $\mathcal{B}$ than to the boundaries of $\mathcal{S}$. The colormap used to represent the distance map exhibits a progressive shift in colors, transitioning from blue to red

The networks can be initialized by random weights or existing pre-trained weights. The multi-threaded plugin's architecture makes it possible to annotate, train and observe the current segmentation results simultaneously. The user can annotate regions where the segmentation is inaccurate in priority, hence reducing the annotation time. The predictions can be restricted to a bounding box at each epoch of the training process, to reduce the processing time, which is particularly helpful for large scale images. Finally, users can work with a single image or a set of images for the inference and training steps.

Two approaches can be used for labeling. In the first one the user directly draws a few strokes for the background, foreground and boundaries. The brush size can be adjusted similarly to usual paint software. An entire stroke boundary can be added to the boundary set by a double right-click. The second approach is similar to the one implemented in Cellpose 2.0 [16]. After an initial prediction, the user manually chooses a bounding box and performs a full labeling in the corresponding area. This is more tedious and the first option should be preferred. An overview of the interface is depicted in Fig. 6.

3 Experiments

In this paragraph, we conduct several experiments to explore three distinct use cases of the method.

- Learning from a limited set of annotations on a single image with randomly initialized neural network weights.
- Learning from a limited set of annotations on a single image, starting from a pre-trained neural network.
- Learning with randomly initialized weights using a large dataset with sparse annotations. We study the impact of the percentage of labeled pixels (10%, 25%, 50% and 100%) on the segmentation quality, when training on thousands of cells.

After describing the metrics used for validation, we will turn to the practical results.

3.1 Evaluation metrics

To quantify the predictions quality, we enumerate the true positives (TP), the true negatives (TN) and the false positives (FP). A true positive is an object in the gold standard that can be matched to an object in the prediction with an Intersection over Union (IoU) criterion higher than a

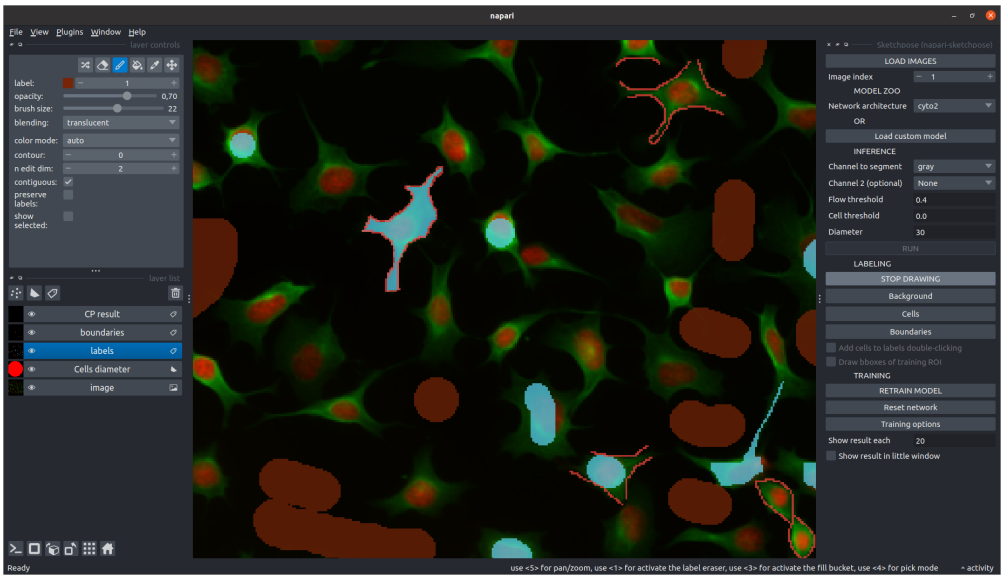


Figure 6: The Sketchpose plugin in annotation mode

threshold τ. We let $TP(\tau)$ denote the total number of true positives. The total number of estimated objects without matches is denoted $FP(\tau)$ (for false positives). The total number of gold standard objects without valid matches is denoted $FN(\tau)$ (for false negatives). Utilizing these values, we compute the standard average precision metric ($AP(\tau)$) for each image using the formula:

$$AP(\tau) = \frac{TP(\tau)}{TP(\tau) + FP(\tau) + FN(\tau)}.$$

The reported average precision is then computed as the average over all images in the test set.

This evaluation process corresponds to utilizing the *matching_dataset* function from the StarDist[20] package, with the *by_image* option set to True. Additionally, we computed the average DICE and the Aggregated Jaccard Index defined as follows:

$$J_{\text{aggregated}}(A, B) = \frac{1}{N} \sum_{i=1}^{N} \frac{|A_i \cap B_i|}{|A_i \cup B_i|},$$

$$\text{average DICE}(A, B) = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \cdot |A_i \cap B_i|}{|A_i| + |B_i|},$$

where A and B are a dataset and its groundtruth.

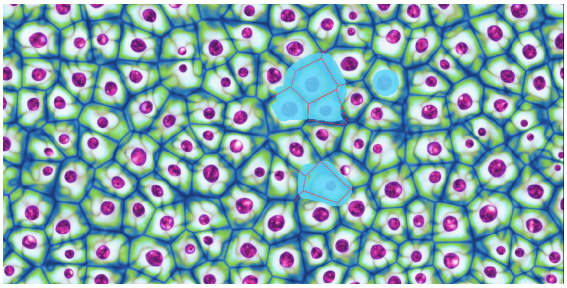
3.2 Training from scratch on a single image

In this section, we will showcase several results achieved while training from scratch on a small set of images. All the gold standards in this section and the next were generated by us and validated by biologists. The tests are made on a variety of biological structures (dendritic cells, osteoclasts, bacteria, insect eggs, adipose tissue, artistic image of cells).

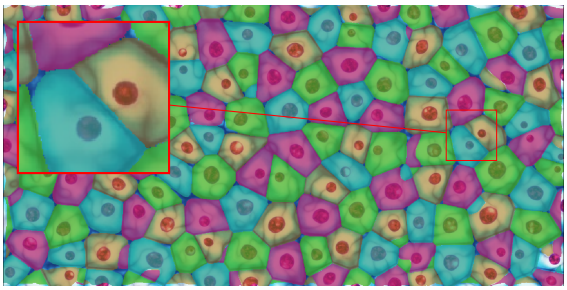
3.2.1 Cells (artistic paint)

In the example of Figure 7, we use an artistic cell representation to showcase the possibility to reach satisfactory segmentation results in short annotation and computing times. On this painting, the nuclei are shown in red and the cytoplasmic membrane in green. To facilitate visualization and ensure that two contiguous areas do not have the same color when labeled differently, we used the 4-color theorem library called ncolor developed by K. J. Cutler.

