Received: 20 August 2021 | Accepted: 11 November 2021
DOI: 10.1111/2041-210X.13787

Methods in Ecology and Evolution BRITISH ECOLOGICAL SOCIETY

APPLICATION

GinJinn2: Object detection and segmentation for ecology and evolution

Tankred Ott | Ulrich Lautenschlager

Evolutionary and Systematic Botany Group, Institute of Plant Biology, University of Regensburg, Regensburg, Germany

Correspondence
Ulrich Lautenschlager
Email: ulrich.lautenschlager@ur.de

Funding information
Deutsche Forschungsgemeinschaft, Grant/Award Number: OB 155/13-1

Handling Editor: Arthur Porto

Abstract

1. Collection and preparation of empirical data still represent one of the most important, but also expensive steps in ecological and evolutionary/systematic research. Modern machine learning approaches, however, have the potential to automate a variety of tasks, which until recently could only be performed manually. Unfortunately, the application of such methods by researchers outside the field is hampered by technical difficulties.
2. Here, we present GinJinn2, a user-friendly toolbox for deep learning-based object detection and instance segmentation on image data. Besides providing a convenient command-line interface to existing software libraries, it comprises several additional tools for data handling, pre- and postprocessing, and building advanced analysis pipelines.
3. We demonstrate the application of GinJinn2 for biological purposes using four exemplary analyses, namely the evaluation of seed mixtures, detection of insects on glue traps, segmentation of stomata and extraction of leaf silhouettes from herbarium specimens.
4. GinJinn2, by providing a coding-free environment, will enable users with a primary background in biology to apply deep learning-based methods for object detection and segmentation in order to automate feature extraction from image data.

KEYWORDS
automation, computer vision, deep learning, instance segmentation, machine learning, morphometrics, object detection, trait extraction

1 | INTRODUCTION

Leveraging image data for ecological and evolutionary/systematic research typically requires substantial effort for data collection and preparation. The ability to automate time-consuming steps of this process, possibly along with further downstream analyses, for example, using programming languages like Python or R, can not only increase productivity, but also allow otherwise infeasible large-scale analyses. Recent advances in machine learning (ML), both on the soft- and hardware side, make it even possible to automate tasks that are difficult to solve by means of classically designed algorithms. Computer vision, in particular, has largely profited from deep

Tankred Ott and Ulrich Lautenschlager have contributed equally to this work.

learning, which increasingly influences even the more traditional branches of organismic biology. Species identification tools running on smartphone devices (for an overview, see Jones, 2020; Wäldchen & Mäder, 2018) are prominent examples for this trend. Beyond pure classification tasks, a technically even more challenging problem consists in localizing objects like cells, organs or individuals on images. Specialized tools address this problem for various areas of application, such as crop or weed detection (e.g. Afonso et al., 2020; Buddha et al., 2019), detection of leaves and other plant organs on herbarium specimens (e.g. Ott et al., 2020; Weaver et al., 2020; Younis et al., 2020), stomata counting using microscopic leaf images (e.g. Fetter et al., 2019), animal counting using camera traps (Norouzzadeh et al., 2021) and many more. Moreover, DeepImageJ (Gómez-de-Mariscal et al., 2021), an optional plugin for the popular ImageJ program (Schneider et al., 2012; Schroeder et al., 2020), provides easy access to a number of trained deep learning models for pre-defined tasks via a graphical user interface.

Despite the availability of increasingly convenient frameworks, adapting well-established ML methods to new areas of application typically requires an amount of technical knowledge that may discourage potential users. GinJinn2, whose core functionality is based on Detectron2 (Wu et al., 2019), aims at lowering this hurdle by providing an easy-to-use command-line interface to the latter, augmented by a number of utility functions, designed to help the user with building custom analysis pipelines. While GinJinn (Ott et al., 2020) focused on extracting leaves from digitized herbarium specimens, GinJinn2 aims at a wider scope of application. Unlike the former, which was based on the Tensorflow object detection API, it is not restricted to bounding-box object detection, but also incorporates functionality for instance segmentation, that is, pixel-precise detection and classification of individual objects.

