SOFTWARE TOOL ARTICLE

Interactive biomedical segmentation tool powered by deep learning and ImJoy [version 1; peer review: 1 approved, 1 approved with reservations]

Wei Ouyang , Trang Le, Hao Xu , Emma Lundberg

Science for Life Laboratory, School of Engineering Sciences in Chemistry, Biotechnology and Health, KTH - Royal Institute of Technology, Stockholm, Sweden

V1 First published: 24 Feb 2021, 10:142
<https://doi.org/10.12688/f1000research.50798.1>

Latest published: 24 Feb 2021, 10:142
<https://doi.org/10.12688/f1000research.50798.1>

Abstract

Deep learning-based methods play an increasingly important role in bioimage analysis. User-friendly tools are crucial for increasing the adoption of deep learning models and efforts have been made to support them in existing image analysis platforms. Due to hardware and software complexities, many of them have been struggling to support re-training and fine-tuning of models which is essential to avoid overfitting and hallucination issues when working with limited training data. Meanwhile, interactive machine learning provides an efficient way to train models on limited training data. It works by gradually adding new annotations by correcting the model predictions while the model is training in the background. In this work, we developed an ImJoy plugin for interactive training and an annotation tool for image segmentation. With a small example dataset obtained from the Human Protein Atlas, we demonstrate that CellPose-based segmentation models can be trained interactively from scratch within 10-40 minutes, which is at least 6x faster than the conventional annotation workflow and less labor intensive. We envision that the developed tool can make deep learning segmentation methods incrementally adoptable for new users and be used in a wide range of applications for biomedical image segmentation.

Keywords

Deep Learning, Segmentation, Image Analysis, Interactive Machine Learning, ImJoy

This article is included in the **Artificial Intelligence and Machine Learning** gateway.

Open Peer Review

Approval Status

	1	2
version 1		
24 Feb 2021	view	view

1. **Jean-Yves Tinevez** , Pasteur Institute, Paris, Paris, France
Dmitry Ershov, Pasteur Institute, Paris, Paris, France
2. **Jan Funke** , Howard Hughes Medical Institute, Ashburn, USA

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the **Bioinformatics** gateway.

This article is included in the **NEUBIAS - the Bioimage Analysts Network** gateway.

Corresponding authors: Wei Ouyang (wei.ouyang@scilifelab.se), Emma Lundberg (emma.lundberg@scilifelab.se)

Author roles: **Ouyang W:** Conceptualization, Methodology, Project Administration, Software, Visualization, Writing – Original Draft Preparation; **Le T:** Software, Visualization, Writing – Review & Editing; **Xu H:** Data Curation, Software, Visualization, Writing – Review & Editing; **Lundberg E:** Funding Acquisition, Investigation, Methodology, Project Administration, Resources, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: The authors are funded by the Knut and Alice Wallenberg Foundation (grant no. 2016-0204 and 2018.0172, granted to E. L.), the Swedish Research Council (grant no. 2017–05327, granted to E. L.). This work is also supported by the Collaboration within Life Science Research 2020 grant from the LST Platform at KTH Royal Institute of Technology (granted to W. O.).
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2021 Ouyang W *et al.* This is an open access article distributed under the terms of the **Creative Commons Attribution License**, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Ouyang W, Le T, Xu H and Lundberg E. **Interactive biomedical segmentation tool powered by deep learning and ImJoy [version 1; peer review: 1 approved, 1 approved with reservations]** F1000Research 2021, 10:142
<https://doi.org/10.12688/f1000research.50798.1>

First published: 24 Feb 2021, 10:142 <https://doi.org/10.12688/f1000research.50798.1>

