BiaPy: A unified framework for versatile bioimage analysis with deep learning

Daniel Franco-Barranco[1,2], Jesús A. Andrés-San Román[3,4], Ivan Hidalgo-Cenalmor[1,5], Lenka Backová[1,6], Aitor González-Marfil[1,2], Clément Caporal[7], Anatole Chessel[7], Pedro Gómez-Gálvez[3,4,8,9], Luis M. Escudero[3,4], Donglai Wei[10], Arrate Muñoz-Barrutia[11,12,✉], Ignacio Arganda-Carreras[1,2,6,13,✉]

[1] Dept. of Computer Science and Artificial Intelligence, University of the Basque Country (UPV/EHU), San Sebastian, Spain
[2] Donostia International Physics Center (DIPC), San Sebastian, Spain
[3] Instituto de Biomedicina de Sevilla (IBiS), Hospital Universitario Virgen del Rocío/CSIC/ Universidad de Sevilla and Dept. de Biología Celular, Facultad de Biología, Universidad de Sevilla, Seville, Spain
[4] Biomedical Network Research Centre on Neurodegenerative Diseases (CIBERNED), Madrid, Spain
[5] Instituto Gulbenkian de Ciência, Oeiras, Portugal
[6] Biofisika Institute (CSIC-UPV/EHU), Leioa, Spain
[7] Laboratoire d'Optique et Biosciences, CNRS, Inserm, Ecole Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
[8] MRC Laboratory of Molecular Biology, Cambridge, UK
[9] Dept. of Physiology, Development and Neuroscience, University of Cambridge, Cambridge, UK
[10] Dept. of Computer Science, Boston College, USA
[11] Dept. de Bioingenieria, Universidad Carlos III de Madrid, Madrid, Spain
[12] Area de Bioingenieria, Instituto de Investigación Sanitaria Gregorio Marañón, Madrid, Spain
[13] IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
✉ Co-corresponding authors

Abstract. BiaPy, a unified open-source bioimage analysis library, offers a comprehensive suite of deep learning-powered workflows. Tailored for users of all levels, BiaPy features an intuitive interface, zero-code notebooks, and Docker integration. With support for 2D and 3D image data, it addresses existing gaps by providing multi-GPU capabilities, memory optimization, and compatibility with large datasets. As a collaborative and accessible solution, BiaPy aims to empower researchers by democratizing the use of sophisticated and efficient bioimage analysis workflows.

Keywords: Bioimage analysis, deep learning, image segmentation, object detection, denoising, single image super-resolution, self-supervised learning, image classification.

Bioimage analysis is a cornerstone of modern life sciences, powering discoveries and insights derived from biological image data. With deep learning emerging as the standard for analyzing microscopy datasets, its application in biomedical contexts [1] has become prevalent. However, the prerequisite for high-level programming skills has often acted as a barrier, limiting accessibility to researchers without a specific computational background [2]. The

rapid evolution of deep learning methods, coupled with the diverse nature of bioimage analysis applications, has created a dynamic landscape, demanding continuous adaptation from researchers who must grapple with an ever-expanding array of frameworks.

Furthermore, researchers often face the challenge of navigating through a multitude of tools when employing various bioimage analysis pipelines, commonly known as *workflows*. These workflows encompass tasks ranging from semantic/instance segmentation to object detection, tracking, image classification, and reconstruction. To democratize these diverse workflows through deep learning, a range of solutions are available [3–15]. The integration of deepImageJ [4] brought pretrained deep learning models into Fiji [3], leveraging a diverse range of models accessible through the Bioimage Model Zoo [16]. Covering a broad spectrum of workflows, the Bioimage Model Zoo facilitates the sharing and reuse of pretrained deep learning models. Complementary to this, ZeroCostDL4Mic [5] provides well-documented Jupyter notebooks reproducing workflows from various authors. ImJoy [6] also simplifies access to this technology through a web interface with plenty of deep learning solutions. QuPath [13] is actively adapting BioImage Model Zoo models to incorporate semantic and instance segmentation workflows. All the software mentioned above, together with Ilastik [9] and Icy [17], are Community Partners of the BioImage Model Zoo.

A number of tools are designed for building pipelines for classification, semantic segmentation and instance segmentation [7–10, 14, 15]. Classification and object detection workflows can also be built in some of the aforementioned tools [8, 9, 13, 14]. Among the solutions discussed, only a few provide a graphical user interface (GUI), with most relying on command-line actions for straightforward use. Even those equipped with a GUI often need command-line interaction to start the program, perpetuating accessibility challenges and restricting the user base. Web-browser-based approaches, exemplified by ZeroCostDL4Mic [5] and ImJoy [6], alleviate the hurdle of software installation at the cost of reduced flexibility. For instance, ZeroCostDL4Mic [5] relies on a third-party platform, i.e., Google Colab, demanding regular updates to accommodate any changes on that platform. Addressing this concern, the same authors recently introduced DL4MicEverywhere [18], a platform that uses Docker containers to encapsulate the ZeroCostDL4Mic notebooks along with their dependencies, ensuring portability and reproducibility across computing environments. Regarding software usability, deepImageJ [4] is limited as yet to making predictions with pretrained models and does not support training with custom datasets. These constraints pose a significant drawback for researchers seeking versatility in adapting tools to their specific experimental setups and datasets.