In the present contribution, a number of example analyses demonstrate how ecological, agricultural or evolutionary/systematic studies may benefit from GinJinn2. Those include pest monitoring using yellow glue traps, leaf shape extraction from herbarium specimens, stomata segmentation and the evaluation of seed mixtures. We hope to encourage interested researchers to consider deep learning-based object detection or segmentation when faced with similar tasks. Using GinJinn2 together with pretrained models from Detectron2's model zoo, new applications can be explored with a minimum of invested time and effort, which makes it a potentially useful tool for both beginners and advanced users.

2 | SOFTWARE

2.1 | Overview

GinJinn2 is a toolbox for deep learning-based bounding-box object detection and instance segmentation. As such, it provides functionality for model training, evaluation and application based on the Detectron2 framework, segmentation refinement based on CascadePSP (Cheng et al., 2020), a set of data pre- and postprocessing tools for handling annotated image datasets, and capabilities for data insight and visualization. GinJinn2 is not meant to be a replacement for existing frameworks like Detectron2 or the Tensorflow Object Detection API (Huang et al., 2017), but rather a toolkit enabling code-free access to deep learning-based object detection technologies. All of GinJinn2's functionality is accessible via an easy-to-use command-line interface (CLI).

2.2 | Dataset splitting

Besides the data used to train the model, it is generally advisable to use a so-called validation dataset in order to detect overfitting and to optimize model choice and training parameters. Using a separate dataset for those purposes is necessary because the model's fit to the training data does not provide information about its generalization capability. In other words, a trained model may accurately reproduce the training data, but perform poorly on images that have not been presented to it before. However, as soon as any optimizing decision has been made based on the validation data (e.g. when to stop the training process), the model may again show overly optimistic performance for this particular dataset. To obtain an unbiased evaluation of the final model, it is therefore necessary to provide an additional test dataset, which should not have been used for any other task beforehand. The *ginjinn split* command partitions an input dataset in such a way that each image along with its annotated objects is assigned to one of the resulting subsets. To be representative for the original dataset, each of the latter should comprise similar proportions of objects from each category. Aiming at a high level of homogeneity, the proposed splits are generated by a greedy optimization algorithm (see Appendix S1). Despite being a relatively rough heuristic, this approach is often sufficient to create acceptable splits and can even be applied to large datasets.

2.3 | Object detection and instance segmentation

GinJinn2, by leveraging Detectron2's model zoo, offers several Faster R-CNN (Ren et al., 2015) and Mask R-CNN (He et al., 2017) models for bounding-box detection and instance segmentation respectively. These are used in a supervised manner, that is, before being able to predict objects on new images in a meaningful way, their parameters ('weights') have to be fitted to images with known object occurrences ('training'). While training such models de novo can be highly GPU intensive, this process can be considerably abbreviated by starting from pretrained rather than randomly initialized weights ('transfer learning'). Accordingly, all available Detectron2 models have already been trained on a large image dataset. Using those pretrained networks reduces the training time for new, custom datasets as well.

Once the user has prepared datasets for training, and, optionally, validation and test (see Dataset splitting), a GinJinn2 project can be initialized using *ginjinn new*. Training models using *ginjinn train* constitutes the computationally most demanding part of a typical GinJinn2 pipeline. This process consists of a prespecified number of iterations, at each of which one or multiple images from the training dataset are presented to the model. The objects predicted by the latter are then compared to the known annotations and the model weights are adjusted to reduce deviations ('loss') from the desired output. While minimizing the loss with respect to the training dataset, at some point, the model's generalization capability may begin to degrade. This so-called overfitting can be recognized by an increasing loss for the validation dataset. The latter is therefore evaluated at predefined intervals. To enable a better assessment of the learning progress, COCO (Lin et al., 2014) evaluation metrics (AP, AP50, AP75, APs, APm and APl; for details see https://cocodataset.org) for the validation dataset are calculated as well. Since the model weights are stored periodically, in case of overfitting, the user can go back to an earlier checkpoint without having to discard the complete training. Since GinJinn2 is using Detectron2 as modelling backend, all models that are trained in the context of a GinJinn2 project can be used with Detectron2's Python interface without modification.

The quality of the final, trained model is best assessed based on a hitherto unused dataset with known object occurrences. This can be done using *ginjinn evaluate*, which calculates COCO evaluation metrics for the specified test dataset.