Introduction

Deep learning-based methods have been widely used to analyze biomedical images for common tasks such as segmentation^{1,2}, denoising^{3,4} and classification⁵. Despite their potential, building user-friendly deep learning tools and distributing them to non-experts remains challenging. Some attempts to tackle this challenge, like **ImageJ** plugins^{3,6,7} or web applications⁸⁻¹⁰, have been proposed. Among them, due to practical and technical challenges, most tools (e.g. **ImageJ** plugins) only allow for inference with pre-trained models. While it lowers the barriers for users to test and evaluate advanced tools, pre-trained models can easily fail or generate unexpected artifacts due to mismatch of the data distribution for many applications. Depending on the training data distribution and many other factors, applying pre-trained deep learning models can easily suffer from overfitting or other generalization problems. This makes them vulnerable to subtle changes such as noise patterns generated by different microscopes, or morphological changes from different samples. Therefore, it is important to improve the generalization of the pre-trained models, or to re-train and finetune models with users' own data. Previous work including **nucleAIzer**¹⁰ and **CellPose**¹ for nuclei and cell segmentation have demonstrated that more generalized models can be trained by using a large amount of labelled data covering many possible variations such as different imaging modalities and object types. The resulting pre-trained models show promising generalization capability and robustness when applied on unseen data without retraining. However, the fact that this requires a much larger and richer training dataset prevents its use in cases where such a dataset is too expensive to obtain. For most cases, it is thus required to re-train or fine-tune models with a user's own data.

In a typical workflow for training a deep learning model, the first step is to annotate the objects (e.g. cells) in the image manually and obtain the corresponding masks. Typically, this step is labor intensive and time consuming. The second step is to use the labelled images to train a deep learning model (e.g. a U-Net). After training, one can use the trained model to process new images. Despite some metrics such as the Jaccard similarity coefficient which can be used to monitor the training, a common issue is that it is hard to predict how much annotation is required for training the model. As a result, users often need to iterate several rounds between annotation and training.

Meanwhile, interactive machine learning tools such as **ilastik**¹¹ can be used to interactively annotate data and train models by combining the two steps in a more efficient way. It works by first annotating a small amount of data to allow the model to start training in the background. While the model is training, users can work on the annotation and add newly annotated data to the training dataset. In a later stage, instead of annotating all the objects in the image, users can use the model to obtain predicted objects and decide to accept or re-annotate them. This greatly improves the efficiency of both annotations and model training, which not only reduces the overall amount of work required but makes the new annotations more

targeted to the weaker part of the model. By coupling the processing of active learning¹², the total amount of training data can be further reduced. Besides that, the whole process is beneficial for users in helping them to understand the failure modes of the model.

Despite these advantages, many existing interactive machine learning tools are built for classic machine learning methods such as random forest. While ongoing efforts have been made to train deep learning models interactively, building tools that are accessible for non-experts remains challenging due to the increased complexity for annotating data and training deep learning models from an interactive graphical interface. Nevertheless, there have been many existing tools such as **Jupyter notebook** and many libraries in the Web and Python ecosystem that can be used for building interactive training processes. To combine them and make it even easier to work with, we developed **ImJoy**⁸ which is a web application for building interactive and scalable data analysis tools (**ImJoy**, RRID: SCR_020935).

In this work, we demonstrate a tool we built with **ImJoy** for interactive deep learning-based image segmentation. From a web based graphical user interface (GUI), it allows interactive annotations of images and training powerful deep learning models including basic U-Net and **CellPose**. While the GUI can run in a desktop web browser, a mobile phone, or a tablet with touch screen, the computation server part can run in a local workstation or remote server. In addition, the GUI can be embedded in **Jupyter** or **Google Colab** notebooks to allow easy customization and further development for the developers. It also works independently as an **ImJoy** plugin which can be easily shared with a hyperlink to the end users.

