frontiers | Frontiers in Plant Science

TYPE Technology and Code
PUBLISHED 06 October 2022
DOI 10.3389/fpls.2022.964058

OPEN ACCESS

EDITED BY
Jucheng Yang,
Tianjin University of Science and Technology, China

REVIEWED BY
Michael Gomez Selvaraj,
Consultative Group on International Agricultural Research (CGIAR), United States
Hui Feng,
Huazhong Agricultural University, China
Yuan Wang,
Tianjin University of Science and Technology, China

*CORRESPONDENCE
Jianqiang Sun
sun@biunit.dev

SPECIALTY SECTION
This article was submitted to Sustainable and Intelligent Phytoprotection, a section of the journal Frontiers in Plant Science

RECEIVED 08 June 2022
ACCEPTED 20 September 2022
PUBLISHED 06 October 2022

CITATION
Sun J, Cao W and Yamanaka T (2022) JustDeepIt: Software tool with graphical and character user interfaces for deep learning-based object detection and segmentation in image analysis. *Front. Plant Sci.* 13:964058. doi: 10.3389/fpls.2022.964058

JustDeepIt: Software tool with graphical and character user interfaces for deep learning-based object detection and segmentation in image analysis

Jianqiang Sun*, Wei Cao and Takehiko Yamanaka

Research Center for Agricultural Information Technology, National Agriculture and Food Research Organization (NARO), Tsukuba, Japan

Image processing and analysis based on deep learning are becoming mainstream and increasingly accessible for solving various scientific problems in diverse fields. However, it requires advanced computer programming skills and a basic familiarity with character user interfaces (CUIs). Consequently, programming beginners face a considerable technical hurdle. Because potential users of image analysis are experimentalists, who often use graphical user interfaces (GUIs) in their daily work, there is a need to develop GUI-based easy-to-use deep learning software to support their work. Here, we introduce JustDeepIt, a software written in Python, to simplify object detection and instance segmentation using deep learning. JustDeepIt provides both a GUI and a CUI. It contains various functional modules for model building and inference, and it is built upon the popular PyTorch, MMDetection, and Detectron2 libraries. The GUI is implemented using the Python library FastAPI, simplifying model building for various deep learning approaches for beginners. As practical examples of JustDeepIt, we prepared four case studies that cover critical issues in plant science: (1) wheat head detection with Faster R-CNN, YOLOv3, SSD, and RetinaNet; (2) sugar beet and weed segmentation with Mask R-CNN; (3) plant segmentation with U^2-Net; and (4) leaf segmentation with U^2-Net. The results support the wide applicability of JustDeepIt in plant science applications. In addition, we believe that JustDeepIt has the potential to be applied to deep learning-based image analysis in various fields beyond plant science.

KEYWORDS

Deep learning, image recognition, object detection, instance segmentation, leaf segmentation, plant segmentation, graphical user interface

1 Introduction

Over the past decade, remarkable advances have been made in image analysis based on deep learning in various fields (Jiang and Li, 2020; Ben Yedder et al., 2021; Liu and Wang, 2021). In practical applications in plant science field, deep learning for image analyses has been applied at different scales. For example, at the field scale, studies such as high-throughput phenotyping and yield prediction via images captured by drones or hyperspectral cameras are hot topics (Kamilaris and Prenafeta-Boldú, 2018; Jiang and Li, 2020). On an individual scale, studies including species classification, crop disease detection, and weed detection are well researched (Christin et al., 2019; Hasan et al., 2021; Liu and Wang, 2021). In addition, at the cell level, studies such as cell type identification and stomata identification via microscopic images have been performed (Moen et al., 2019; Zhu et al., 2021). The increased availability of these techniques in various fields enhances the importance of the roles they are expected to play in the future.

Image analysis based on deep learning can be roughly categorized into three main tasks: object classification, object detection, and instance segmentation. Object classification determines the class of an object in an image. Object detection specifies the types of objects in an image and their locations, generally through bounding boxes (i.e., rectangular delimiting areas). Instance segmentation selects a pixel-wise mask for each object in an image. As images generally contain multiple objects, object detection and instance segmentation have broader practical applications than object classification. In addition to instance segmentation, salient object detection is often used to detect the primary object in an image at the pixel level (Wang et al., 2017; Borji et al., 2019). It can also be applied to one-class instance segmentation for background removal, leaf segmentation, and root segmentation.

Applying machine learning models to images captured under conditions different from those of images captured for model training degrades the inference performance. This is called the frame problem in machine learning (Ford and Pylyshyn, 1996). It is impossible to solve this problem without collecting training images under all conditions. Hence, most practical applications restrict the usage conditions to ensure that the frame is not exceeded by limiting the target objects, shooting conditions, or by other means. In scientific studies, this problem is addressed through models devised for specific projects instead of previously developed models.

Python programming language and its libraries PyTorch (Paszke et al., 2019), MMDetection (Chen et al., 2019), and Detectron2 (Wu et al., 2019), have facilitated image analysis using deep learning. However, programming experience and machine learning expertise are required to implement complicated neural networks (i.e., deep learning models) for object detection and instance segmentation tasks.