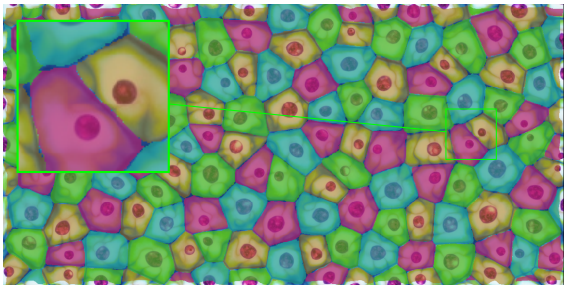
After drawing for less than one minute and training for 100 epochs (2 minutes), we achieve an excellent result with no over segmentation, contrarily to the trained model of Omnipose (see Figure 7). The quality metrics is shown



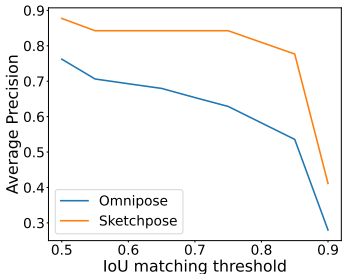
(a) The sparse labels



(b) Omnipose cyto2 model



(c) Sketchpose result, trained from scratch (≈ 2 minutes)



(d) Evaluation of the segmentation quality. Omnipose: DICE=0.99, Jaccard index=0.91. Sketchpose: DICE=0.95, Jaccard index=0.90

Figure 7: (a,b,c) A training from scratch with sparse labels. Image Credit: Eduard Muzhevskyi. The colormap has been changed to improve visualization. (d) Evaluation of segmentation quality.

in Figure 7d.

3.2.2 Eggs on a tree leaf

In this section, we picked an image from the Omnipose dataset, which likely represents eggs of an insect on a tree leaf. At first sight, the segmentation task is uneasy, since the objects are tightly connected, with identical textures and blurry boundaries. We first annotated a subset of 5 eggs in Fig. 8b with a minimal amount of background. The segmentation result after training is already surprisingly good in Fig. 8d, but some objects are not detected, and others are merged. We annotated 2 extra eggs in Fig. 8c. With this extra information, retraining the network now produces a near perfect segmentation mask, with a single error (2 pink eggs on the left). This experiment illustrates a unique feature of Sketchpose: it is possible to interactively annotate while training. This offers a possibility to label a minimum amount of regions to reach the desired output. This principle sometimes called "active learning" or "human-in-the-loop" [4] is significantly enhanced by using partial annotations and the user-friendly Napari interface.

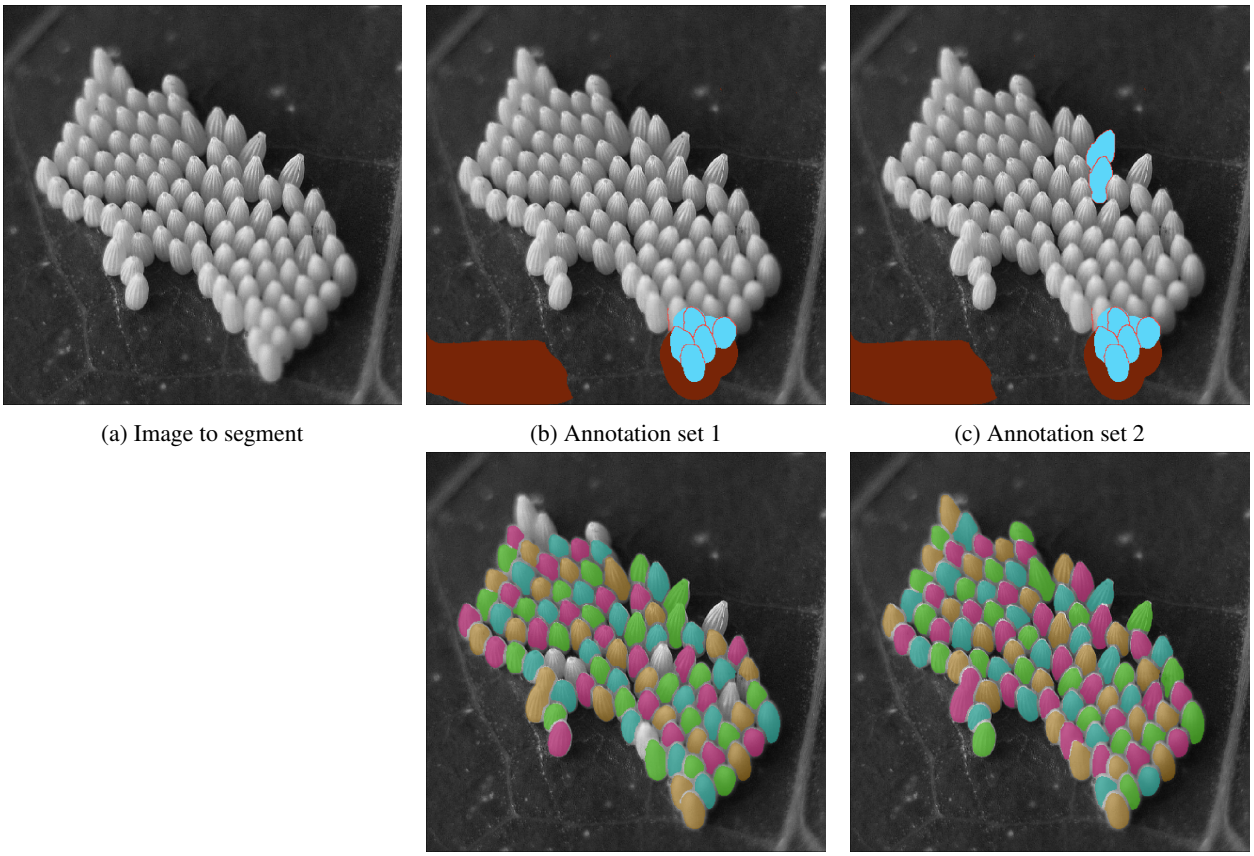
3.3 Transfer learning on a single image

In this section, we explore the feasibility of improving pre-trained weights using transfer learning.