Methods

Implementation

The tool is implemented as two **ImJoy** plugins¹³: a model trainer plugin and an image viewer plugin. The trainer plugin is mainly responsible for performing model training and the viewer plugin is built on top of **Kaibu** and used for visualizing and annotating the images. There are additional panels in the GUI for controlling and monitoring the trainer. The model trainer plugin requires **Python** (version 3.7+) along with other python modules, the image viewer plugin runs in a web browser which requires **Chrome** (version 80+) or **Firefox** (version 73+). The two plugins can run directly in **ImJoy** using **Jupyter** notebooks with **ImJoy Jupyter** extension installed, or in a **Google Colab** notebook.

As shown in **Figure 1**, the image viewer interface consists of 1) a set of tools for drawing or editing polygons to mark the objects in the image; 2) the image along with the annotation markups shown as different layers; and 3) control panels for listing the samples, controlling the trainer and monitoring the training loss. The viewer can run independently in modern web browsers on a desktop computer or tablets. Meanwhile, the trainer plugin is written in **Python** and can run through any **Jupyter** notebook server locally or remotely. Depending on

Figure 1. An overview of the interactive deep learning tool for biomedical image segmentation.

the models, we use either **Tensorflow** or **Pytorch** as the base framework. If available, graphical processing units (GPUs) can be used for acceleration. The two plugins are connected via the remote procedure calls (RPC) provided by ImJoy such that the plugins can call each other's functions and pass data between them in a transparent manner. For example, the viewer plugin can call the predict function in the trainer plugin to infer the labels and the trainer plugin can call a function in the viewer to obtain the user corrected labels. For interactive image annotation, a powerful combination is to use a touchscreen device (e.g. tablet or mobile phone) with a pen to draw the markups, and in the meantime, train the model with a remote Jupyter notebook server with GPU.

In addition to the ImJoy interface, with our newly developed Jupyter notebook and Google Colab extension for ImJoy, this interactive annotation tool can also be embedded directly in a notebook interface. This allows developers to quickly customize the trainer plugin in a notebook environment, debug and interact with the trainer through Python code. By running the tool on **Binder**¹⁴ or Google Colab with free computing resources (including GPU access), it allows users and developers to share and reproduce interactive workflows without setting up a local computational environment.

Operation

In order to use the tool, the user needs to prepare a set of unlabeled images and organize them into folders as required by the trainer. Then the user should open the plugin with ImJoy (version 0.11.29+) in a web browser such as Chrome (version 80+) and Firefox (version 73+) to see the image annotation interface. To run the trainer plugin, the user needs to connect to a local or remote Jupyter notebook server (**Miniconda 3** or **Anaconda 3** is recommended for installing the Jupyter notebook server). GPU hardware is recommended but not mandatory. Alternatively, the user can also use the annotation tool in a Google Colab notebook with GPU runtime type selected, and this is recommended for evaluation without any local installation.

To train a model from scratch, the user can start by loading an unlabeled image into the viewer and start the initial annotation process (Figure 2a). With the markup tool, the user can then draw polygons to outline each object (e.g. cell) in the image. Once done, these polygons will be sent to the trainer plugin and be saved into a text-based format named **GeoJSON**. Importantly, the image along with the GeoJSON annotation will be moved and added into a training sample pool. After annotating a few images, the user can start training

Figure 2. Interactive annotation and training workflow and training progression. (a) Interactive annotation and training workflow. Starting with a set of unlabelled images, annotations are gradually made and moved to the training sample pool for training the model. After some iterations, instead of annotating images from scratch, the model can be used to make predictions such that the user can quickly correct the labels. **(b)** Segmentation results showing the progression of the interactive training. Specifically, a CellPose model being trained from scratch with 4 images at time 0, then we gradually add new samples to the training sample pool (20s/image). We test the model with a fixed image which is not seen by the network.

the model. In another thread, a training loop will be started and for each iteration, the trainer will randomly take a batch of samples (e.g. 1–3 images) from the training sample pool, and train the model for one iteration and repeat the process until the user stops the training loop. After being trained for some iterations (e.g. in 2–5 minutes with a GPU from scratch), the user can start to use the model to perform predictions. From then on, instead of annotating the entire image, the user can send the image to the trainer to obtain the predicted labels, fix the wrong labels and keep the correct ones. Since the label correction process is typically much faster than annotating from scratch, the entire workflow is accelerated. The curated labels will be added to the training sample pool and used for training. Video 1¹⁵ is a screen recording for using our tool to

annotate images and train a CellPose segmentation model from scratch in Google Colab.