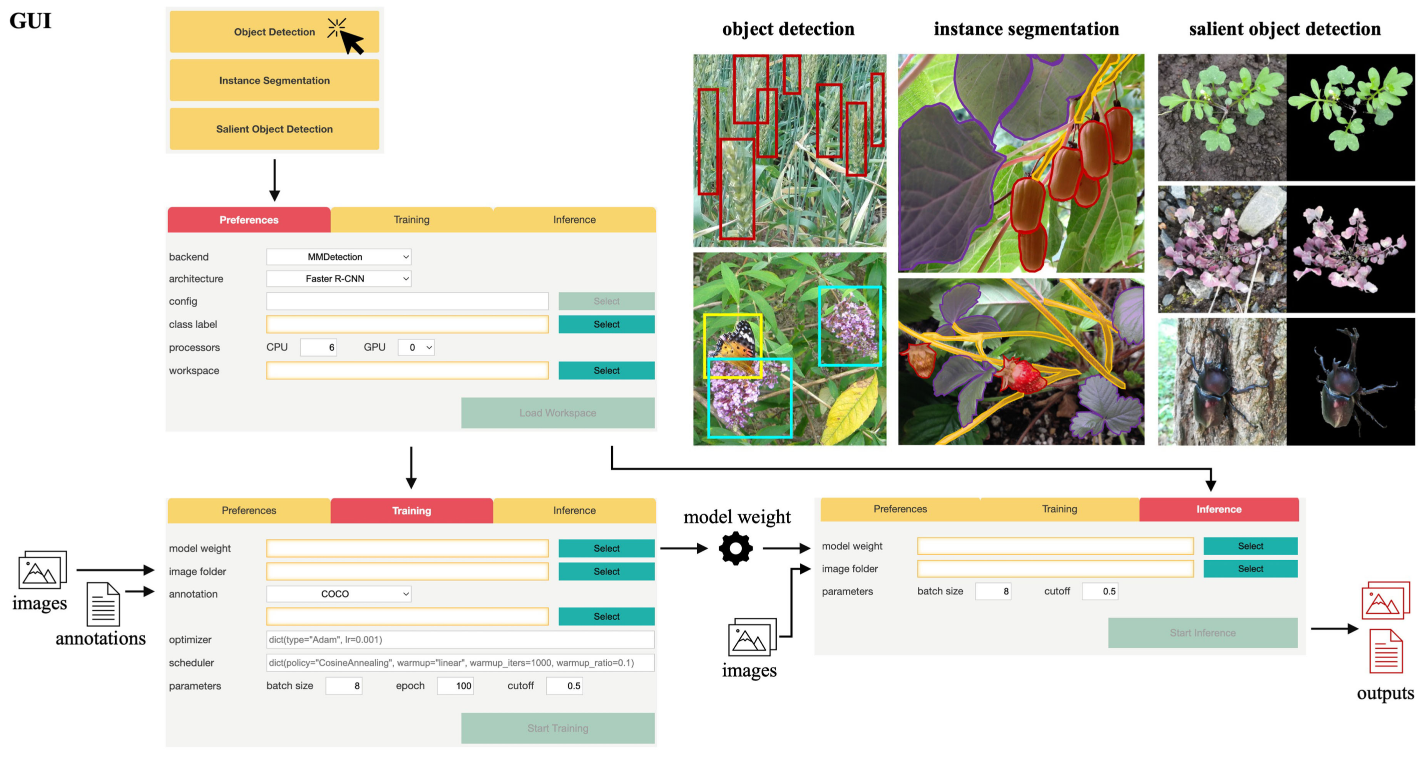
Given the required programming skills or machine learning expertise, the application of deep learning remains challenging for most beginners. Many experimentalists working full-time on wet experiments use graphical user interfaces (GUIs) in their daily work. In contrast, informatics researchers use character user interfaces (CUIs) because it is easy to perform large-scale experiments under different parameter combinations owing to the scalabilities of CUIs. Thus, suitable software should be supported on GUIs and CUIs to serve various users, making it advantageous for experimental and informatics researchers to use the same software and conduct collaborative research. Nevertheless, most existing open-source GUI-based software for deep learning-based image analysis only supports segmentation or is intended for specific purposes. For instance, RootPainter (Smith et al., 2022) and DeepMIB (Belevich and Jokitalo, 2021) support biological image segmentation using U-Net (Ronneberger et al., 2015). ZeroCostDL4Mic (von Chamier et al., 2021) implements You Only Look Once version 2 (YOLOv2) (Redmon and Farhadi, 2017) and U-Net for object detection and instance segmentation against microscopy images. Maize-IAS (Zhou et al., 2021) partially uses a faster region-based convolutional neural network (Faster R-CNN) (Ren et al., 2017) for leaf segmentation and leaf counting of maize images captured under the controlled environment. Moreover, most solutions are focused on GUIs but neglect CUIs, thus hindering expansion on the user side. Therefore, there is a high demand for image analysis software based on deep learning supporting both easy-to-use GUIs and high-scalability CUIs, to satisfy a diverse user base.

We developed the JustDeepIt software supporting GUI and CUI to train models and perform inference for object detection, instance segmentation, and salient object detection. JustDeepIt can be applied to many biological problems, such as wheat head detection, plant segmentation, and leaf segmentation. In addition, it provides an intuitive solution for biologists lacking programming experience and machine learning expertise, simplifying implementation compared with conventional programming schemes.

2 Method

JustDeepIt is implemented using Python and is easy to interoperate with various Python packages. It provides GUI and CUI for deep learning-based image analysis, including object detection, instance segmentation, and salient object detection (Figure 1). The source code is deposited in GitHub at https://github.com/biunit/JustDeepIt under an MIT License. An overview of implementations of the user interfaces and main functions of JustDeepIt are described in the following subsections. Detailed documentation, including installation guides and tutorials, is available on the project website.