3.3.1 Bacteria segmentation

Bacteria are often used as biological models (e.g. in DNA studies). A precise segmentation can be difficult to achieve, because they have elongated shapes and can be clustered.

The Omnipose[7] model was initially conceived to address the shortcomings of Cellpose for this task.



(a) Image to segment

(b) Annotation set 1

(c) Annotation set 2

(d) Sketchpose, from scratch (≈ 10 minutes)

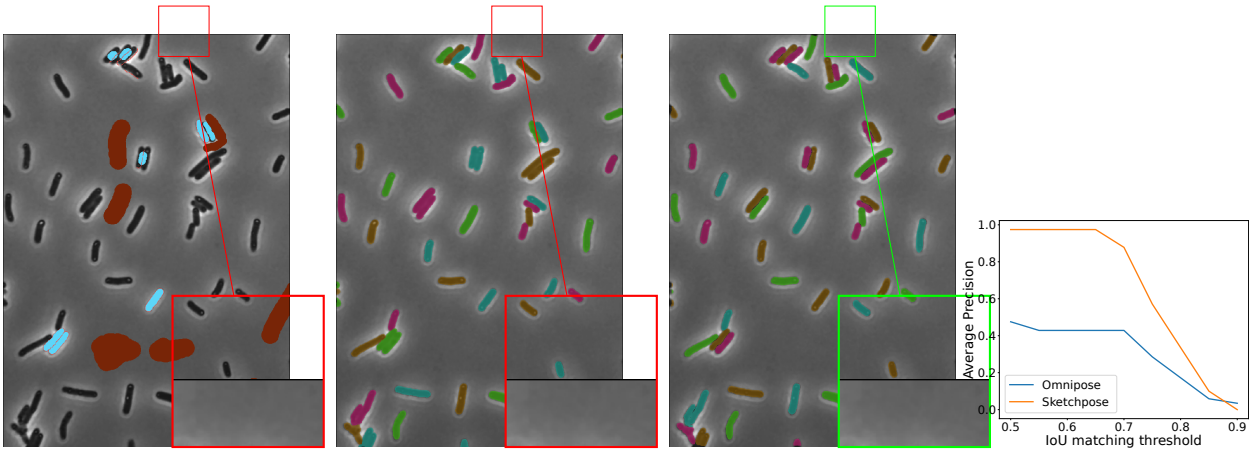
(e) Sketchpose, retrained adding annotations (≈ 3 extra minutes)

Figure 8: Progressive training in Sketchpose. In this example, we show that it is possible to improve the segmentation performance of Sketchpose by progressively annotating at places where the network failed. Here, a quite minimal annotation set, is enough to near perfectly separate the eggs on the leaf.

Figure 9 shows how transfer learning with sparse annotations can improve the Omnipose results by separating touching bacterias. Figure 9d shows a quantitative comparison of both methods. As can be seen, Sketchpose's adapted weights provide much higher performance. A visual inspection indicates that all objects have been correctly separated, apart from the cluster touching the boundary on the bottom left.

3.3.2 Adipocytes segmentation

The image in Fig. 10d shows a crop of a very large image of a skin explant provided by DIVA Expertise. One can see a part of the dermis (in pink) and above it, adipose tissue (large white circular cells). Adipose tissue is the third skin layer after the epidermis and dermis, also known as the hypodermis. Hypodermal cells (adipocytes) secrete specific molecules (e.g. adiponectin, leptin) which have a direct impact on the biology of fibroblasts present in the dermis, and also on keratinocytes present in the epidermis. They are the subject of numerous studies (see [3]



(a) The sparse labels

(b) Omnipose bact_phase model

(c) Sketchpose, transfer learning (≈ 1 minute)

(d) Evaluation of segmentation quality. Omnipose: DICE=0.89, Jaccard index=0.69. Sketchpose: DICE=0.89, Jaccard index=0.79

Figure 9: Improvement of bacteria segmentation starting from the Omnipose bact_phase model. The zoomed-in view shows an under-segmentation resolved by Sketchpose transfer learning. Image from [24]

and [19] for instance). For most of the studies where skin explants are imaged, we first need to count the adipocytes number in the image, and remove any potential outliers detected in the dermis and epidermis.

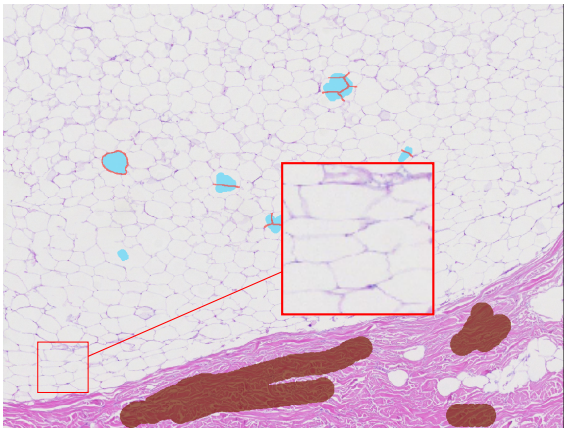
While Omnipose cyto2 results in some undersegmentation for this task, the adapted weights provided by Sketchpose yields significantly enhanced results. Annotating 6 cells and a training for 100 epochs (less than 1 minute) were sufficient to significantly improve the quality of the segmentation and to remove the outliers from the dermis (see Figure 10). Figure 10d shows a quantitative comparison between Omnipose and Sketchpose on this example.

3.3.3 Osteoclasts segmentation

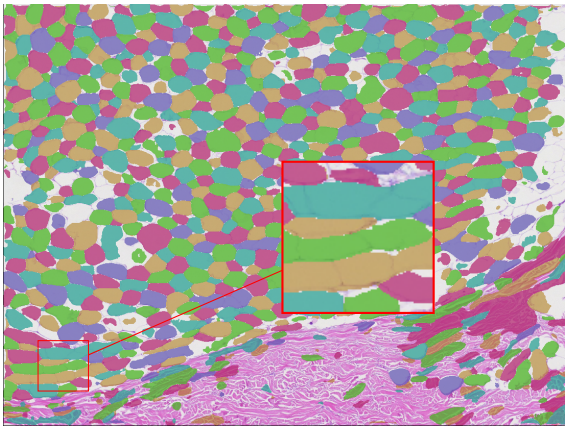
Osteoclasts are responsible for bone resorption, and are widely studied (see [10] for instance) as being responsible for certain pathologies such as osteoporosis when dysfunctional. Their differentiation goes through several stages, culminating in the activated osteoclast. The latter is generally large and contains numerous nuclei. Atlantic Bone Screen (ABS) company is investigating the effect of different drugs in inducing either proliferation or cell death in these activated osteoclasts, in order to regulate their population. To do so, they extract osteoclasts from biopsies, culture them, apply the drugs and image them under a bright-field microscope.