The trainer supports different types of segmentation models including U-Nets with various types of encoder variations¹⁶ and the CellPose¹ model. It is worth noting that when a suitable pretrained model is used as a start, the user can also skip the initial annotation process and start to correct predicted labels.

Use cases

The tool can be generally applied for annotating images and training deep learning models for segmentation. It is easy

to customize and extend to support different types of data organization, image formats and model architectures.

For demonstration purposes, we provide a small example dataset¹⁷ from the [Human Protein Atlas](#)¹⁸ with 163 samples in total. Each sample is a 4-channel (microtubules, endoplasmic reticulum (ER), nuclei and protein of interest) cell image with manually annotated cell mask in GeoJSON format. The samples cover ~20 different cell-line types with variations in morphology.

For the model training part, we used a slightly modified CellPose model and training process. Specifically, we switched off the style connection in the U-Net used by CellPose and used Adam¹⁹ as the optimizer. The same rotation and scaling augmentations were used as in the original CellPose training workflow. Only two channels (ER and Nuclei) were used as input to the model. Starting by using four labelled images in the training sample pool, new images and labels were gradually moved to the training sample pool during training. [Figure 2b](#) illustrates the rapid progression of the interactive learning process. Specifically, we started the trainer (at time 0) and gradually added one annotated sample every 20s into the training pool. With the same test image, we checked the predicted labels. As shown in the figure, the result rapidly improved in less than a minute. In order to better understand the learning process, we initialized the model with random weights (i.e. no pretrained model is loaded). Compared to a conventional annotation workflow, the interactive tool can accelerate the annotation by roughly 6 times (it takes ~2 minutes to manually annotate the image from scratch, and ~20s to correct labels with our tool).

Conclusions

We demonstrated an interactive annotation and training tool that is capable of accelerating the annotation process for image segmentation. It works by running a training loop in the background while adding new annotations. The feedback loop improves the annotation efficiency and allows more flexibility for the user to control the training and decide on whether new annotation is needed. With the example dataset, we demonstrated improved efficiency for training and annotation. This type of incremental process is more user-friendly for people who want to adopt deep learning-based methods.

We foresee that the developed model can be further improved to increase its efficiency, reproducibility and areas of applicability. A more advanced version could for example support 3D annotation and segmentation, allow recording of the annotation and training activities, or use more advanced sampling schemes when selecting which images the user should annotate (also known as active learning).

Data availability

Example dataset

Zenodo: HPA Cell Image Segmentation Dataset. <https://doi.org/10.5281/zenodo.4430892>¹⁷

This project contains the following underlying data:

- hpa_cell_segmentation_dataset_v2_512x512_4train_159test.zip

Videos

Video 1: Interactive biomedical segmentation tool powered by deep learning and Imjoy.

1 video

<https://doi.org/10.6084/m9.figshare.13721410.v1>¹⁵

Software availability

Sample dataset, source codes and tutorial are available from: <https://github.com/imjoy-team/imjoy-interactive-segmentation>

Archived source code at time of publication: <https://doi.org/10.5281/zenodo.4461080>¹³

Data are available under the terms of the [Creative Commons Attribution 4.0 International license](#) (CC-BY 4.0).

Acknowledgements

We thank Christian Gnann for testing and evaluating the performance of the interactive annotation tool. The example dataset is annotated by Jay Kaimal and Peter Thul.

This publication was supported by COST Action NEUBIAS (CA15124), funded by COST (European Cooperation in Science and Technology).