GUI



Preferences

backend MMDetection
architecture Faster R-CNN
config
class label
processors CPU 6 GPU 0
workspace

Training

model weight
image folder
annotation COCO
optimizer dict(type="Adam", lr=0.001)
scheduler dict(policy="CosineAnnealing", warmup="linear", warmup_iters=1000, warmup_ratio=0.1)
parameters batch size 8 epoch 100 cutoff 0.5

Inference

model weight
image folder
parameters batch size 8 cutoff 0.5

CUI

```
from justdeepit.models import OD

coco_fmt = '/path/to/coco/annotation.json'
train_images_dpath = '/path/to/folder/images'

model = OD('class_label.txt', model_arch='fasterrcnn')

model.train(train_images_dpath, coco_fmt, 'COCO')

model.save('trained_weight.pth')
```

```
from justdeepit.models import OD

test_images = ['test1.jpg', 'test2.jpg']

model = OD('class_label.txt', model_arch='fasterrcnn',
           model_weight='trained_weight.pth')

outputs = model.inference(test_images)

outputs.format('coco', 'predicted_outputs.coco.json')
```

FIGURE 1
Overview of GUI and CUI implementation of JustDeepIt. For GUI usage, four main screens are implemented to select the analysis task, set standard parameters (e.g., neural network architecture, temporary folder), train a model, and infer new images with the trained model, respectively. For CUI usage, import the modules based on analysis tasks and then use the initialization method (e.g., **OD**), **train, save, and inference** methods to set up a model, train the model, save the model, and infer new images with the trained model, respectively.

2.1 Implementation of user interfaces

The GUI of JustDeepIt is implemented using FastAPI (Ramírez, 2018), a straightforward Python library for building simple web applications. It allows deep learning-based image analysis tasks with simple mouse and keyboard operations without writing codes (Figure 1). Upon launching the GUI, the user is prompted to select an analysis task; after this selection, the analysis screen is displayed. The analysis screen has three tabs, *Preferences*, *Training*, and *Inference*. The *Preferences* tab allows users to set standard parameters for training and inference, such as neural network architecture, class labels (object names targeted for detection or segmentation), a temporary directory to save the intermediate and final results, and more. The *Training* and *Inference* tabs are used to train a model by specifying training images and the corresponding annotations and inferring new images with the trained model, respectively.

The CUI of JustDeepIt can be used via application programming interfaces (APIs). The most complicated procedures (e.g., data registration, model initialization, output adjustment) are encapsulated into the APIs containing a few intelligible functions to simplify usage. The three main API functions are **train** for training detection or segmentation models, **save** for saving the trained model weights, and **inference** for detection or segmentation on test images. Example codes for training object detection models and using the model for inference are shown in Figure 1. Additional usage examples (e.g., building a web application) and arguments of these functions are available on the project website.

2.2 Object detection and instance segmentation

Object detection and instance segmentation models in JustDeepIt are internally built based on the MMDetection (Chen et al., 2019) or Detectron2 (Wu et al., 2019) libraries. The user can choose MMDetection or Detectron2 as the backend to build the corresponding models. JustDeepIt supports various well-known neural network architectures. For object detection, Faster R-CNN (Ren et al., 2017), YOLOv3 (Redmon and Farhadi, 2018), Single-Shot Multibox Detector (SSD) (Liu et al., 2016), and RetinaNet (Lin et al., 2018) are available to meet different user needs. For instance segmentation, Mask R-CNN (He et al., 2017) is available. Furthermore, JustDeepIt allows user-customized neural network architectures to accommodate users who wish to use architectures that are not implemented in the software. For example, Faster R-CNN implemented in JustDeepIt uses VGGNet (Simonyan and Zisserman, 2015) for feature extraction during the detection process; the users may want to change VGGNet to other architectures such as ResNet (He et al., 2015). To accomplish this, users can (i) either download already-prepared architecture configuration files from MMDetection or Detectron2 GitHub repositories or create their own configuration file from scratch and then (ii) input the configuration file into JustDeepIt to build a model for training and inference.

For model training, JustDeepIt requires image annotations in the Common Objects in Context (COCO) format, which can be generated using GUI-based free software such as COCO Annotator (Brooks, 2021) and Computer Vision Annotation Tool (Boris et al., 2021). After the user specifies the location of the image dataset and corresponding annotations through the *Training* tab of GUI or the training function of CUI, JustDeepIt uses them to build the related model.

For object detection using the trained model, the user can specify the trained weights and folder containing the test images for detection through the *Inference* tab of GUI or the inference function of CUI. For GUI usage, the inference results are automatically saved as images with bounding boxes or contour lines around the detected objects and a JSON file of the inference results in COCO format. For CUI usage, the user can specify whether the inference results should be saved as annotated images or an annotation file.

2.3 Salient object detection

The module for salient object detection in JustDeepIt is based on U^2-Net (Qin et al., 2020) and written using the PyTorch library. The GUI and the training and detection functions processing are similar to those used for object detection.

JustDeepIt requires training images with annotations (either COCO format annotations or mask images) for model training. Although the U^2-Net implementation for JustDeepIt requires images of 288 × 288 pixels as the canonical input, images of various sizes are captured for applications in plant science. Thus, JustDeepIt provides two approaches for training on images of various sizes: *resizing* and *random cropping* (Figure 2A). *Resizing* changes the image resolution to 288 × 288 pixels for training in U^2-Net. This approach is suitable for images containing a few large target objects. In contrast, *random cropping* randomly selects small areas of $p \times p$ pixels at random angles from the original images, where p can be specified according to the complexity of the intended images and tasks. The images of $p \times p$ pixels are then resized to 288 × 288 pixels for training. This approach is suitable for images containing many small target objects and details.