To fill these gaps in the current bioimage analysis landscape, we introduce BiaPy (`https://biapyx.github.io`), an all-in-one, open-source library designed to support an extensive array of bioimage analysis tasks tailored for multi-channel, multi-dimensional image microscopy data (Fig. 1a). This comprehensive toolkit simplifies the analytical process, catering to the diverse needs of researchers. A key feature of BiaPy's workflows is deep learning, leveraging both traditional convolutional neural networks (CNNs) and modern Transformer architectures [19]. This combination of established and cutting-edge models provides researchers with enhanced accuracy and adaptability, contributing to the evolving landscape of bioimage analysis. BiaPy is accessible to both computer vision experts seeking to integrate their latest algorithms into a cohesive software solution and life scientists in need of powerful tools to accelerate their research. Unlike other tools that rely on command-line interactions, BiaPy boasts an intuitive

interface (Fig. 1b) and zero-code online notebooks (Fig. 1c), ensuring that researchers, regardless of their technical proficiency, can effortlessly leverage the provided workflows. Moreover, the incorporation of Docker containers streamlines the installation process and facilitates the execution of containerized workflows across diverse computing environments without conflicts with existing software versions or dependencies.

BiaPy distinguishes itself as an end-to-end solution, featuring advanced capabilities essential for both local environments and image facility frameworks. It provides support for multi-GPU setups and compatibility with large files in Zarr/H5 formats, enhancing its scalability for handling substantial datasets and computationally demanding analyses (Fig. 1c). Additionally, the integration of BiaPy with popular deep learning ecosystems, including the BioImage Model Zoo [16], promotes the community's ability to reuse pretrained models and workflows while leveraging the computational benefits of our framework (Fig. 1a). BiaPy adopts a simple yet powerful workflow structure comprising three key elements: data preprocessing, model training (including novel microscopy-specific data augmentation methods) or inference, and data post-processing (Fig. 1c). This structure serves a dual purpose. Firstly, it facilitates the prototyping of novel workflows, empowering users to effortlessly choose pre- and post-processing methods based on their data and specific downstream tasks. Users can interchange various state-of-the-art deep learning models available within the library and its compatible projects (e.g., TorchVision [1] and the Bioimage Model Zoo). Secondly, this design simplifies the work for developers, allowing them to concentrate on specific parts of the workflows rather than requiring them to contribute entire workflows.

Built on Python, with PyTorch as its backend for deep learning, BiaPy reflects a commitment to an accessible and widely adopted computational environment. BiaPy is engineered with memory optimization strategies. Options for automatic mixed precision (for faster training on GPUs) and efficient RAM utilization make it adaptive to varied computational setups, enhancing its applicability across different hardware configurations. For computationally demanding analyses, BiaPy introduces robust support for multi-GPU configurations. Both training and inference tasks can be efficiently distributed across multiple GPUs, mitigating computational bottlenecks. Multi-GPU operations are implemented in chunks to accommodate varying system capabilities, enhancing flexibility and scalability (see Fig. 1c). Each workflow in BiaPy is encapsulated in a single configuration file (Fig. 1c), which users can directly modify in a standard text editor or effortlessly generate through the GUI or zero-code notebooks. Sharing workflows becomes as straightforward as sharing this file, optionally coupled with the model weights file for inference purposes. In BiaPy, a generic workflow can take raw images as input, along with corresponding labels or annotations. Depending on the specified task, the output may manifest as images, predicted annotations, analysis report files, or even a pretrained model.

In its current release, BiaPy provides support for the following workflows in both 2D and 3D image data: instance segmentation, semantic segmentation, object detection, image denoising, single image super-resolution, self-supervised learning, and image classification (see Methods). Our instance segmentation workflows follow a bottom-up approach (Fig. 2a), where objects of interest in the input image are detected, segmented, and classified. The model initially learns binary masks, contours, and, optionally, the distance map of the objects. These representations are then combined to generate seeds for a marker-controlled watershed trans-