The *ginjinn predict* command allows applying a trained model to predict object occurrences for arbitrary images. Instance segmentations can optionally be refined using CascadePSP (Cheng et al., 2020); while slowing down the predictions, this may considerably improve the quality of the object outlines, especially in case of clear object boundaries. To facilitate the further use of the predictions, GinJinn2 provides various output options: (a) visualization of the predictions on the original images, (b) writing a new COCO annotation file and (c) saving a cropped image and, if applicable, segmentation mask for each predicted object.

2.4 | Further functionality

GinJinn2 offers several utilities for data pre- and postprocessing

As a counterpart to the already described splitting command (*ginjinn split*), datasets can also be merged (*ginjinn utils merge*), which is particularly useful when using COCO's annotation format. In doing so, the input datasets are also checked for duplicated images.

Object annotations can be filtered by either category or size using *ginjinn utils filter_cat* or *ginjinn utils filter_size* respectively. The latter command is also capable of removing only small disjunct fragments from existing objects.

To simplify existing data, nested image directories can be summarized, making them compatible with GinJinn2 and other tools. *ginjinn utils flatten* recursively collects all images from a given directory and its sub-directories, renames and copies them into a single directory, and modifies associated annotations accordingly.

Due to the limited spatial resolution of common object detection models, detecting or segmenting objects that are small in relation to the image size can be difficult. To mitigate this problem, a sliding window approach can be used to split the original images into smaller sub-images (*ginjinn utils sw_split*), preserving annotated objects, if available. Conversely, predictions based on such fragmented images can be merged again (*ginjinn utils sw_merge*) in order to generate an annotation of the original image.

The *ginjinn utils crop* command creates an annotated sub-image for each annotated object from a given dataset. Similar to the sliding window approach, this can be utilized to increase objects sizes relative to the images. Specifically, performing instance segmentation based on previously cropped bounding boxes may lead to improved results.

Besides the aforementioned data processing features, the following commands aim to provide additional convenience

The contents of a dataset can be briefly summarized using *ginjinn info*. More detailed information is provided by *ginjinn utils count*, which lists object occurrences individually for each image in a given dataset. Object annotations can be visualized with *ginjinn visualize*, which produces images overlaid by bounding boxes and, if available, segmentation polygons. Moreover, Ginjinn2 allows to generate artificial datasets for testing purposes (*ginjinn simulate*).

2.5 | Installation and usage

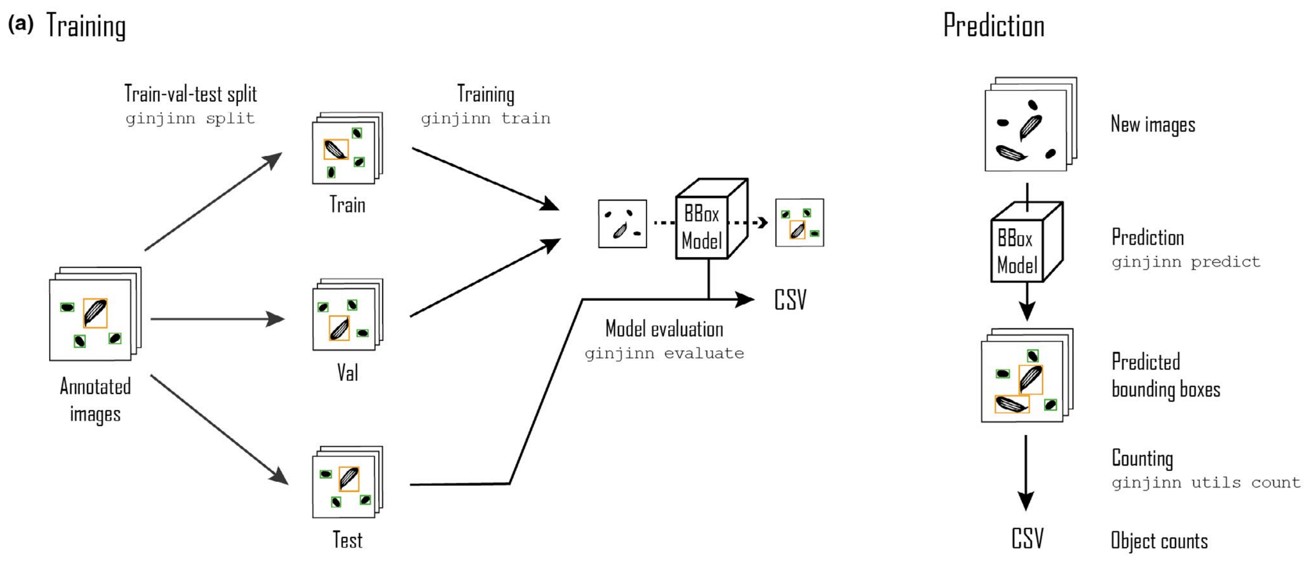
GinJinn2 is implemented in Python3 and can be installed using the Conda package manager, which also takes care of most of its dependencies. *ginjinn* and all its subcommands provide a help option to list available parameters along with a short description. Further guidelines regarding installation and usage, along with an introductory tutorial and exemplary applications, are provided at https://ginjinn2.readthedocs.io.