JustDeepIt implements *resizing* and *sliding* for salient object detection (Figure 2B). *Resizing* changes the scale of the input images to 288× 288 pixels for detection and then restores the detection result to the input size. *Sliding* first crops square areas of $p \times p$ pixels from the top left to the bottom right of the image step-by-step, where p can be specified by the user. It then resizes the areas to 288 × 288 pixels to perform salient object detection

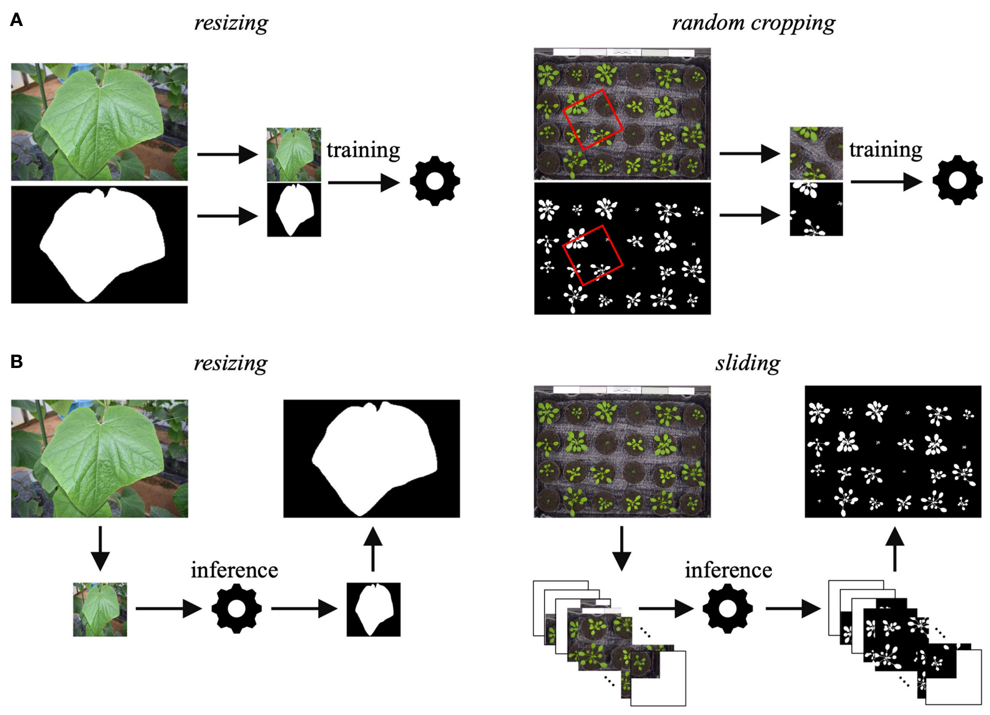


FIGURE 2
(A) Training approaches of *resizing* and *random cropping* are implemented in JustDeepIt. **(B)** Inference approaches of *resizing* and *sliding* (corresponding to training using *resizing* and *random cropping*, respectively) are implemented in JustDeepIt.

in each area. Finally, it merges the processed areas into a single image. In addition to salient object detection, summarization functions (e.g., counting the number of objects in the image, quantifying colors, measuring the area of each object) are available in JustDeepIt.

3 Results

We prepared four case studies as practical examples of JustDeepIt and reported the results in this section. Detailed procedures for these case studies can be found in the JustDeepIt documentation.

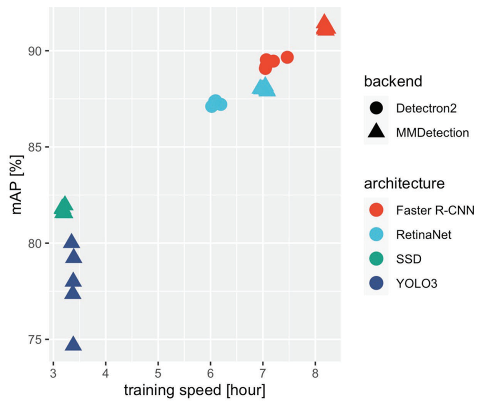
3.1 Wheat head detection

We show an example of JustDeepIt performing object detection for the wheat head, a prevalent task in plant science. The global wheat head detection (GWHD) dataset, containing 4700 images of 1024 × 1024 pixels for wheat head detection evaluation (David et al., 2020), was used in this case study. We randomly selected 80% of the images in the GWHD dataset for training and used the remaining 20% for validation. We constructed Faster R-CNN, YOLOv3, SSD, and RetinaNet with MMDetection backend and Faster R-CNN and RetinaNet with Detectron2 backend for training and validation with the GWHD dataset. To initialize each architecture, we retrieved the pretraining weights from the GitHub repositories of MMDetection and Detectron2.