The studied image is a crop of an image containing around 20,000 cells. We can see touching cells presenting a great variety in size, shape color. The image is complex to segment and poses a real challenge. What is more, ABS does not want to count pre osteoclasts (small black nuclei), but only the mature cells (according to specific nuclei criteria). Each study comprises around sixty images, hence manual counting task performed at ABS is costly and laborious.

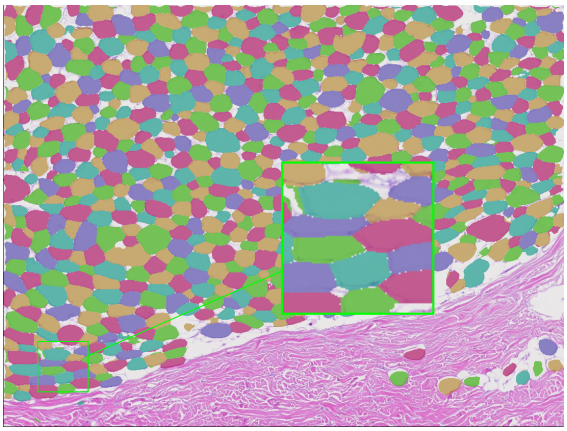
In Figure 11, we present a qualitative depiction that underscores the enhancement in segmentation accuracy attained through transfer learning with just a few labels. Labeling required approximately 2 minutes, while the training process took about 5 minutes. A quantitative comparison is available in Figure 11d.



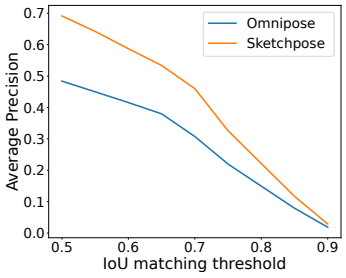
(a) A crop of the huge adipocytes image



(b) Omnipose cyto2 model segmentation



(c) Sketchpose, transfer learning (≈ 2 minutes)



(d) Evaluation of the segmentation quality. Omnipose: DICE=0.90, Jaccard index=0.68. Sketchpose: DICE=0.95, Jaccard index=0.80

Figure 10: (a,b,c) Improvement of adipocytes segmentation starting from the Omnipose cyto2 model. (d) Evaluation of segmentation quality.

3.4 Training from scratch on a large dataset

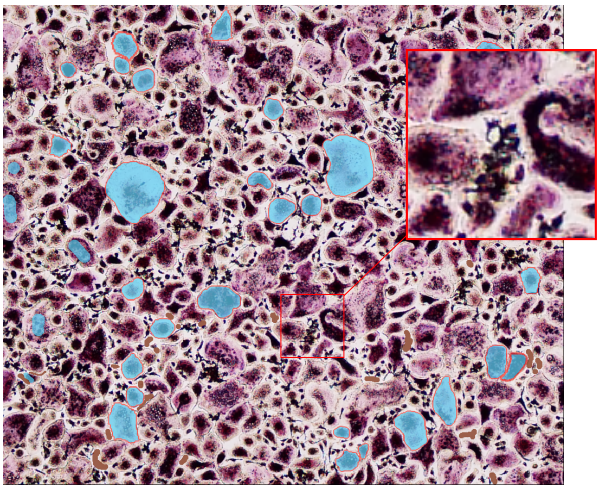
The aim of this experiment is to highlight the possibility to train our model on a large dataset with sparse annotations.

3.4.1 The Cellpose dataset

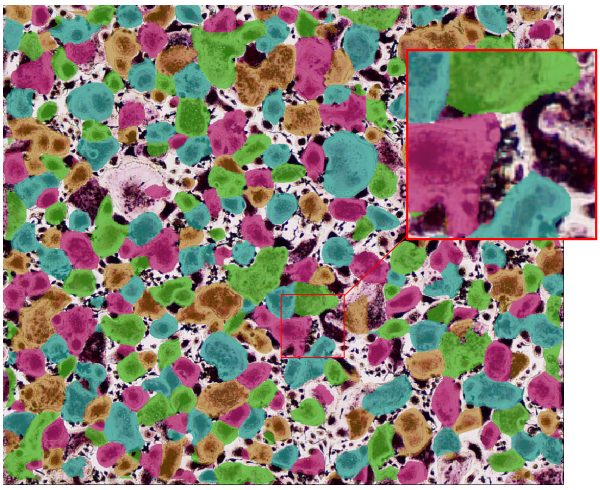
To compare trainings with full or partial annotations, we used the Cellpose dataset, which was initially conceived with the data science bowl [5]. This dataset used for training and evaluation is a carefully curated compilation of 540 images, each capturing various cellular morphologies and imaging scenarios. A sample of this dataset is presented in Figure 12. Notice that the latest Cellpose models were trained using a considerably enriched dataset. To the best of our knowledge, this larger dataset is not publicly accessible.

3.4.2 Selecting annotation subsets

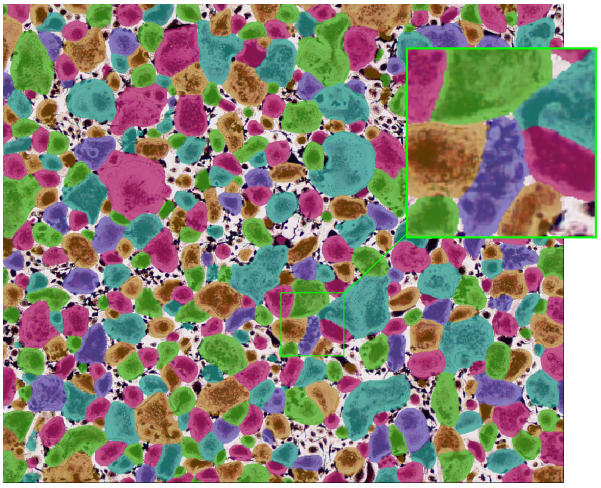
In this section, we investigate the model robustness across various annotation levels each characterized by a different percentage of annotated pixels: 10%, 25%, 50%, and 100%. We generate randomly binary masks by threshold-



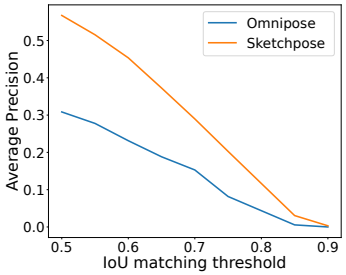
(a) The sparse labels



(b) Omnipose cyto2 segmentation



(c) Sketchpose, transfer learning (≈ 5 minutes)



(d) Evaluation of segmentation quality. Omnipose: DICE=0.81, Jaccard index=0.53. Sketchpose: DICE=0.90, Jaccard index=0.72

Figure 11: (a,b,c) A training with a few labels on osteoclasts. (d) Evaluation of segmentation quality.

ing white Gaussian noise with a Gaussian filter of variance σ^2. The resulting Gaussian process is then thresholded to keep only a given proportion of pixels.