[1] https://pypi.org/project/torchvision/0.1.9/

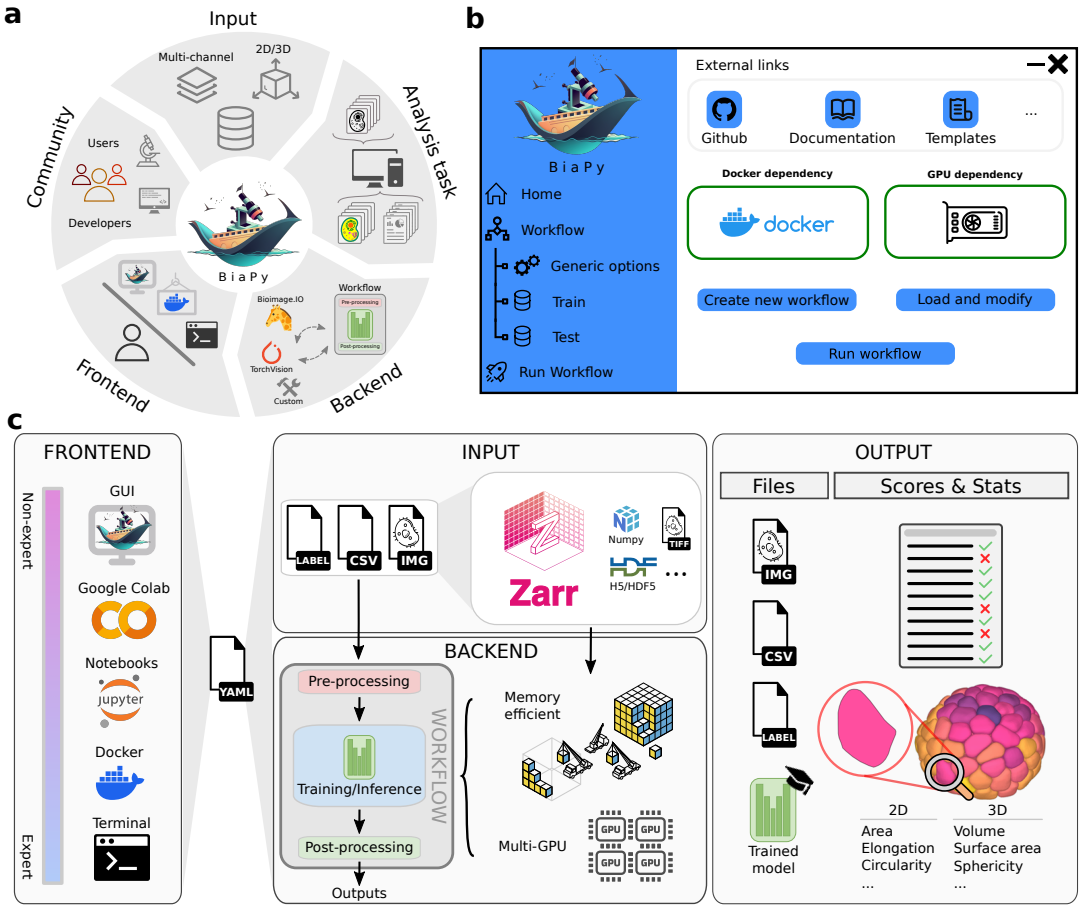

Fig. 1: **BiaPy environment and scope. a.** General overview. By addressing their distinct needs, BiaPy accommodates life science users and computer vision developers. Multi-channel, 2D/3D microscopy datasets serve as inputs for BiaPy workflows. Recurrent bioimage analysis tasks are executed by workflows that can be customized, imported/exported from/to the BioImage Model Zoo [16], or leverage pretrained models from TorchVision. The backend allows for customization, while the frontend caters to a broader user base. **b.** Graphical User Interface (GUI): BiaPy's GUI serves as an intuitive interface for user interaction. **c.** Users have the flexibility to install and run BiaPy through different methods tailored to various experience levels. During execution, a YAML configuration file is created, defining input data and providing instructions for the chosen workflow. BiaPy supports classic image formats (TIFF, Numpy) and modern, memory-efficient formats (H5, Zarr). The workflow involves three main elements: 1) Pre-processing to prepare input data for the workflow. 2) Model training or inference with predictions obtained segment by segment, patch by patch, and montaged with overlapping/padding strategies and supporting multi-GPU settings. 3) After inference, probability maps undergo post-processing to produce the workflow outcome. The setup process includes configuring model specifications. BiaPy outputs in various formats and dimensions (images, tables, text files) and offers evaluation metrics and morphological statistics for comparison with user-provided ground truth images.

form, producing individual instances. In the semantic segmentation workflows (Fig. 2b), every pixel/voxel of the input image is associated with a label leveraging the wide range of models available in BiaPy. For object detection (Fig. 2c), BiaPy adopts a center-of-mass strategy, commonly employed in bioimage analysis, rather than annotation by bounding boxes. The quality of the input images can be enhanced thanks to BiaPy's denoising (Fig. 2d) and super-resolution (Fig. 2e) workflows, where the output images are cleaner or super-resolved versions of the input ones. A distinctive feature of BiaPy is its self-supervised learning workflows (Fig.2f), where models undergo pretraining by solving a pretext task without labels. This approach enables models to learn a representation that can later be transferred to address a downstream task in a labeled, often smaller dataset. Finally, BiaPy's image classification workflows (Fig.2g) involve training any of the available models to label entire input images, assigning them to a predefined set of classes. See Methods for more details about the workflows.

In conclusion, BiaPy emerges as a versatile, portable and accessible solution, offering a comprehensive suite of workflows for bioimage analysis, supporting both 2D and 3D image data. Its unique combination of intuitive GUI, zero-code notebooks, and robust Docker integration ensures accessibility for users with varying technical proficiency. With a commitment to reproducibility and flexibility, BiaPy provides a seamless experience for both computer vision experts and life scientists. By addressing limitations in current tools, such as a lack of comprehensive GUIs, multi-GPU support, and adaptability to large datasets, BiaPy stands as a powerful, end-to-end solution for the evolving landscape of bioimage analysis. As an open-source initiative, BiaPy invites collaboration and contributions from the scientific community, aiming to empower researchers and accelerate advancements in bioimage analysis.

Code availability

The complete source code for the BiaPy ecosystem, encompassing the library's main code, GUI, and associated documentation, is accessible at `https://github.com/BiaPyX`. For comprehensive documentation, tutorials, and examples, please refer to BiaPy's documentation website (`https://biapy.readthedocs.io/en/latest/`).

Acknowledgments

This work is partially supported by grant GIU23/022 (to I.A-C.) funded by the University of the Basque Country (UPV/EHU), grants PID2021-126701OB-I00 (to I.A-C.) and PID2019-109820RB-I00 (to A.M-B.), funded by the Ministerio de Ciencia, Innovación y Universidades, AEI, MCIN/AEI/10.13039/501100011033, and by "ERDF A way of making Europe" (to I.A-C.). P.G-G. has been funded by Margarita Salas Fellowship – NextGenerationEU. J.A.A-S. work has been funded by the Junta de Andalucía (Consejería de economía, conocimiento, empresas y Universidad) grant PY18-631 co-funded by FEDER funds. L.M.E. also wants to thank PIE-202120E047-Conexiones-Life network for networking and input. I.H-C. and A.M-B. received funding from the European Union through the Horizon Europe program (AI4LIFE project with grant agreement 101057970-AI4LIFE). Funded by the European Union. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be