3 | EXAMPLE ANALYSES

3.1 | Seed counting

In this section, we demonstrate how GinJinn2 can be applied for seed mixture analysis, an illustrative use case for bounding-box detection with subsequent counting. This approach could, for instance, be used to examine commercial seed mixtures or be applied to ecological samples (e.g. from seed traps). The presented analysis is based on a dataset consisting of 284 microscopic images of sand-contaminated seed mixtures of the two plant genera *Sedum* L. and *Arabidopsis* (DC.) Heynh.

For all images, intact seeds were annotated with bounding boxes using the Computer Vision Annotation Tool (CVAT, https://github.com/openvinotoolkit/cvat), resulting in 6,732 and 1,964 annotated seeds for *Arabidopsis* and *Sedum* respectively. The annotated images were exported as COCO dataset, which was then flattened (*ginjinn utils flatten*), and split into sub-datasets for training, validation and testing. A Faster R-CNN model was simultaneously trained and validated (Figure 1a). The quality of the fit model was assessed using COCO evaluation metrics for bounding-box detection. In addition, instances predicted for the test dataset were



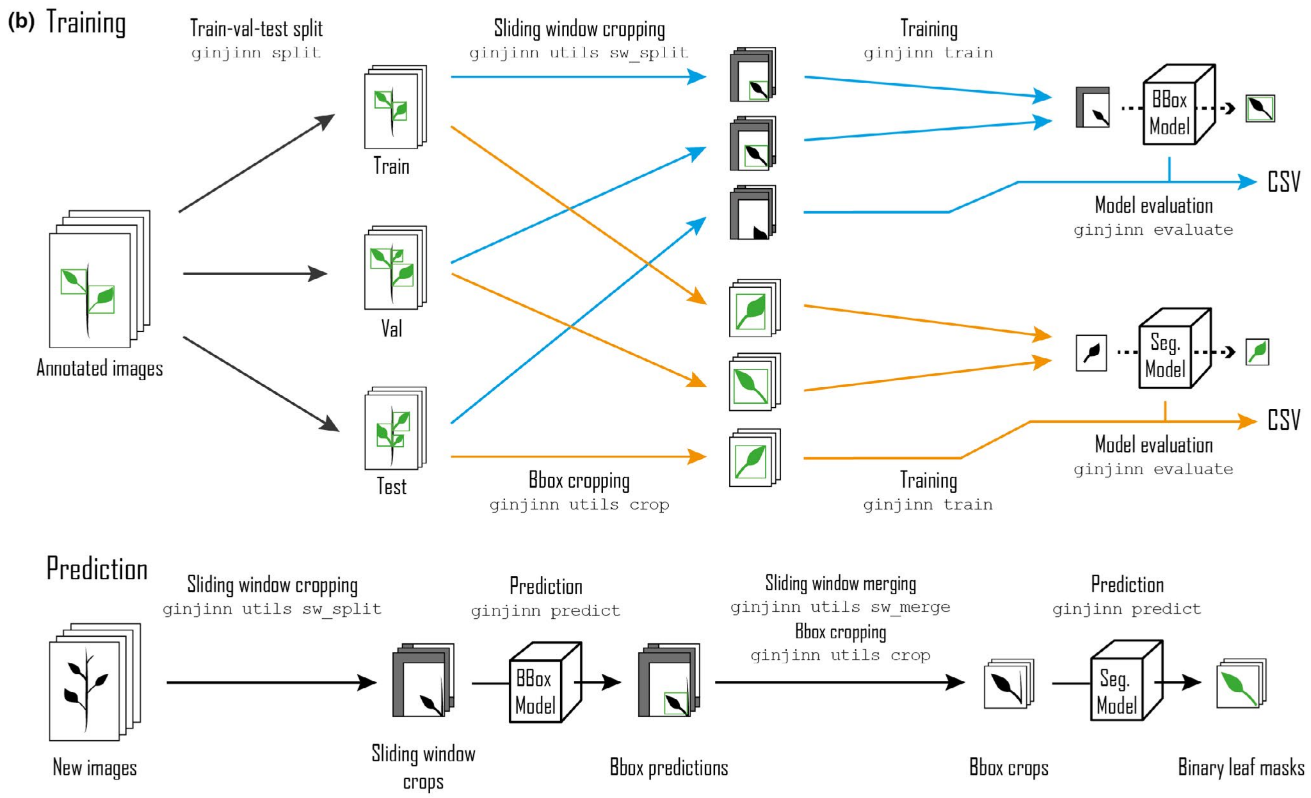


FIGURE 1 Seeds (a) and *Leucanthemum* (b) analysis workflows. The seeds dataset is split into training, validation and test datasets, which are used to train and evaluate a bounding-box model (a, Training). The trained model is applied to new data for seed counting (a, Prediction). The *Leucanthemum* dataset is also split into training, validation and test datasets, but the workflow comprises training and evaluation of two separate models (b, Training). The blue branch refers to a bounding-box model for the detection of leaves on sliding window crops of the split dataset. The orange branch depicts the training and evaluation of an instance segmentation model on padded bounding boxes cropped from the split datasets. Leaf segmentations for new data are predicted by combining both models (b, Prediction)

counted (*ginjinn utils count*) and compared with the manually obtained counts.

After training, the AP50 was 98.6 and 98.9 for the validation and test dataset, respectively, which indicates that no overfitting occurred. The mean absolute error (MAE) of the class counts for the training dataset was 0.77 for *Arabidopsis* and 0.58 for *Sedum*, meaning that on average, less than a single object per image was misclassified, missed or falsely detected. The MAE of the seed proportions was 0.01, that is, only 1% deviation from the true seed proportions. Exemplary predictions are shown in Figure S1a (Appendix S2).