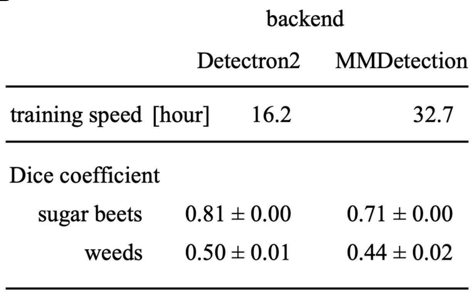
The training was performed for 100 epochs with a batch size of eight and an initial learning rate of 0.0001. Validation was performed using the trained architectures against the validation images, and the mean average precision (mAP) was calculated from the validation results. Training and validation were independently repeated five times to mitigate the influence of randomness. These processes were executed on an Ubuntu 18.04 system equipped with an NVIDIA Tesla V100 SXM2 graphics processor, an Intel Xeon Gold 6254 processor, and 64 GB of memory.

For the five training and validation runs, Faster R-CNN provided a relatively high validation mAP with a relatively slow training speed, YOLOv3 and SSD provided lower mAP with faster training speed, and RetinaNet provided intermediate mAP and training speed (Figure 3A). In addition, the Faster R-CNN and RetinaNet with MMDetection backend provided slower training and higher validation mAP than those with the Detectron2 backend. Hence, different backends and neural network architectures provided distinct performances and training speeds, and users should select the appropriate backend and architecture according to the application.

A

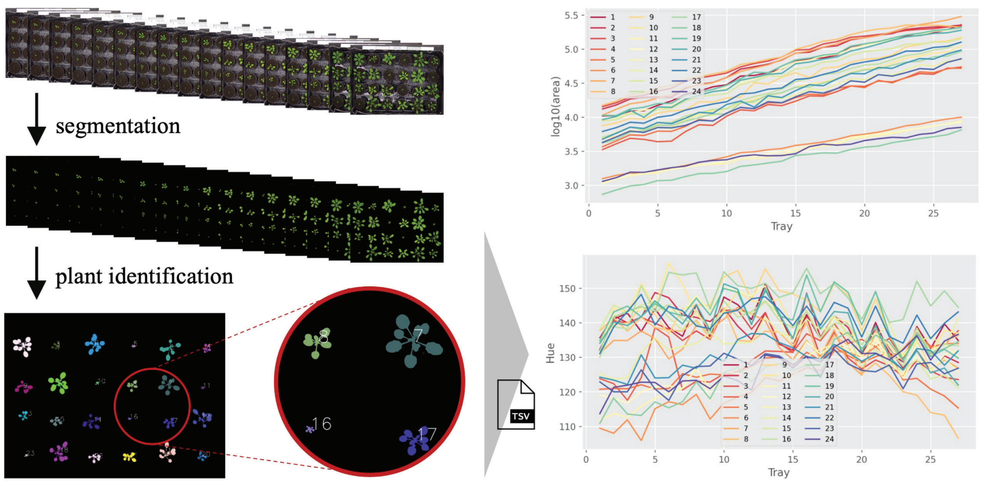


B



	backend	
	Detectron2	MMDetection
training speed [hour]	16.2	32.7
Dice coefficient		
sugar beets	0.81 ± 0.00	0.71 ± 0.00
weeds	0.50 ± 0.01	0.44 ± 0.02

C



D

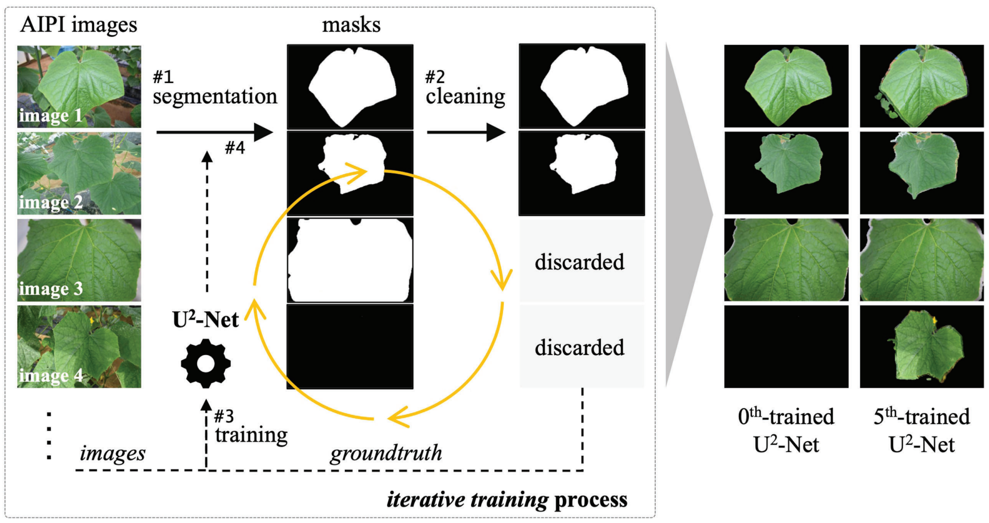


FIGURE 3
Results of three case studies. **(A)** Training time and validation mAP of object detection. **(B)** Training time and validation mAP of instance segmentation. **(C)** The process and analysis result (projected area and color hue) of plant segmentation with time-series plant images. **(D)** Processes and sample results of iterative training of U^2-Net for leaf segmentation.

3.2 Sugar beets and weeds segmentation

To represent the case study of instance segmentation with JustDeepIt, we performed weed and crop (here, sugar beet) segmentation, which is one of the tasks in high-demand in the agriculture sector, on the SugarBeets2016 dataset (Chebrolu et al., 2017). The SugarBeets2016 dataset has 11,552 RGB images captured under fields and the annotations for sugar beets and weeds. We randomly selected 5,000 and 1,000 images for training and validation, respectively. We constructed Mask R-CNN with MMDetection and Detectron2 backends for training and validation. We retrieved the pretraining weights from the GitHub repositories for initializing each architecture, as in subsection 3.1.