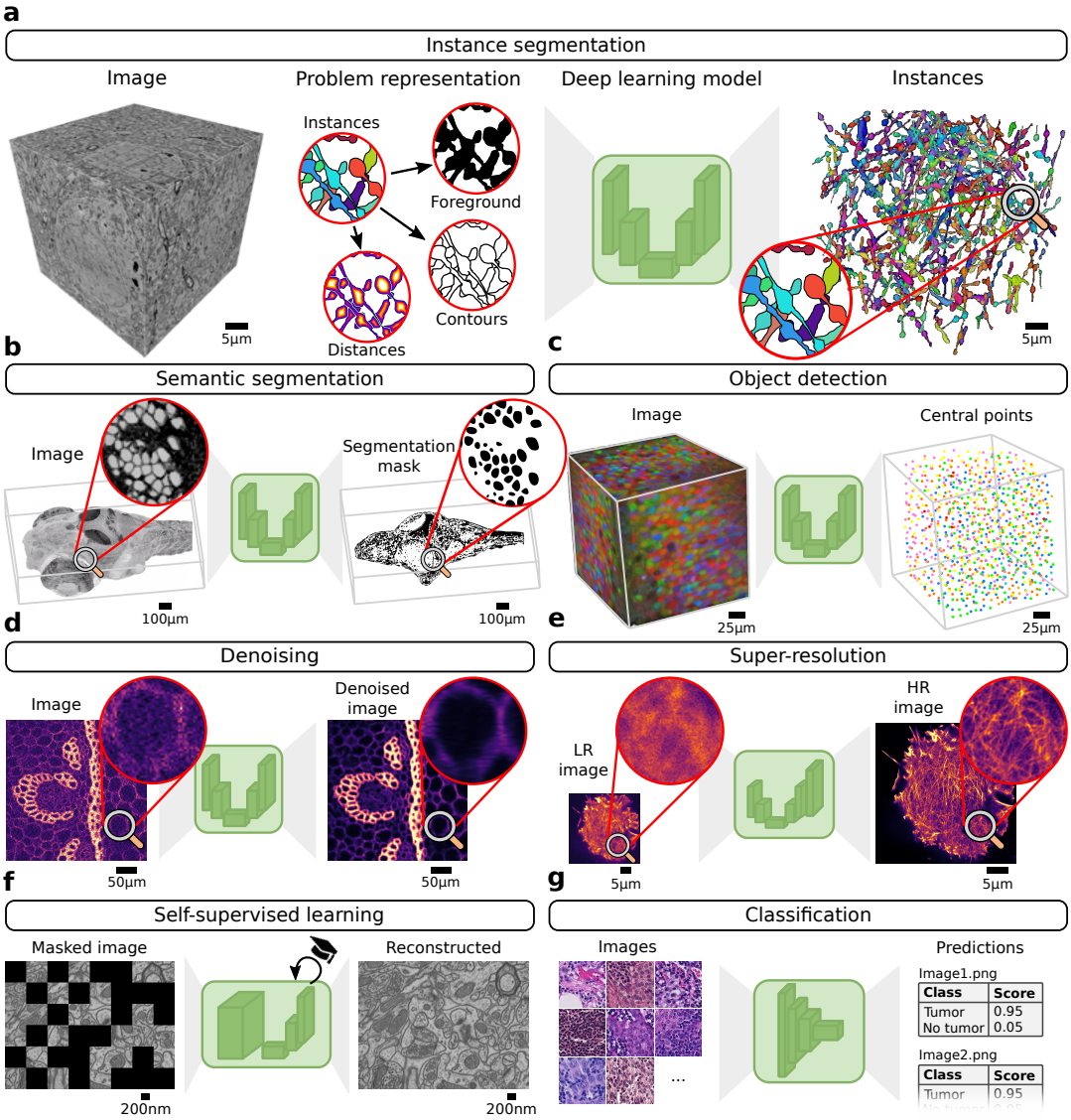

Fig. 2: **Overview of BiaPy workflows.** BiaPy supports the following workflows for 2D/3D grayscale, RGB and multi-channel images: **a.** Instance segmentation involves detecting, segmenting, and classifying individual objects using a bottom-up approach by learning the binary masks, contours, and (optionally) the distance map of the objects of interest. **b.** Semantic segmentation associates a label to every pixel or voxel of the input image. **c.** Object detection localizes objects in the input image, not requiring a pixel-level class, by extracting individual points at their center of mass. **d.** Denoising removes noise from the input image. **e.** Super-resolution reconstructs high-resolution (HR) images from low-resolution (LR) ones. **f.** Self-supervised learning pretrains a backbone model on a pretext task without labels, enabling transfer to downstream tasks in labeled datasets. **g.** Image classification labels full input images as belonging to a predefined set of classes.

held responsible for them. I.H-C. also acknowledges the support of the Gulbenkian Foundation (Fundação Calouste Gulbenkian).

Author contributions

Conceptualization: D.F-B., A.C., L.M.E, D.W., A.M-B. and I.A-C. Software: D.F-B., J.A.A-S., I.H-C., L.B., A.G-M., P.G-G., C.C. and I.A-C. Validation: D.F-B., I.H-C., L.B., A.G-M., P.G-G., C.C. and I.A-C. Writing—initial outline: D.F-B., A.M-B. and I.A-C. Writing—original draft: D.F-B., A.M-B. and I.A-C. Writing—review and editing: all authors. Visualization: D.F-B. and I.A-C. Supervision: L.M.E, D.W., A.M-B. and I.A-C. Funding acquisition: A.C., L.M.E, D.W., A.M-B. and I.A-C.

Competing interests

The authors declare that they have no competing interests.

References

[1] A. Esteva et al. "Deep learning-enabled medical computer vision". In: *NPJ Digital Medicine* 4.1 (2021), pp. 1–9.

[2] R. F. Laine et al. "Avoiding a replication crisis in deep-learning-based bioimage analysis". In: *Nature Methods* 18.10 (2021), pp. 1136–1144.

[3] J. Schindelin et al. "Fiji: an open-source platform for biological-image analysis". In: *Nature Methods* 9.7 (2012), pp. 676–682.