References

1. Stringer C, Wang T, Michaelos M, et al.: **Cellpose: a generalist algorithm for cellular segmentation**. *Nat Methods*. 2021; **18**(1): 100–106.
[PubMed Abstract](#) | [Publisher Full Text](#)
2. Falk T, Mai D, Bensch R, et al.: **U-Net: deep learning for cell counting, detection, and morphometry**. *Nat Methods*. 2019; **16**(1): 67–70.
[PubMed Abstract](#) | [Publisher Full Text](#)
3. Weigert M, Schmidt U, Bothe T, et al.: **Content-aware image restoration: pushing the limits of fluorescence microscopy**. *Nat Methods*. 2018; **15**(12): 1090–1097.
[PubMed Abstract](#) | [Publisher Full Text](#)
4. Ouyang W, Aristov A, Lelek M, et al.: **Deep learning massively accelerates super-resolution localization microscopy**. *Nat Biotechnol*. 2018; **36**(5): 460–468.
[PubMed Abstract](#) | [Publisher Full Text](#)

5. Ouyang W, Winsnes CF, Hjelmare M, *et al.*: **Analysis of the Human Protein Atlas Image Classification competition**. *Nat Methods*. 2019; **16**(12): 1254–1261.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
6. Gómez-de-Mariscal E, García-López-de-Haro C, Donati L, *et al.*: **DeepImageJ: A user-friendly plugin to run deep learning models in ImageJ**. *bioRxiv*. 2019; 799270.
[Publisher Full Text](#)
7. Krull A, Buchholz TO, Jug F: **Noise2Void - Learning Denoising from Single Noisy Images**. *ArXiv181110980 Cs*. 2019.
[Reference Source](#)
8. Ouyang W, Mueller F, Hjelmare M, *et al.*: **ImJoy: an open-source computational platform for the deep learning era**. *Nat Methods*. 2019; **16**(12): 1199–1200.
[PubMed Abstract](#) | [Publisher Full Text](#)
9. Bannon D, Moen E, Schwartz M, *et al.*: **DeepCell Kiosk: scaling deep learning-enabled cellular image analysis with Kubernetes**. *Nat Methods*. 2021; **18**(1): 43–45.
[PubMed Abstract](#) | [Publisher Full Text](#)
10. Hollandi R, Szkalitsity A, Toth T, *et al.*: **nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer**. *Cell Syst*. 2020; **10**(5): 453–458.e6.
[Publisher Full Text](#)
11. Berg S, Kutra D, Kroeger T, *et al.*: **ilastik: interactive machine learning for (bio)image analysis**. *Nat Methods*. 2019; **16**(12): 1226–1232.
[PubMed Abstract](#) | [Publisher Full Text](#)
12. Sener O, Savarese S: **Active Learning for Convolutional Neural Networks: A Core-Set Approach**. *ArXiv170800489 Cs Stat*. 2018.
[Reference Source](#)
13. Ouyang W, Xu H; trangle1302: **CellProfiling/imjoy-interactive-segmentation: v0.1.0 (Version v0.1.0)**. *Zenodo*. 2021.
<http://www.doi.org/10.5281/zenodo.4461081>
14. Jupyter P, Bussonnier M, Forde J, *et al.*: **Binder 2.0 - Reproducible, interactive, sharable environments for science at scale**. *Proc 17th Python Sci Conf*. 2018; 113–120.
[Publisher Full Text](#)
15. Ouyang W, Le T, Xu H, *et al.*: **Interactive biomedical segmentation tool powered by deep learning and ImJoy**. *f1000research.com*. Media. 2021.
<https://doi.org/10.6084/m9.figshare.13721410.v1>
16. Yakubovskiy P: **Segmentation Models**. GitHub. 2019.
[Reference Source](#)
17. Kaimal J, Thul P, Xu H, *et al.*: **HPA Cell Image Segmentation Dataset (Version v2)**. *Zenodo*. [Data set], 2021.
<http://www.doi.org/10.5281/zenodo.4430901>
18. Thul PJ, Åkesson L, Wiking M, *et al.*: **A subcellular map of the human proteome**. *Science*. 2017; **356**(6340): eaal3321.
[PubMed Abstract](#) | [Publisher Full Text](#)
19. Kingma DP, Ba J: **Adam: A Method for Stochastic Optimization**. *ArXiv1412.6980 Cs*. 2017.
[Reference Source](#)

Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 20 April 2021

<https://doi.org/10.5256/f1000research.53884.r80102>

© 2021 Funke J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Jan Funke

Howard Hughes Medical Institute, Ashburn, VA, USA

This paper introduces software infrastructure for an important up and coming field: the interactive annotation of biomedical images. The authors convincingly introduce the challenges and opportunities of this field, e.g., the difficulty of setting up computational infrastructure for contemporary machine learning and the potential gains of manual interference during the training process. The presented solution consists of a browser-based tool for image annotation, training, prediction, and correction. Crucially, the tool supports several computational backends for the machine learning part, which can easily be selected without requiring deep expertise in distributed computing or machine learning.

As it is, the tool should be seen as an important stepping stone towards a truly interactive training paradigm for experimentalists. Further efforts will be needed to lift this contribution from a proof of concept to a tool that can widely be used by experimentalists and easily extended by machine learning researchers.

The manuscript itself is well written and provides a high-level introduction of the tool, its underlying infrastructure, and design decisions. Nevertheless, it remains somewhat unclear who the target audience of this article is. For machine learning researchers in the life sciences, the manuscript would be more valuable if it would describe in detail how the infrastructure can be extended and modified to fit different algorithms. For experimentalists, a more detailed explanation of how to apply this tool would be beneficial.

Depending on who this manuscript is targeting, I would suggest to answer the following questions.

For machine learning researchers:

- What are the components that make the infrastructure "easy to customize and extend"? How do they interact with each other?
- Are there any constraints on ML models that can be used?

- What is the communication protocol between the front- and backend? Can different annotation tools be used if they implement the protocol?
- Are prediction (by an ML model) and post-processing (for point and polygon detection) coupled? What is the intended way to implement custom post-processors, e.g., for tracking of cells over video frames?
- What motivated the choice for GeoJSON for annotation? Does this choice preempt training on 3D data?
- What are the implicit assumptions about the data? (2D, in-memory storage of training samples).
- What is the standard that describes trained models for sharing? What needs to be done to convert an existing model to be used by this framework?

For experimentalists:

- What kind of use cases does the tool support? What are the limitations of the current tool? (E.g., can it be used on 3D data, on movies, on very large datasets, for semantic segmentation, for denoising, etc.).
- What kind of augmentations are used by the training algorithm? Are they configurable?
- How exactly does one ingest custom datasets and train from scratch? What is the meaning of some of the parameters ("channels", "style_on", "default_diameter") and how should they be picked for a new dataset?
- Once trained, how does one obtain a segmentation for a set of images? "Save Annotation" seems to produce a GeoJSON next to the images in the "data" directory, but there is no notification in the frontend. How would one turn this into a segmentation for further analysis?

Other remarks:

- What is the difference between "Save Annotation" and "Send for Training"?
- How to accept a prediction as a true positive to be included in training?
- "Send for Training": Where does the sample go? It does not seem to appear in the "train" folder. Related, how does one remove a sample from the training pool?

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

No

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Machine learning, computer vision, bioimage analysis, computational neuroscience

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 08 March 2021

<https://doi.org/10.5256/f1000research.53884.r80101>

© 2021 Tinevez J et al. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Jean-Yves Tinevez

Image Analysis Hub, Pasteur Institute, Paris, Paris, France

Dmitry Ershov

Image Analysis Hub, Pasteur Institute, Paris, Paris, France

Main comments.