Training and validation were performed using the same parameters, processes, and system devices as in subsection 3.1, except that the batch size was set to four. The Dice coefficient was calculated from the validation results. For the five training and validation runs, Mask R-CNN with Detectron2 backend provided relatively higher Dice coefficients and faster training speed than that with MMDetection backend (Figure 3B). As in the case of object detection, the result was well characterized by the backend.

3.3 Plant segmentation with U^2-Net

As a case study of salient object detection with JustDeepIt, we use the Plant Phenotyping Dataset, a popular benchmark dataset for plant segmentation (Minervini et al., 2015; Minervini et al., 2016). The dataset contains 27 images of 3108 × 2324 pixels, and each image contains 24 individual plants. We used U^2-Net implemented in JustDeepIt for plant segmentation. We randomly selected three images to train U^2-Net using the *random cropping* approach. The training was performed with the default parameters in a macOS Big Sur system equipped with a Quad-Core Intel Core i5 processor (2.3 GHz) and 16 GB of memory. The training took approximately 6.5 hours with four processors but without any graphics processor.

Salient object detection was performed using the trained U^2-Net with the *sliding* approach for the 27 images with default parameters. Detection took approximately 2.3 hours, as the 27 images (3108 × 2324 pixels) were sliced into 27 × 130 small images (320 × 320 pixels) by the *sliding* approach, requiring substantial processing time. In addition, as the 27 images show time-series data, we summarized object statistics (e.g., projected area, plant color) over time (Figure 3C). The result indicates that the areas of the plants increased, and their colors varied over time.

3.4 Iterative training of U^2-Net for improved leaf segmentation

Extracting salient objects by removing the background may improve the performance of image analysis and can be applied to various image analysis tasks. As another case study for salient object detection, we trained U^2-Net for leaf segmentation on the Pest Damage Image Dataset (National Agriculture and Food Research Organization (NARO), 2021). The dataset comprises images of four crops, including cucumber tagged by disease or pest names, whereas no annotations of bounding boxes or segmentation are available. Here, we proposed iterative training to U^2-Net using unannotated images for leaf segmentation for cucumber.

Iterative training proceeded as follows (Figure 3D). In step 1, we used U^2-Net trained on the DUTS dataset (Wang et al., 2017) (0th-trained U^2-Net) obtained from the corresponding GitHub repository (Qin et al., 2021) for leaf segmentation of cucumber leaf images. In step 2, the images whose nearly complete area was detected as a salient object (image 3 in Figure 3D) or without detection (image 4 in Figure 3D) were discarded. In step 3, we used the remaining images and detection results (i.e., mask images) to retrain U^2-Net. In step 4, we used the trained U^2-Net from step 3 to perform salient object detection for the cucumber leaf images again. Then, we repeated steps 2–4 to retrain U^2-Net five times, obtaining the 5th-trained U^2-Net, and training was performed using the CUI for efficiency.

The 0th-trained U^2-Net failed to detect leaves in images containing multiple leaves. In contrast, the 5th-trained U^2-Net successfully detected the main salient leaf in every image (Figure 3D). Thus, even without annotations, we built a model for leaf segmentation using existing techniques. General users can use such approaches *via* the simple JustDeepIt API to extend the range of applications.

4 Discussion

The widespread use of deep learning technologies is gradually contributing to various scientific fields. Thus, it is vital to support the ease of technology usage for everyone, regardless of their research backgrounds and programming skills. As experimental researchers use GUIs and informatics researchers use CUIs mostly, developing software, which supports GUIs and CUIs and is not restricted to any specific tasks, is essential.

In the field of plant sciences, various GUI-based software, implemented for deep learning-based image analysis, has been developed. However, most software implementing one or a few neural network architectures to solve specific problems and only support GUI (Belevich and Jokitalo, 2021; von Chamier et al., 2021; Smith et al., 2022; Zhou et al., 2021). In contrast, JustDeepIt was developed to fill these gaps, enabling various image analysis tasks using deep learning technologies in a single software. JustDeepIt implements multiple neural network architectures suitable for different application scenarios and supports both GUI and CUI, increasing flexibility according to the task. As shown in our case studies with JustDeepIt, GUI is ideal for building models from available data effortlessly, and CUI is suitable for facilitating the use of a model as an extension (e.g., iterative training of U^2-Net). Furthermore, we believe that by supporting both GUI and CUI, collaborative research between experimental and informatics researchers can proceed more efficiently.

Other than deep learning-based software, scikit-image (van der Walt et al., 2014), ImageJ (Abràmoff et al., 2004), and PlantCV (Gehan et al., 2017) are also available for image processing and are broadly used in many applications (e.g., plant detection and leaf segmentation). Scikit-image and ImageJ require users to set thresholds manually for multiple color spaces to segment leaf areas. Therefore, if an image consists of various phenotypes of plants, for example, plants with green and red leaves due to some stress, simultaneously segmenting both plants may be challenging. PlantCV supports building task-specific machine learning models for instance segmentation. However, it does not support GUI and requires programming skills. Given these open-source packages, choosing the appropriate one for a task or a specific problem is often demanding. JustDeepIt is expected to address complicated issues and accelerate research on image analysis when used in combination with other software.