[4] E. Gómez-de-Mariscal et al. "DeepImageJ: A user-friendly environment to run deep learning models in ImageJ". In: *Nature Methods* 18.10 (2021), pp. 1192–1195.

[5] L. von Chamier et al. "Democratising deep learning for microscopy with ZeroCostDL4Mic". In: *Nature Communications* 12.1 (2021), pp. 1–18.

[6] W. Ouyang et al. "ImJoy: an open-source computational platform for the deep learning era". In: *Nature Methods* 16.12 (2019), pp. 1199–1200.

[7] C. Stringer et al. "Cellpose: a generalist algorithm for cellular segmentation". In: *Nature Methods* 18.1 (2021), pp. 100–106.

[8] D. R. Stirling et al. "CellProfiler 4: improvements in speed, utility and usability". In: *BMC Bioinformatics* 22.1 (2021), pp. 1–11.

[9] S. Berg et al. "Ilastik: interactive machine learning for (bio) image analysis". In: *Nature Methods* 16.12 (2019), pp. 1226–1232.

[10] A. Wolny et al. "Accurate and versatile 3D segmentation of plant tissues at cellular resolution". In: *Elife* 9 (2020), e57613.

[11] N. Körber. "MIA is an open-source standalone deep learning application for microscopic image analysis". In: *Cell Reports Methods* 3.7 (2023).

[12] N. F. Greenwald et al. "Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning". In: *Nature Biotechnology* 40.4 (2022), pp. 555–565.

[13] P. Bankhead et al. "QuPath: Open source software for digital pathology image analysis". In: *Scientific Reports* 7.1 (2017), pp. 1–7.

[14] D. J. E. Waibel, S. Shetab Boushehri, and C. Marr. "InstantDL: an easy-to-use deep learning pipeline for image segmentation and classification". In: *BMC Bioinformatics* 22 (2021), pp. 1–15.
[15] R. Hollandi et al. "nucleAIzer: a parameter-free deep learning framework for nucleus segmentation using image style transfer". In: *Cell Systems* 10.5 (2020), pp. 453–458.
[16] W. Ouyang et al. "Bioimage model zoo: a community-driven resource for accessible deep learning in bioimage analysis". In: *bioRxiv* (2022), pp. 2022–06.
[17] Fabrice De Chaumont et al. "Icy: an open bioimage informatics platform for extended reproducible research". In: *Nature methods* 9.7 (2012), pp. 690–696.
[18] I. Hidalgo-Cenalmor et al. "DL4MicEverywhere: Deep learning for microscopy made flexible, shareable, and reproducible". In: *bioRxiv* (2023), pp. 2023–11.
[19] A. Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

Methods

Technical descriptions and software requirements are provided in the following lines. As BiaPy is an ongoing project, we strongly recommend both users and developers to check the documentation at BiaPy's website (`https://biapyx.github.io/`).

Design goals of BiaPy

The design philosophy of BiaPy addresses the multifaceted challenges inherent in the field of bioimage analysis while democratizing access to cutting-edge computational techniques. At its core, BiaPy strives to fulfill several key objectives:

1. **Unified framework:** BiaPy is crafted as a unified platform, integrating both backend and frontend functionalities. This design ensures accessibility and benefits for both novice and expert users.
2. **Versatility in handling image modalities:** BiaPy is engineered to support a wide range of image modalities, including multi-channel, frequently anisotropic, 2D, and 3D images, addressing the diversity of bioimaging data.
3. **Customizable for various output targets/tasks:** BiaPy's architecture is designed to be adaptable to different output targets and tasks, offering customizable solutions for tasks such as semantic segmentation, instance segmentation, object detection, denoising, and super-resolution.
4. **Scalability and optimization:** BiaPy focuses on scalability and optimization, incorporating multi-GPU capabilities and memory optimization strategies to handle computationally demanding bioimage analyses with large datasets.
5. **Community-driven and collaborative approach:** BiaPy is envisioned as a collaborative tool, encouraging contributions and feedback from the research community. This open-source nature fosters continuous improvement and innovation in bioimage analysis.

Through these design goals, BiaPy aims to provide an accessible, portable and flexible tool for the bioimage analysis community, enabling researchers to harness the power of deep

learning and other advanced computational methods in their work. Table 1 provides a detailed comparison of BiaPy's features with those of similar tools currently available in the bioimage analysis ecosystem.

Installation and usage: from novice to expert

For a comprehensive guide to installation, refer to `https://biapy.readthedocs.io/en/latest/get_started/installation.html`. BiaPy offers multiple avenues for installation and usage to accommodate users with varying technical proficiencies:

- **Graphical user interface (GUI)**: BiaPy includes a GUI for users seeking a user-friendly interface. The GUI performs checks for Docker image installation and GPU availability, enabling easy workflow execution.
- **Google Colab integration**: BiaPy provides code-free notebooks in Google Colab, lowering entry barriers and enabling researchers to use BiaPy without local installations.
- **Jupyter notebooks**: Users can access templates and executable Jupyter notebooks locally for each workflow, accompanied by example datasets for better understanding.
- **Docker**: BiaPy is available as a Docker image, ensuring a consistent environment across systems. Two containers support different PyTorch/CUDA versions for broader accessibility.
- **Command line**: For users familiar with conventional workflows, obtaining BiaPy is as straightforward as executing a repository clone command or using standard package installation tools (e.g., *pip*). Subsequent installation of dependencies grants users direct access to BiaPy's functionalities.