3.2 | Yellow-sticky-traps insect detection and counting

As an example project for counting small, low-contrast objects on large images, the yellow-sticky-traps dataset (Nieuwenhuizen et al., 2018) was analysed. This dataset consists of images of yellow glue traps that were placed in greenhouses to monitor insect abundance. Three categories of insects (true bugs) were annotated with bounding boxes: Whitefly (WF), *Macrolophus* (MR) and *Nesidiocoris* (NC).

After removing redundant images and correcting erroneous or missing annotations using CVAT, a cleaned sub-dataset comprising 120 images along with 4,913 bounding-box annotations (WF: 3,660, MR: 1,069, NC: 184) was exported in COCO format. In contrast to the seeds dataset, these bounding-box annotations are of considerably lower quality, often enclosing the insects only loosely.

The cleaned dataset was split into training, validation and test datasets using *ginjinn split*. Since the insects are relatively small compared to the total image size, a sliding window approach was applied (*ginjinn utils sw_split*) to crop sub-images along with corresponding object (sub-)annotations. The cropped datasets were used to train and evaluate a Faster R-CNN model for bounding-box detection. Finally, object instances predicted for the test dataset were counted (*ginjinn untils count*) and compared with true object counts.

The trained model achieved a validation and test AP50 of 90.12 and 92.4 respectively. The mean absolute error (MAE) of the instance counts was 1.67 for WF, 0.21 for NC and 0.79 for MR at an average of 27.1, 1.67 and 7.41 annotated instances per image for the respective object categories. The former amounts to a relative counting error of 6% for WF, 12.5% for NC and 10.6% for MR (weighted average: 7.24%). Exemplary predictions are illustrated in Figure S1b (Appendix S2).

3.3 | Stomata segmentation

To demonstrate basic instance segmentation with the aim of detecting stomata, we applied GinJinn2 to microscopic images of epidermal plant material, retrieved from the Cuticle Database Project (Barclay et al., 2012). Results of such a segmentation can be used in downstream analyses for counting, measuring density or examining size and shape of the stomata.