In plant science and agriculture, fruit detection and plant segmentation are two high-priority tasks (Arya et al., 2022; Zhou et al., 2022). This is because these tasks can estimate growth stages and yields of plants, including crops, and improve plant environmental robustness (e.g., disease resistance, fruit quality, and fruit yields) by collaborating with genomic technologies (e.g., genome-wide association study and expression quantitative trait locus analysis) (Varshney et al., 2021; Xiao et al., 2022). In this study, we represented the four case studies covering the two high-priority tasks with both GUI and CUI of JustDeepIt. The results support the robustness of JustDeepIt against critical issues in plant science. In addition, although JustDeepIt was intended for plant research, it may be applicable for image analysis in various disciplines beyond plant science. Furthermore, in a future version, we will continue to update the software in response to user demand and the technology flow.

Data availability statement

The original contributions presented in the study are publicly available. This data can be found here: https://github.com/biunit/JustDeepIt.

Author contributions

JS conceived the ideas and designed the software. JS and WC developed the software. JS, WC, and TY wrote the manuscript. All authors have read and approved the final manuscript.

Funding

This study was supported by a research project of the Ministry of Agriculture, Forestry, and Fisheries and a Public/Private R&D Investment Strategic Expansion Program (PRISM) of the cabinet office of Japan, and JSPS KAKENHI Grant Number 22H05179.

Acknowledgments

Computations were partially performed on the SHIHO supercomputer at the National Agriculture and Food Research Organization (NARO), Japan.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abràmoff, M. D., Magalhães, P. J., and Ram, S. J. (2004). Image processing with ImageJ. *Biophotonics Int.* 11 (7), 36–42.

Arya, S., Sandhu, K. S., Singh, J., and Kumar, S. (2022). Deep learning: as the new frontier in high-throughput plant phenotyping. *Euphytica* 218, 47. doi: 10.1007/s10681-022-02992-3

Belevich, I., and Jokitalo, E. (2021). DeepMIB: User-friendly and open-source software for training of deep learning network for biological image segmentation. *PloS Comput. Biol.* 17, e1008374. doi: 10.1371/journal.pcbi.1008374

Ben Yedder, H., Cardoen, B., and Hamarneh, G. (2021). Deep learning for biomedical image reconstruction: a survey. *Artif. Intell. Rev.* 54, 215–251. doi: 10.1007/s10462-020-09861-2

Boris, S., Nikita, M., Maxim, Z., Andrey, Z., Dmitry, K., Ben, H., et al. (2021) *Computer vision annotation tool (CVAT)*. Available at: https://github.com/openvinotoolkit/cvat (Accessed October 5, 2021).

Borji, A., Cheng, M.-M., Hou, Q., Jiang, H., and Li, J. (2019). Salient object detection: A survey. *Comput. Visual Media* 5, 117–150. doi: 10.1007/s41095-019-0149-9

Brooks, J. (2021) *COCO annotator*. Available at: https://github.com/jsbroks/coco-annotator (Accessed October 5, 2021).

Chebrolu, N., Lottes, P., Schaefer, A., Winterhalter, W., Burgard, W., and Stachniss, C. (2017). Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields. *Int. J. Rob. Res.* 36 (10):1045–1052. doi: 10.1177/0278364917720510

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., et al. (2019) *MMDetection: Open MMLab detection toolbox and benchmark* (arXiv). Available at: http://arxiv.org/abs/1906.07155 (Accessed October 8, 2021).

Christin, S., Hervet, É., and Lecomte, N. (2019). Applications for deep learning in ecology. *Methods Ecol. Evol.* 10 (10), 1632–1644. doi: 10.1111/2041-210X.13256

David, E., Madec, S., Sadeghi-Tehran, P., Aasen, H., Zheng, B., Liu, S., et al (2020). Global wheat head detection (GWHD) Dataset: A large and diverse dataset of high-resolution RGB-Labelled images to develop and benchmark wheat head detection methods. *Plant Phenomics*, 2020, 3521852. doi: 10.34133/2020/3521852

Ford, K. M., and Pylyshyn, Z. W. (1996). "The frame problem in artificial intelligence," in *The robot's dilemma revisited* (USA: Ablex Publishing Corp).

Gehan, M. A., Fahlgren, N., Abbasi, A., Berry, J. C., Callen, S. T., Chavez, L., et al (2017). PlantCV v2: Image analysis software for high-throughput plant phenotyping. *PeerJ*, 5:e4088. doi: 10.7717/peerj.4088

Hasan, A. S. M. M., Sohel, F., Diepeveen, D., Laga, H., and Jones, M. G. K. (2021). A survey of deep learning techniques for weed detection from images. *Comput. Electron Agric.*, 184:106067. doi: 10.1016/j.compag.2021.106067

He, K., Gkioxari, G., Dollár, P., Girshick, R., and Mask, R-CNN. (2017) (arXiv). Available at: http://arxiv.org/abs/1703.06870 (Accessed October 8, 2021).

He, K., Zhang, X., Ren, S., and Sun, J. (2015) *Deep residual learning for image recognition* (arXiv). Available at: https://arxiv.org/abs/1512.03385 (Accessed October 8, 2021).

Jiang, Y., and Li, C. (2020). Convolutional neural networks for image-based high-throughput plant phenotyping: a review. *Plant Phenomics*, 2020:4152816. doi: 10.34133/2020/4152816

National Agriculture and Food Research Organization (NARO) (2021) *Pest Damage Image Database (in Japanese)*. Available at: https://www.naro.affrc.go.jp/org/niaes/damage/ (Accessed October 5, 2021).