Software dependencies and hardware requirements

BiaPy has undergone successful testing on Linux, MacOS, and Windows operating systems. Given the deep-learning core of BiaPy, a machine equipped with a GPU is recommended for optimal training and execution speed. Users accessing BiaPy through its GUI need only Docker, as the GUI operates BiaPy indirectly using Docker. The GUI is provided as a binary file, available for download from BiaPy's documentation (`https://biapy.readthedocs.io/en/latest/get_started/installation.html`) or GitHub (`https://github.com/BiaPyX/BiaPy-GUI`) pages. After downloading, it can be launched with a simple double-click. For other cases, such as Google Colab, BiaPy's dependencies are effortlessly installed using *pip*.

To accommodate a broader user base, BiaPy is distributed through two separate Docker containers: one based on PyTorch 2.1 and another on PyTorch version 1.2.1, corresponding to CUDA versions 11.8 and 10.2, respectively. This approach ensures compatibility with older GPUs that may have outdated drivers while retaining full functionality in BiaPy.

Multi-GPU setting

BiaPy offers training on a multi-GPU setting using PyTorch's distributed data-parallel (DDP) training, a widely adopted paradigm for single-program multiple-data training. Moreover, BiaPy introduces a novel strategy for multi-GPU inference (depicted in Fig. 1c). Unlike the conventional method of distributing all test images across available GPUs for accelerated

processing, BiaPy's approach is tailored for biological microscopy image data, addressing challenges posed by very large images. More specifically, our method addresses the constraints related to memory and disk space. BiaPy enables multi-GPU processing per image by chunking large images into patches with overlap and padding to mitigate artifacts at the edges. Each GPU processes a chunk of the large image, storing the patch in its designated location within an output file, typically in Zarr or H5 format. These file formats facilitate reading and storing data chunks without requiring the entire file to be loaded into memory. Consequently, our approach allows the generation of predictions for large images, overcoming potential memory bottlenecks.

BiaPy workflows

BiaPy is designed to process multi-channel, frequently anisotropic, 2D, and 3D images by means of a variety of workflows. These workflows are enhanced by the integration of models from the pretrained Bioimage Model Zoo [16] and TorchVision models [2], accessible via their official repository, when applicable to the specific task. Additionally, BiaPy supports our own custom deep learning models, details of which are provided for each specific task. The following section outlines the technical specifics and the diverse model support of the existing BiaPy workflows:

Semantic segmentation. Images are processed to assign a class to each pixel or voxel of the input image. For this task, the deep learning models currently available in BiaPy are custom U-Net [20, 21], Residual U-Net [21, 22], ResUNet++ [23], Attention U-Net [21, 24], MultiResUnet [25], Squeeze-and-Excitation (SE) block U-Net [21] and UNETR [26]. Training image patches can be selected based on a probability map generated from the ground truth masks. This approach ensures a higher frequency of patches containing a foreground class is input into the deep learning model. Apart from the full 3D workflow implemented, at test time, a workflow segmenting 2D images can also produce a 3D output if test images constitute a 3D stack. To address potential 3D inconsistencies and enhance the smoothness of the predictions, BiaPy offers two post-processing methods. These include applying median filtering either along the z-axis or concurrently on the x- and y-axes. The semantic segmentation results are evaluated using the intersection over union (IoU) score.

Instance segmentation. The goal of this workflow is to assign a unique identifier, i.e., integer, to each object of interest in the input image. For this task, BiaPy currently supports the following models: U-Net [20, 21], Residual U-Net [21, 22], ResUNet++ [23], Attention U-Net [21, 24], MultiResUnet [25], SE U-Net [21] and UNETR [26]. BiaPy uses a bottom-up approach for instance segmentation, where models are trained to predict intermediate representations of the object masks, such as binary masks, boundaries, central points, and/or distance maps [27, 28]. Then, marker-controlled watershed [29] is used to convert these representations into object instances. In addition to the post-processing methods described for semantic segmentation, predicted instances can be refined using morphological operators or based on a Voronoi tessellation [30]. The quality of the output instances can be measured by a large variety of metrics, including average precision, association, and matching metrics [28].

Object detection. This workflow aims to localize objects within the input image without requiring pixel-level classification. Common strategies include producing bounding boxes

[2] https://pytorch.org/vision/stable/models.html

around objects or pinpointing their center of mass [31], the latter being the approach employed by our tool. Multi-class central points are also supported in this workflow. BiaPy supports several models for this task, such as U-Net [20, 21], Residual U-Net [21, 22], ResUNet++ [23], Attention U-Net [21, 24] and SE U-Net [21]. For training, BiaPy uses additional input CSV files that list the coordinates for the center of mass of each object within each class. Then, those coordinates are used to create a target image with point masks. During inference, the model computes probabilities for each center of mass, which are subsequently processed to identify final detections. BiaPy includes several post-processing methods, such as non-maximum suppression, to remove close points based on a predefined radius. Complete instances can be reconstructed from the predicted centers using marker-controlled watershed [29], offering adaptability to various object shapes. The same filtering techniques applied in the instance segmentation workflow are available here, tailored to the instances' shapes. For evaluation, BiaPy computes precision, recall, and F1 metrics for the final predicted points based on a predetermined tolerance to assess the model's accuracy.

Denoising. This workflow aims to eliminate noise from images. Our library incorporates the Noise2Void framework [32], an unsupervised method that only needs as input the noisy images and tries to reduce the probabilistic noise. Available models for this task include U-Net [20, 21], Residual U-Net [21, 22], ResUNet++ [23], Attention U-Net [21, 24] and SE U-Net [21].

Single image super-resolution. This workflow focuses on reconstructing high-resolution (HR) images from their low-resolution (LR) counterparts. Available models include the Enhanced Deep Residual Network (EDSR) [33], Deep Residual Channel Attention Networks (RCAN) [34], Deep Fourier Channel Attention Network (DFCAN) [35], Wide Activation for Efficient and Accurate Image Super-Resolution (WDSR) [36], U-Net [20, 21], Residual U-Net [21, 22], ResUNet++ [23], Attention U-Net [21, 24], MultiResUnet[25], SE U-Net [21]. For 3D images, only the custom U-Net-like models are supported. For evaluation, the popular peak signal-to-noise ratio (PSNR) metric is calculated between the reconstructed HR images and the available ground truth data.