Using CVAT, 147 images were annotated with 2,314 polygons, each enclosing the guard cells of a stoma. The annotated images were exported as COCO dataset and split into training, validation and test datasets used to train and evaluate a Mask R-CNN model.

The trained model achieved an AP of 49.46 and 51.32 for the validation and test dataset respectively. The mean absolute counting error amounts to 2.34 at an average of 14.69 stomata per image. An exemplary prediction is shown in Figure 2a.

3.4 | *Leucanthemum* leaf segmentation

Morphometric studies often rely on outline data of specific animal or plant organs like, for example, leaves in the latter organism group. A common workflow to generate such data is to manually remove leaves from a living or herborized plant, fixate them on a contrasting surface, capture digital images and finally apply semi-automatic thresholding methods (e.g. OTSU-thresholding) to construct binary segmentation masks. In this exemplary application of GinJinn2, we show an alternative way to segment individual leaves from digitized herbarium specimens based on a two-step approach involving separate models for bounding-box detection and segmentation.

For this purpose, the Botanic Garden and Botanical Museum Berlin provided us with 303 digitized herbarium specimens from 12 different *Leucanthemum* Mill. (ox-eye daisy) species. Using CVAT, the specimen images were annotated with polygons of the single object category 'leaf'. This category represents largely intact leaves, which are a prerequisite for reliable morphometric analyses. The annotated images, comprising 950 'leaf' instances, were exported from CVAT as COCO dataset, flattened (*ginjinn utils flatten*) and split into training, validation and test datasets.

A two-step pipeline (Figure 1b) was applied, consisting of (a) a Faster R-CNN bounding-box detection model that allows to extract individual leaves, and (b) a Mask R-CNN model to segment the leaves on those image parts. The Faster R-CNN was trained and evaluated on sliding window crops (*ginjinn utils sw_split*) of the three datasets. For the Mask R-CNN, sub-images (*ginjinn utils crop*) were cropped from the original annotated images, each containing a single annotated leaf. Based on those cropped datasets, the Mask R-CNN was trained and evaluated. In addition, segmentation refinement was applied to the predictions for the test dataset.

After training, the Faster R-CNN achieved an AP of 30.57 and 25.85 for the validation and test dataset respectively. The Mask R-CNN's AP scores were 76.44 and 74.54. Figure 2b illustrates an exemplary prediction. For new image data, the complete prediction process also involves sliding window merging as illustrated in Figure 1b in order to remove duplicated objects.

4 | DISCUSSION

The GinJinn2 toolkit advances the original GinJinn by reimplementing its ideas on the basis of Detectron2, while also introducing new