While the model is stochastic in nature, the generated data is created once and for all, enabling its deterministic reuse across multiple training sessions. Figure 13 shows an example of training image from which we have kept 25% of the label mask pixels.

3.4.3 Results

After training four models, 500 epochs each (2.5 days per model with a single Nvidia RTX5000) on the whole dataset with the four percentages of annotated pixels we described above, we evaluated the performance through our three metrics (see Figure 14).

Surprisingly, it seems that training with 100%, 50% or 25% of labeled pixels lead to near identical performance. The performance seems to break down for "10"%

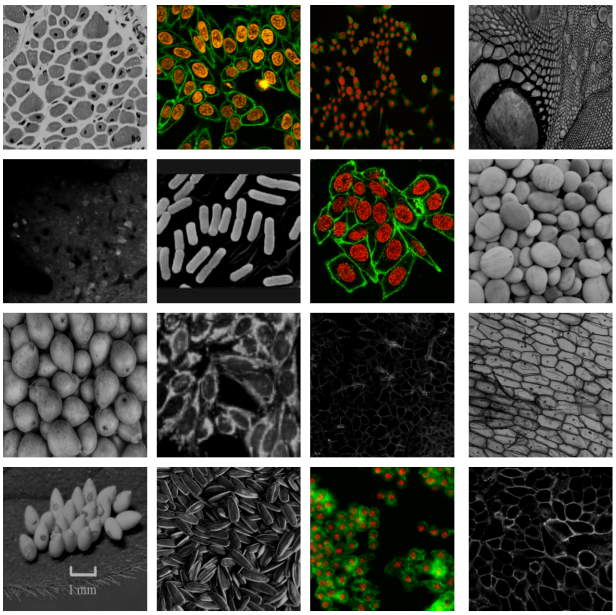


Figure 12: Some image crops extracted from Cellpose dataset, showing its variability. In addition to cells, we can observe various objects (e.g., fruits, pebbles, scales)

annotation, yet the average precision (56%) decreased by only 16% compared to the best model (72%) with an IoU threshold of 50%. As a conclusion, it seems that training with only 25% of annotated pixels is a good strategy to save labelling time without degrading the segmentation quality. Figure 15 shows an example of how segmentation quality declines as the percentage of annotated pixels decreases.

4 Discussion & conclusion

We introduced Sketchpose, an open-source plugin to extend the applicability of Cellpose and Omnipose to partial annotations. From a methodological aspect, we developed a theory making it possible to use distance functions, despite having only access to partial information on the objects boundaries. From a more practical viewpoint, we developed an interactive interface within Napari, which facilitates efficient online learning with a real-time visualization of the training progress. The multi-threaded implementation allows users to continue annotating while the neural network trains or infers.

The new training procedure was tested in three different frames: i) training a neural network from scratch and just a few strokes, ii) improving the weights of a pre-trained network (a.k.a. transfer learning or human in the loop), iii) training with massive, but partial annotations.

For point i), frugal annotation seems to works surprisingly well despite really limited information. Just a few strokes are already enough to provide results on par – or better – than pre-trained networks.

For point ii), our experiments demonstrated the potential benefits of using transfer learning. That is, starting with a pre-trained Omnipose models, we can further refine it using our methodology.

As for point iii), we showed that training the model with just one fourth of the annotations leads to near identical performance, with a substantial reduction in annotation time. This suggests that partial, but less time-intensive annotations, might be an effective strategy to generate training sets without sacrificing the quality of the final model.

The plugin also shows a few limitations. First, the method would benefit from faster training times to make the method even more interactive. We plan to improve this aspect in the forthcoming versions. Second, it is important to mention that our formalism is currently restricted to the two dimensional setting with two labels (background/foreground). The proposed strategy could be used if the user was able to delineate a surface surrounding the objects of interest, but not just curves. In 3D, this would result in an empty valid distance set (see Theorem 1) and unadapted loss functions. This limitation of the method must be put into perspective by the fact that even the Cellpose 3D model is based on 2D predictions only, which are aggregated in post-processing.

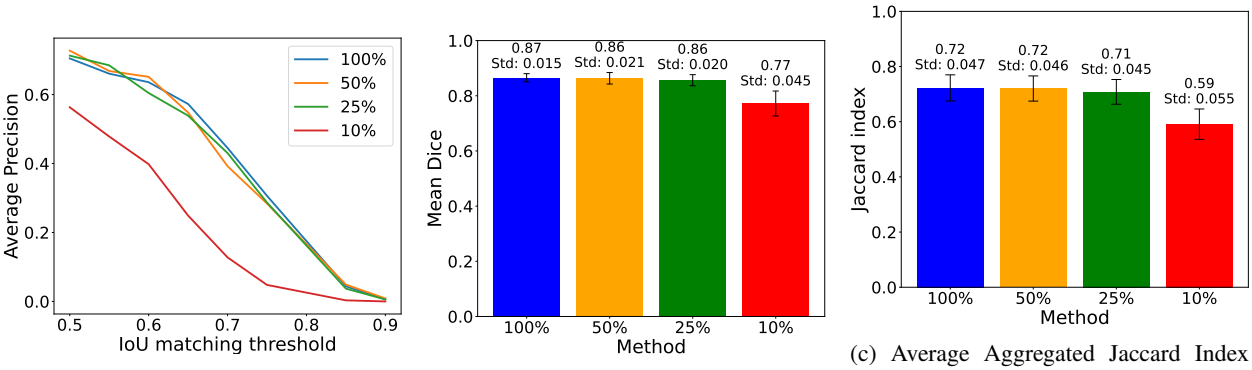
In summary, the proposed method demonstrated numerous qualities in 2D for partial annotations, but further developments are needed to accelerate the training process and for an multi-class extension in 3D.