Self-supervised learning. In this workflow, a backbone model is pretrained by solving a so-called pretext task without the need for labels. This approach allows the model to learn a representation that can later be transferred to solve a downstream task using a labeled, albeit smaller, dataset. In BiaPy, we adopt two pretext tasks: reconstruction and masking. The reconstruction task involves recovering the original image from a degraded version of it [37]. In the masking pretext task, random patches of the input image are masked, and the network is trained to reconstruct the missing pixels or voxels [38]. Available models include the Vision Transformer (ViT) [19], Masked Autoencoder (MAE) [38], EDSR [33], RCAN [34], DFCAN [35], WDSR [36], U-Net [20, 21], Residual U-Net [21, 22], ResUNet++ [23], Attention U-Net [21, 24], MultiResUnet[25], SE U-Net [21] and UNETR [26]. For evaluation purposes, the PSNR metric is calculated, as both pretext tasks aim to recover an image from a distorted input.

Image classification. The aim of this workflow is to assign a specific label to the whole input image. The custom models available for this task include a simple convolutional neural network, EfficientNet [39] and ViT [19]. Popular evaluation metrics such as accuracy, precision, recall, and F1 score are calculated to assess the performance of the models.

Microscopy-tailored data augmentation

BiaPy incorporates a range of data augmentation methods commonly used in the classification of natural images, but customized for 3D and multi-channel microscopy images of various resolutions. These methods encompass Cutout [40], CutBlur [41], CutMix [42], CutNoise (a variation of CutBlur with additional noise), and GridMask [43], among others. Additionally, BiaPy introduces novel data augmentation techniques specifically designed to mimic typical distortions found in microscope image acquisition, such as misalignment or missing sections (e.g., z-slices). BiaPy also integrates with the imgaug library [3], allowing for the implementation of custom augmentation strategies.

Input and output data management

BiaPy provides two distinct approaches to data management for training, validation, and testing datasets. The first approach involves loading data into memory, which accelerates processing but requires considerable memory allocation. In this approach, the validation dataset can be generated by partitioning the training dataset. The alternative approach is to dynamically load data from the disk as needed, which conserves memory but may result in slower processing times.

For all workflows, except for classification, data loaded into memory is cropped to a specific patch size. This cropping can accommodate predefined overlap and padding. In this context, having images stored with uniform size is not necessary. Nevertheless, when data is dynamically loaded from disk, it should be uniform in size to facilitate consistent patch cropping, which is not always the default condition. To address this, BiaPy includes a random cropping feature, allowing users to ensure consistent patch sizes throughout datasets. In the classification workflows, random cropping is automatically implemented whenever the selected patch size does not correspond with the dimensions of the loaded image.

During the inference phase, each test image can be processed in a full-sized setting or by cropping it to a predetermined patch size. In the latter, this cropping process may also incorporate predetermined overlap and padding, similar to the method applied to training and validation datasets. This technique facilitates the reconstruction of the final image while efficiently minimizing memory usage, thereby ensuring scalability, as detailed in Section 1 of the Online Methods. Furthermore, BiaPy employs test-time augmentation by averaging the predictions from multiple orientations of each patch, specifically creating 8 versions in 2D and 16 in 3D, achieved through multiple 90-degree rotations and mirrored versions of the input images.

Use case examples

BiaPy has been employed in diverse projects encompassing various image modalities, including electron microscopy (EM), confocal microscopy, microcomputed tomography (uCT), and fluorescence microscopy. The range of object shapes, image resolution, and image contrast in those examples illustrates BiaPy's adaptability to varied scenarios. The following section briefly describes some representative research projects already published using BiaPy:

[3] `https://github.com/aleju/imgaug`

Mitochondria instance segmentation in large EM volumes ([27, 28]). In a previous work [27], we introduced the MitoEM dataset, a large 3D EM volume including mitochondria of very complex morphology and varying size. This dataset challenged existing instance segmentation methods and motivated the creation of the MitoEM challenge[4] on 3D instance segmentation of mitochondria in EM images. Together with our report of the findings of the challenge [28], we published our own baseline method (U2D-BC) as a BiaPy workflow, which is fully reproducible using the detailed instructions contained in the tutorials section of BiaPy's documentation site.

Modeling of wound healing in *Drosophila* embryos ([44]). This study compiled a dataset of time-lapse sequences showing *Drosophila* embryos as they recover from a laser-made incision. We approached the modeling of the wound-healing process as a video prediction task, employing a two-stage strategy that combines a vector quantized variational autoencoder with an autoregressive transformer. Our trained model successfully generates realistic videos based on the initial frames of the healing process. In this work, a BiaPy workflow was used for the semantic segmentation of the wounds on each video frame.

CartoCell: large-scale analysis of epithelial cysts ([30]). A significant challenge in creating neural networks for cell segmentation is the requirement for labor-intensive manual curation to develop training datasets. CartoCell overcomes this limitation by creating an automated image-analysis pipeline in BiaPy, which efficiently leverages small datasets to generate accurate cell labels in intricate 3D epithelial settings. This workflow enables fast generation of high-quality epithelial reconstructions and, thus, detailed analysis of morphological features. A detailed description and step-by-step tutorial of this workflow are available in BiaPy's documentation site under "Tutorials".

Analysis of stable deep learning architectures for mitochondria segmentation ([21]). Recent advancements in deep learning models have demonstrated remarkable results in mitochondria segmentation; however, the absence of code and detailed training information frequently impedes their reproducibility. This study adheres to best practices, thoroughly comparing leading-edge architectures and various U-net model adaptations, all implemented as full workflows in BiaPy.