Kamilaris, A., and Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Comput. Electron Agric.* 147, 70–90. doi: 10.1016/j.compag.2018.02.016

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2018) *Focal loss for dense object detection* (arXiv). Available at: http://arxiv.org/abs/1708.02002v2 (Accessed October 5, 2021).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et al. (2016). SSD: single shot multiBox detector. *Comput. Vision – ECCV 2016*, 9905:21–37. doi: 10.1007/978-3-319-46448-0_2

Liu, J., and Wang, X. (2021). Plant diseases and pests detection based on deep learning: a review. *Plant Methods* 17, 22. doi: 10.1186/s13007-021-00722-9

Minervini, M., Fischbach, A., Scharr, H., and Tsaftaris, S. A. (2015) *Plant phenotyping datasets*. Available at: https://www.plant-phenotyping.org/datasets-home (Accessed October 8, 2021).

Minervini, M., Fischbach, A., Scharr, H., and Tsaftaris, S. A. (2016). Finely-grained annotated datasets for image-based plant phenotyping. *Pattern Recognit. Lett.* 81, 80–89. doi: 10.1016/j.patrec.2015.10.013

Moen, E., Bannon, D., Kudo, T., Graf, W., Covert, M., and Valen, D. V. (2019). Deep learning for cellular image analysis. *Nat. Methods* 16, 1233–1246. doi: 10.1038/s41592-019-0403-1

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019) *PyTorch: an imperative style, high-performance deep learning library* (arXiv). Available at: http://arxiv.org/abs/1912.01703 (Accessed October 8, 2021).

Qin, X., Zhang, Z., Huang, C., Dehghan, M., Zaiane, O., and Jagersand, M. (2020). U2-net: going deeper with nested U-structure for salient object detection. *Pattern Recognition* 106, 107404. doi: 10.1016/j.patcog.2020.107404

Qin, X., Zhang, Z., Huang, C., Dehghan, M., Zaiane, O. R., and Jagersand, M. (2021) *U2-net: going deeper with nested U-structure for salient object detection*. Available at: https://github.com/xuebinqin/U-2-Net (Accessed June 10, 2021).

Ramírez, S. (2018) *FastAPI*. Available at: https://github.com/tiangolo/fastapi (Accessed October 8, 2021).

Redmon, J., and Farhadi, A. (2017). "YOLO9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (IEEE)*. doi: 10.1109/cvpr.2017.690

Redmon, J., and Farhadi, A. (2018) *YOLOv3: An incremental improvement* (arXiv). Available at: http://arxiv.org/abs/1804.02767 (Accessed October 8, 2021).

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 1137–1149. doi: 10.1109/TPAMI.2016.2577031

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: convolutional networks for biomedical image segmentation. *Lect. Notes Comput. Sci.* 9351, 234–241. doi: 10.1007/978-3-319-24574-4_28

Simonyan, K., and Zisserman, A. (2015) *Very deep convolutional networks for large-scale image recognition* (arXiv). Available at: https://arxiv.org/abs/1409.1556 (Accessed October 8, 2021).

Smith, A. G., Han, E., Petersen, J., Olsen, N. A. F., Giese, C., Athmann, M., et al. (2022). RootPainter: deep learning segmentation of biological images with corrective annotation. *New Phytol. press.* doi: 10.1111/nph

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., et al. (2014). Scikit-image: image processing in Python. *PeerJ.* 2, e453. doi: 10.7717/peerj.453

Varshney, R. K., Bohra, A., Roorkiwal, M., Barmukh, R., Cowling, W. A., Chitikineni, A., et al. (2021). Fast-forward breeding for a food-secure world. *Treads Genet.* 37 (12), 1124–1136. doi: 10.1016/j.tig.2021.08.002

von Chamier, L., Laine, R. F., Jukkala, J., Spahn, C., Krentzel, D., Nehme, E., et al. (2021). Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nat. Commun.* 12,2276. doi: 10.1038/s41467-021-22518-0

Wang, L., Lu, H., Wang, Y., Feng, M., Wang, D., Yin, B., et al. (2017). "Learning to detect salient objects with image-level supervision," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2017.404

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019) *Detectron2*. Available at: https://github.com/facebookresearch/detectron2 (Accessed June 10, 2021).

Xiao, Q., Bai, X., Zhang, C., and He, Y. (2022). Advanced high-throughput plant phenotyping techniques for genome-wide association studies: A review. *J. Adv. Res.* 35, 215–230. doi: 10.1016/j.jare.2021.05.002

Zhou, S., Chai, X., Yang, Z., Wang, H., Yang, C., and Sun, T. (2021). Maize-IAS: a maize image analysis software using deep learning for high-throughput plant phenotyping. *Plant Methods* 17, 48. doi: 10.1186/s13007-021-00747-0

Zhou, J., Vong, C. N., and Zhou, J. (2022). "Imaging technology for high-throughput plant phenotyping," in *Sensing, data managing, and control technologies for agricultural systems. agriculture automation and control*. Eds. S. Ma, T. Lin, E. Mao, Z. Song and K. C. Ting (Cham: Springer), 75–99. doi: 10.1007/978-3-031-03834-1_4

Zhu, C., Hu, Y., Mao, H., Li, S., Li, F., Zhao, C., et al. (2021). A deep learning-based method for automatic assessment of stomatal index in wheat microscopic images of leaf epidermis. *Front. Plant Sci.* 12. doi: 10.3389/fpls.2021.716784