Acknowledgments

We are grateful for the information provided by Kevin John Cutler about the original Omnipose implementation.

(a) A raw training image (b) The sparsity mask (σ = 100) (c) The sparse labels

Figure 13: A training image with only 25% of labeled pixels



(a) Average precision over the test dataset (b) Average DICE over the test dataset (c) Average Aggregated Jaccard Index over the test dataset

Figure 14: Evaluation of segmentation precision over Cellpose's test dataset as a function of the percentage of annotated pixels. (b) and (c) represent the mean dice and the mean Jaccard index over the test dataset respectively. We also show the standard deviations computed over the whole test dataset.

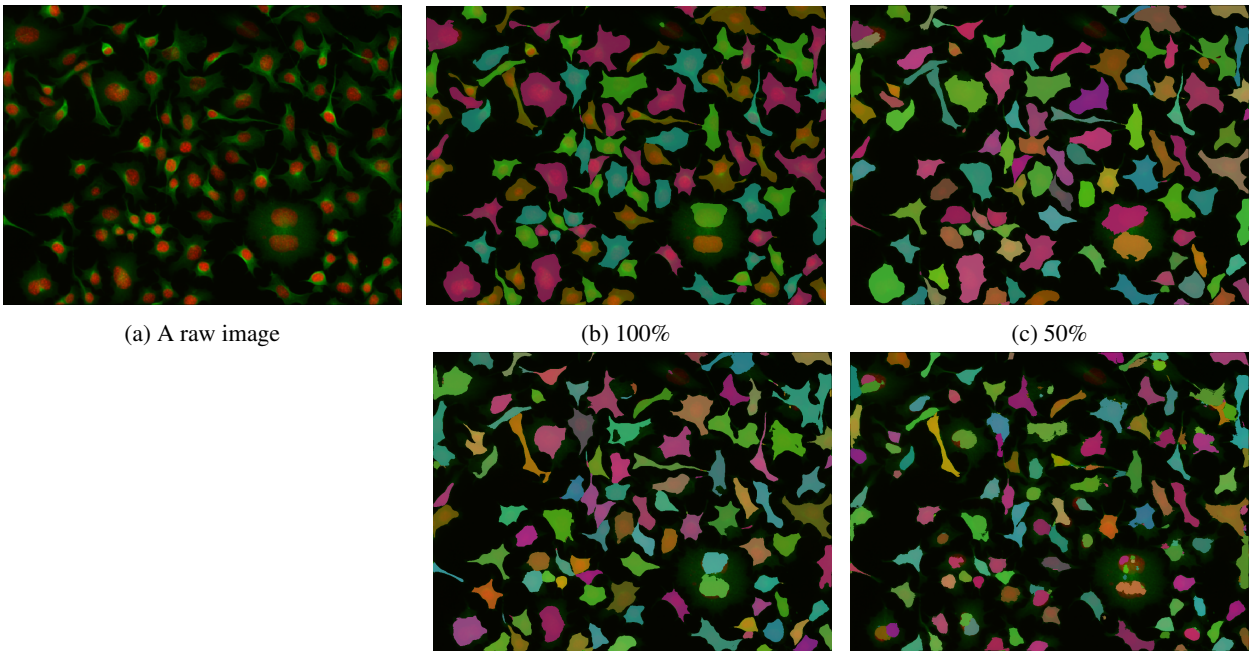
The authors would like to thank the authors of Cellpose and Omnipose for making their datasets available. The authors acknowledge Atlantic Bone Screen for providing the osteoclasts image and DIVA Expertise for providing the adipocytes image.

Funding Statement

C.Cazorla was a recipient of ANRT (Agence Nationale pour la Recherche et la Technologie) in the context of the CIFRE Ph.D. 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. This work was performed using HPC resources from GENCI-IDRIS (Grant 2021-AD011012210R1).

Source code and documentation

The code is available there: `https://bitbucket.org/koopa31/napari-sketchpose/src/master/`. The documentation is available there: `https://sketchpose-doc.readthedocs.io/en/latest/`.



(a) A raw image (b) 100% (c) 50%

(e) 25% (f) 10%

Figure 15: An example of segmentation precision loss on a Cellpose test dataset image when percentage of annotated pixels in the training dataset decreases from 100 to 10%. One can see segmentation quality highly decreases with 10% of labeled pixels

Appendix

5 Proof of the valid distance set theorem

We start with a basic observation.

Proposition 1 (Properties of the distance function)**.**

- $\mathcal{A}_1 \subset \mathcal{A}_2 \Rightarrow \forall \mathbf{x} \in \mathcal{X}, \text{dist}(\mathbf{x}, \mathcal{A}_2) \leq \text{dist}(\mathbf{x}, \mathcal{A}_1)$.
- $\mathcal{A}_1 \subset \mathcal{A}_2$ *and* $\mathbf{x} \in \mathcal{A}_1 \Rightarrow \text{dist}(\mathbf{x}, \partial\mathcal{A}_1) \leq \text{dist}(\mathbf{x}, \partial\mathcal{A}_2)$.

Proof. The first item is direct:

$$\text{dist}(\mathbf{x}, \mathcal{A}_1) = \inf_{\mathbf{x}' \in \mathcal{A}_1} \text{dist}(\mathbf{x}', \mathbf{x}) \geq \inf_{\mathbf{x}' \in \mathcal{A}_2} \text{dist}(\mathbf{x}', \mathbf{x}) = \text{dist}(\mathbf{x}, \mathcal{A}_2).$$

Here is one proof of the second iten by separating the two cases: either $x \in \mathring{\mathcal{A}}_1$ or $x \in \partial\mathcal{A}_1$.