[4] `https://mitoem.grand-challenge.org/`

Software Tool	Analysis Tasks						Accessibility & Portability			Advanced Features		
	Classification	Semantic/Instance Segmentation	Detection	Denoising	Super-resolution	Self-supervised Learning	Integrated Solution	GUI/Notebooks	Docker Containers	Transformer Models	Multi-GPU	Large files Support
Cellpose [7]	✓	✓					✓	✓				
Cellprofiler [8]	✓	✓	✓				✓	✓				
Ilastik [9]	✓	✓					✓	✓			✓	✓
PlantSeg [10]		✓					✓	✓			✓	✓
DL4MicEverywhere [18]	✓	✓	✓	✓	✓			✓	✓			
MIA [11]		✓					✓					
HistomicsML2 [45]	✓							✓				
InstantDL [14]	✓	✓										
CDeep3M2 [46]	✓										✓	✓
NucleAIzer [15]		✓						✓				
ImJoy [6]	✓	✓										
PyTC [47]	✓	✓		✓	✓			✓		✓		
QuPath [13]	✓	✓	✓				✓	✓				✓
Icy [17]		✓	✓	✓			✓	✓				
BiaPy (ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of features available in BiaPy and similar state-of-the-art software tools, based on (1) the bioimage analysis tasks they target, (2) their accessibility and portability features, and (3) the advanced technological features they provide.

References

[20] O. Ronneberger, P. Fischer, and T. Brox. "U-Net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer. 2015, pp. 234–241.

[21] D. Franco-Barranco, A. Muñoz-Barrutia, and I. Arganda-Carreras. "Stable deep neural network architectures for mitochondria segmentation on electron microscopy volumes". In: *Neuroinformatics* 20.2 (2022), pp. 437–450.

[22] Z. Zhang, Q. Liu, and Y. Wang. "Road extraction by Deep Residual U-Net". In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018), pp. 749–753.

[23] D. Jha et al. "Resunet++: An advanced architecture for medical image segmentation". In: *2019 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2019, pp. 225–2255.

[24] O. Oktay et al. "Attention U-Net: Learning where to look for the pancreas". In: *arXiv preprint arXiv:1804.03999* (2018).

[25] N. Ibtehaz and M. S. Rahman. "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation". In: *Neural Networks* 121 (2020), pp. 74–87.

[26] A. Hatamizadeh et al. "UNETR: Transformers for 3D medical image segmentation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 574–584.

[27] D. Wei et al. "MitoEM Dataset: Large-Scale 3D Mitochondria Instance Segmentation from EM Images". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer. 2020, pp. 66–76.

[28] D. Franco-Barranco et al. "Current Progress and Challenges in Large-Scale 3D Mitochondria Instance Segmentation". In: *IEEE Transactions on Medical Imaging* 42.12 (2023), pp. 3956–3971. DOI: 10.1109/TMI.2023.3320497.

[29] F. Meyer. "Topographic distance and watershed lines". In: *Signal Processing* 38.1 (1994), pp. 113–125.

[30] J. A. Andres-San Roman et al. "CartoCell, a high-content pipeline for 3D image analysis, unveils cell morphology patterns in epithelia". In: *Cell Reports Methods* 3.10 (2023).

[31] X. Zhou, V. Koltun, and P. Krähenbühl. "Tracking objects as points". In: *European Conference on Computer Vision*. Springer. 2020, pp. 474–490.

[32] A. Krull, T.-O. Buchholz, and F. Jug. "Noise2void-learning denoising from single noisy images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2129–2137.

[33] B. Lim et al. "Enhanced deep residual networks for single image super-resolution". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 136–144.

[34] Y. Zhang et al. "Image super-resolution using very deep residual channel attention networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 286–301.

[35] C. Qiao et al. "Evaluation and development of deep neural networks for image super-resolution in optical microscopy". In: *Nature Methods* 18.2 (2021), pp. 194–202.

[36] J. Yu et al. "Wide activation for efficient and accurate image super-resolution". In: *arXiv preprint arXiv:1808.08718* (2018).

[37] D. Franco-Barranco et al. "Deep learning based domain adaptation for mitochondria segmentation on EM volumes". In: *Computer Methods and Programs in Biomedicine* 222 (2022), p. 106949.

[38] K. He et al. "Masked autoencoders are scalable vision learners". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16000–16009.

[39] M. Tan and Q. Le. "EfficientNet: Rethinking model scaling for convolutional neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6105–6114.

[40] T. DeVries and G. W. Taylor. "Improved regularization of convolutional neural networks with cutout". In: *arXiv preprint arXiv:1708.04552* (2017).

[41] J. Yoo, N. Ahn, and K.-A. Sohn. "Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8375–8384.

[42] S. Yun et al. "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6023–6032.

[43] P. Chen et al. "GridMask data augmentation". In: *arXiv preprint arXiv:2001.04086* (2020).

[44] L. Backová et al. "Modeling Wound Healing Using Vector Quantized Variational Autoencoders and Transformers". In: *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*. IEEE. 2023, pp. 1–5.

[45] Sanghoon Lee et al. "HistomicsML2. 0: Fast interactive machine learning for whole slide imaging data". In: *arXiv preprint arXiv:2001.11547* (2020).

[46] Matthias G Haberl et al. "CDeep3M-Preview: Online segmentation using the deep neural network model zoo". In: *BioRxiv* (2020), pp. 2020–03.

[47] Zudi Lin et al. "PyTorch connectomics: a scalable and flexible segmentation framework for EM connectomics". In: *arXiv preprint arXiv:2112.05754* (2021).