Ouyang *et al.* present in this article a new module of the software ImJoy, focused on offering interactive Deep-Learning training for segmentation of microscopy images. The goal of this work is to offer biologists a way to iteratively train a Deep-Learning model by manually annotating images. Once an object has been annotated, the training phase runs immediately and displays the results to the user, who can then correct them, add new annotations or stop the training phase if the results are satisfactory. As noted by the authors, this takes some inspiration from the workflow of software like Ilastik, which proved successful.

The paper is concise and well written. The software works as expected and we only have minor points. We recommend the publication of this article.

Minor points.

1. We are both Python users but I seldom use it. I tried to install the software on a Windows machine. Despite getting very clear installation instructions, I failed and the experience was very frustrating. A core library (pytorch) could not be installed. I attach my report of errors at the bottom of this review in case it helps.

I fully understand that the unavailability of a library is not the responsibility of the author, and that a user must be ready to do whatever it takes to get an academic software running. However I am convinced that the impact of the tool would be greater if the installation would become more robust. This could be achieved either by a more thorough testing and installation documentation, or by working around unavailable libraries. This issue is not specific to Imjoy. The frustration against the difficulty of installing Python academic software is shared by many of our colleagues, who are not expert Python developers but still want to benefit from the many software found in this ecosystem for their Research.

2. On mac OS we could run the tutorial notebook and train. But without GPU it is very slow. Maybe the author could insist on the importance of GPU? And document typical training time for both cases?

3. Page 3, 1st column, 1st paragraph: *"Among them, due to practical and technical challenges, ..."* Could you give examples of such challenges?

4. Page 3, 1st column, 1st paragraph: *"However, the fact that this requires a much larger and richer training dataset prevents its use in cases where such a dataset is too expensive to obtain."* Can you detail a bit more this sentence for biological imaging? How many images in common use cases? What "too expensive" means in practical cases?

5. Page 3, 2nd column, 1st paragraph: *"Despite these advantages, many existing interactive machine learning tools are built for classic machine learning methods such as random forest."* This sentence seems imprecise. Could the authors be more precise in its meaning?

6. Page 3, 2nd column, last paragraph: *"1) a set of tools for etc..."*. Perhaps the points 1 to 3 could be shown in Figure 1 for better clarity?

7. Could the software be extended for more flexibility with:

- Adding custom losses (dice, bce, cbce),
- Using custom models,
- Adding a few built-in optimizers,
- Letting the user select the channel for training?

Installing the software.

On Windows 10, from scratch (no Python install present before review).

I failed to install the software and therefore could not evaluate it. Here is the list of steps I took with the error I met, and attempts I made to circumvent them. Following the instructions on <https://github.com/imjoy-team/imjoy-interactive-segmentation> with Anaconda3.

1. The command "pip install -r requirements.txt" cannot complete. I get an error message after

```
getting torch:Collecting torch==1.7.0
Downloading torch-1.7.0-cp37-cp37m-win_amd64.whl (184.0 MB)
|████████████████████████████████████████████████████████████████████████████████| 184.0 MB 1.7 MB/s
ERROR: torch has an invalid wheel, .dist-info directory not found
I tried to downgrade requirements to 1.6.0 but it did not work either:
ERROR: Could not find a version that satisfies the requirement torch==1.6.0
ERROR: No matching distribution found for torch==1.6.0
```

2. The command `python3 -m ipykernel install --user --name imjoy-interactive-ml --display-name "ImJoy Interactive ML"` does not work. It should be `python` instead of `python3`.

3. Running this command generate another error: `python.exe: No module named ipykernel` This is linked to the error #1 above, as `ipykernel` is listed in the requirements file

4. The command `imjoy --jupyter` returns an error '`imjoy`' is not recognized as an internal or external command, operable program or batch file. Again linked to the error #1 above.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: image analysis, microscopy, imaging

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research