- *Case 1:* $x \in \partial\mathcal{A}_1$. This case is trivial since $\text{dist}(\mathbf{x}, \partial\mathcal{A}_1) = 0 \leq \text{dist}(\mathbf{x}, \partial\mathcal{A}_2)$ by positivity of the distance.
- *Case 2:* $x \in \mathring{\mathcal{A}}_1$. In that case, the key argument is to show that the open ball of radius $\text{dist}(\mathbf{x}, \partial\mathcal{A}_1)$ centered in x is included in $\mathring{\mathcal{A}}_1$. Precisely

$$\tilde{\mathcal{B}} \overset{\text{def}}{=} \mathcal{B}\left(x, \text{dist}(\mathbf{x}, \partial\mathcal{A}_1)\right) \subseteq \mathring{\mathcal{A}}_1 \subseteq \mathring{\mathcal{A}}_2. \tag{17}$$

Indeed having Equation (17) established implies by contraposition that

$$\partial\mathcal{A}_2 \subseteq \mathring{\mathcal{A}}_2{}^c \subseteq \tilde{\mathcal{B}}^c \tag{18}$$

where the first inclusion is given by $\partial\mathcal{A}_2 \overset{\text{def}}{=} \bar{\mathcal{A}}_2 \setminus \mathring{\mathcal{A}}_2 \subseteq \mathring{\mathcal{A}}_2{}^c$. Therefore, taking infimum with respect to these sets, implies the following inequalities and by the way

the intended result.

$$\begin{aligned}\operatorname{dist}(\mathbf{x}, \partial\mathcal{A}_2) &= \inf_{z\in\partial\mathcal{A}_2} \|z-x\| \geq \inf_{z\in\mathring{\mathcal{A}}_2{}^c} \|z-x\| \\ &\geq \inf_{z\in\tilde{\mathcal{B}}^c} \|z-x\| = \operatorname{dist}(\mathbf{x}, \partial\mathcal{A}_1).\end{aligned}$$

So let's prove Equation (17): by contradiction, assume that there exists $z \in \tilde{\mathcal{B}} \cap \mathring{\mathcal{A}}_1{}^c$. Notice that $[x, z] \cap \mathring{\mathcal{A}}_1{}^c$ is a compact set (here $[x, z]$ denotes the closed segment between the points x and z). Thus

$$z^* \stackrel{\text{def}}{=} \operatorname*{arg\,min}_{x'\in[x,z]\cap\mathring{\mathcal{A}}_1{}^c} \|x - x'\|$$

is well defined and the semi-open segment $[x, z^*[$ is included in $\mathring{\mathcal{A}}_1$. This implies that $z^* \in \partial\mathcal{A}_1$ since the sequence $z_n \stackrel{\text{def}}{=} x + (1 - \frac{1}{n})(z^* - x) \in [x, z^*[\subseteq \mathring{\mathcal{A}}_1$ converges to z^*. The contradiction comes from

$$\operatorname{dist}(\mathbf{x}, \partial\mathcal{A}_1) \leq \|z^* - x\| \leq \|z - x\| < \operatorname{dist}(\mathbf{x}, \partial\mathcal{A}_1).$$

The first inequality holds because $z^* \in \partial\mathcal{A}_1$, the second one because $z^* \in [x, z]$ and last one because $z \in \tilde{\mathcal{B}}$.

By proof by contradiction, Equation (17) holds.

In conclusion, in all cases the inequality is verified. □

Theorem 1 can be proven in two steps. First, notice that the inclusion $\mathcal{B} \subseteq \mathcal{E}$ (Assumption 1) and the first bullet in Proposition 1 implies that $\operatorname{dist}(\mathbf{x}, \mathcal{E}) \leq \operatorname{dist}(\mathbf{x}, \mathcal{B})$ for any $\mathbf{x} \in \mathcal{X}$.

Let's establish the converse inequality. Let $\mathbf{x}$ denote an arbitrary point in $\mathcal{D}$. Aiming for a proof by contradiction, assume that $\operatorname{dist}(\mathbf{x}, \mathcal{E}) < \operatorname{dist}(\mathbf{x}, \mathcal{B})$. We can proceed by separating two cases:

- *Case 1:* $\operatorname{dist}(\mathbf{x}, \mathcal{E}) = 0$. This implies that $\mathbf{x} \in \mathcal{E}$ since the set $\mathcal{E}$ is closed as a finite union of closed sets $\partial\mathcal{X}_{i,j}$. Moreover, as $\mathbf{x}$ belongs to $\mathcal{D}$, in particular $\mathbf{x}$ belongs to $\mathcal{S}$. It is sufficient to apply (11) and obtain $\mathbf{x} \in \mathcal{E} \cap \mathcal{S} \subseteq \mathcal{B}$ which is inconsistent with $\operatorname{dist}(\mathbf{x}, \mathcal{B}) > 0$.

- *Case 2:* $r \stackrel{\text{def}}{=} \operatorname{dist}(\mathbf{x}, \mathcal{E}) > 0$. The point $\mathbf{x}$ verifies $\operatorname{dist}(\mathbf{x}, \mathcal{B}) \leq \operatorname{dist}(\mathbf{x}, C\mathcal{B})$ as $\mathbf{x} \in \mathcal{D}$. Let us define $r \stackrel{\text{def}}{=} \operatorname{dist}(\mathbf{x}, \mathcal{E}) > 0$ and $\varepsilon \stackrel{\text{def}}{=} \operatorname{dist}(\mathbf{x}, C\mathcal{B}) - \operatorname{dist}(\mathbf{x}, \mathcal{E}) \geq \operatorname{dist}(\mathbf{x}, \mathcal{B}) - \operatorname{dist}(\mathbf{x}, \mathcal{E}) > 0$ by assumption. Since $\mathbf{x}$ belongs to $\mathcal{S}$, there exists $i_0 \in \{0, 1\}$ such that $\mathbf{x} \in \mathcal{S}_{i_0}$. Because $\operatorname{dist}(\mathbf{x}, \mathcal{E}) = r$, there exists a point $\mathbf{z} \in \mathcal{E}$ such that $r \leq \|\mathbf{x} - \mathbf{z}\|_2 = r + \varepsilon/2$.

  – *Case 2.a:* $\mathbf{z} \in \mathcal{S}_{i_0}$. By assumption (11), the contradiction comes quickly since now

  $$\mathbf{z} \in \mathcal{S}_{i_0} \cap \mathcal{E} \subseteq \mathcal{S} \cap \mathcal{E} \subseteq \mathcal{B} \subseteq C\mathcal{B},$$

  and this implies the contradictive inequality

  $$r + \varepsilon = \operatorname{dist}(\mathbf{x}, C\mathcal{B}) \leq \|\mathbf{x} - \mathbf{z}\|_2 \leq r + \varepsilon/2.$$

  – *Case 2.b:* $\mathbf{z} \notin \mathcal{S}_{i_0}$. In that case, we may define the point $\mathbf{y}$ on the line $[\mathbf{x}, \mathbf{z}]$ which is the nearest from the point $\mathbf{x}$ and also in $\partial\mathcal{S}_{i_0}$. Since $\mathbf{y} \in \partial\mathcal{S}_{i_0} \subseteq C\mathcal{B}$, it implies a contradiction as intended:

  $$\begin{aligned} r + \varepsilon = d(\mathbf{x}, C\mathcal{B}) &\leq \|\mathbf{x}, P(s)\|_2 \\ &= s\|\mathbf{x} - \mathbf{z}\|_2 \leq \|\mathbf{x} - \mathbf{z}\|_2 = r + \varepsilon/2. \end{aligned}$$

  The point $\mathbf{y}$ is defined as $P(s)$ where the map $P : t \mapsto t\mathbf{z} + (1 - t)\mathbf{x}$ assigns to each scalar $t \in [0, 1]$ a point $P(t)$ of the line and set

  $$s \stackrel{\text{def}}{=} \inf_{P(t)\in\mathcal{S}_{i_0}} t.$$

  Since $P(0) = \mathbf{x} \in \mathcal{S}_{i_0}$ and $P(1) = \mathbf{z} \notin \mathcal{S}_{i_0}$, the scalar s is well defined. The remaining task is to show that $P(s) \in \partial\mathcal{S}_{i_0}$. The argument works by construction and with a topological argument. Indeed, by definition of the infimum, there exists a sequence $0 < \eta_n \to 0$ such that $P(s + \eta_n) \notin \mathcal{S}_{i_0}$, thus $P(s) \notin \mathring{\mathcal{S}}_{i_0}$. Also by definition, for all $0 \leq \eta < s$, $P(\eta) \in \mathcal{S}_{i_0}$, thus $P(s) \in \bar{\mathcal{S}_{i_0}}$.

In both cases, the assumption $\operatorname{dist}(\mathbf{x}, \mathcal{E}) < \operatorname{dist}(\mathbf{x}, \mathcal{B})$ leads to a contradiction. We deduce that $\operatorname{dist}(\mathbf{x}, \mathcal{E}) \geq \operatorname{dist}(\mathbf{x}, \mathcal{B})$.

The second inequality in Theorem 1 is a consequence of the property (10). Indeed, this property implies that we can separate the strokes $\mathcal{S}_i$ into connected components $\mathcal{S}_{i,j}$. These are subsets of the connected components $\mathcal{X}_{i,j'}$ for some j' depending on j. The inequality is then just a consequence of Proposition 1.

6 Data augmentation

Data augmentation is one of the most expensive parts of the code. In fact, it recalculates the flows and distance

maps on the fly at each iteration. In addition, for learning to converge faster, it is important to ensure that a patch always contains labels, even with partial annotations. In the original implementation, this was done recursively, i.e. by randomly selecting patches until they contained data. This method is inefficient for sparsely annotated images. Instead, we choose to randomly select centroids of patches among annotated pixels, so that the randomly selected patches are never empty. This avoids unnecessary recursive computation of the data augmentation function.

Furthermore, since data augmentation and flow calculation are not commutative, it was not possible to calculate the flows once and then apply the data augmentation. Nevertheless, it would be possible to calculate a deterministic number of data augmentations on the images in the dataset in advance, calculate the flows and then simply pick up the data using a data loader. We will explore this feature in forthcoming versions.

References

[1] Matthias Arzt, Joran Deschamps, Christopher Schmied, Tobias Pietzsch, Deborah Schmidt, Pavel Tomancak, Robert Haase, and Florian Jug. Labkit: labeling and segmentation toolkit for big image data. *Frontiers in computer science*, 4:10, 2022.

[2] 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*, 16(12):1226–1232, 2019.

[3] Marion Bourdens, Yannick Jeanson, Marion Taurand, Noémie Juin, Audrey Carrière, Franck Clément, Louis Casteilla, Anne-Laure Bulteau, and Valérie Planat-Bénard. Short exposure to cold atmospheric plasma induces senescence in human skin fibroblasts and adipose mesenchymal stromal cells. *Scientific reports*, 9(1):8671, 2019.

[4] Samuel Budd, Emma C Robinson, and Bernhard Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis*, 71:102062, 2021.

[5] 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*, 16(12):1247–1253, 2019.

[6] Chi-Li Chiu, Nathan Clack, et al. napari: a python multi-dimensional image viewer platform for the research community. *Microscopy and Microanalysis*, 28(S1):1576–1577, 2022.

[7] Kevin J Cutler, Carsen Stringer, Teresa W Lo, Luca Rappez, Nicholas Stroustrup, S Brook Peterson, Paul A Wiggins, and Joseph D Mougous. Omnipose: a high-precision morphology-independent solution for bacterial cell segmentation. *Nature methods*, 19(11):1438–1448, 2022.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[9] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[10] Marie-Noëlle Labour, Mathieu Riffault, Søren T Christensen, and David A Hoey. Tgfβ1–induced recruitment of human bone mesenchymal stem cells is mediated by the primary cilium in a smad3-dependent manner. *Scientific reports*, 6(1):35542, 2016.

[11] David Legland, Ignacio Arganda-Carreras, and Philippe Andrey. Morpholibj: integrated library and plugins for mathematical morphology with imagej. *Bioinformatics*, 32(22):3532–3534, 2016.

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

[13] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han.

On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[14] Peter Naylor, Marick Laé, Fabien Reyal, and Thomas Walter. Segmentation of nuclei in histopathology images by deep regression of the distance map. *IEEE transactions on medical imaging*, 38(2):448–459, 2018.

[15] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[16] Marius Pachitariu and Carsen Stringer. Cellpose 2.0: how to train your own model. *Nature methods*, 19(12):1634–1641, 2022.

[17] Ryan Peters. Torchvf: Vector fields for instance segmentation. 2022.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[19] Neil S Sadick, Andrew S Dorizas, Nils Krueger, and Amer H Nassar. The facial adipose system: its role in facial aging and approaches to volume restoration. *Dermatologic Surgery*, 41:S333–S339, 2015.

[20] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. Cell detection with star-convex polygons. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11*, pages 265–273. Springer, 2018.

[21] Jean Serra and Pierre Soille. *Mathematical morphology and its applications to image processing*, volume 2. Springer Science & Business Media, 2012.

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

[23] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(06):583–598, 1991.

[24] Lillian Zhu, Manohary Rajendram, and Kerwyn Casey Huang. Effects of fixation on bacterial cellular dimensions and integrity. *iScience*, 24(4):102348, 2021.