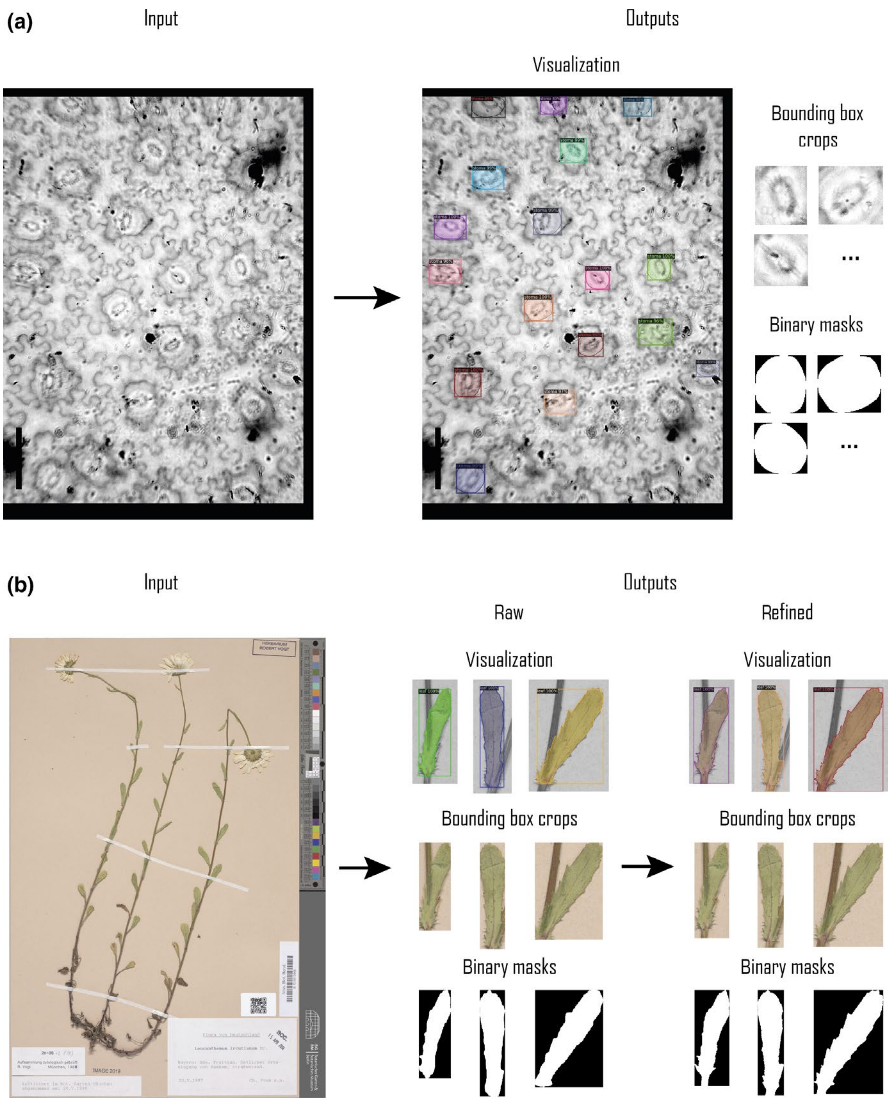


FIGURE 2 Exemplary outputs from the Stomata (a) and *Leucanthemum* (b) analyses. (a) depicts a single input image along with corresponding predictions by the stomata model, showing different output formats. Similarly, (b) shows an input image and corresponding predictions for the *Leucanthemum* pipeline, before and after segmentation refinement

features like segmentation models including mask refinement, as well as several data pre- and postprocessing capabilities.

Based on four exemplary datasets we have shown applications of varying complexity. The seeds and yellow-sticky-traps analyses address multi-category object counting problems using bounding-box detection. We were able to predict the seed ratios with an absolute error of only 1%, proving the potential of our software for the automation of such counting tasks. Considering the similar problem of counting insects on yellow glue traps, with an error of 7.2%, the accuracy of the trained model may appear less convincing. There are two likely causes for this difference in accuracy: (a) low contrast between objects (insects) and background (glue trap) and (b) low quality of annotations. The latter could easily be solved by a more careful annotation scheme. Nevertheless, the achieved accuracy might be sufficient for practical applications, for example, to measure the response to insecticide treatments or released beneficials in greenhouses.

The stomata analysis serves as a basic example of instance segmentation. Despite several previous works on the automated examination of stomata (Carrasco et al., 2020; Casado-García et al., 2020; Fetter et al., 2019; Li et al., 2019; Meeus et al., 2020; Song et al., 2020; Toda et al., 2018), this contribution, to our knowledge, is the first trying to automatically segment whole stomata (represented by their guard cells) using deep learning. With only 88 highly variable training images, our model achieved an AP of 51.32. Depending on the intended downstream analyses, this precision may already be acceptable if, for instance, only few high-quality object instances are required. Undoubtedly, a model trained on a larger dataset will achieve substantially higher predictive power.

Finally, the *Leucanthemum* analysis illustrates how to construct a pipeline consisting of sliding window-based bounding-box detection and subsequent segmentation for the extraction of high-quality leaf silhouettes from herbarium specimens. Here, the Faster R-CNN achieved an AP of 25.85. For potential morphometric analyses, we are not interested in extracting all leaves, but only largely intact ones, even at the cost of discarding viable instances. Therefore, the relatively low AP is sufficient. The Mask R-CNN, with an AP of 74.54 before refinement, was very successful at segmenting the leaves inside the bounding boxes. This pipeline already allows to generate leaf outlines for downstream analyses like Elliptic Fourier Analysis or Leaf Dissection Index calculation (for an overview of such methods, see McLellan & Endler, 1998) with little manual effort.

With the presented exemplary analyses, we hope to provide guidance for the application of GinJinn2 for automatic data collection and feature extraction. Despite GinJinn2's progress compared to its predecessor, there is still room for further improvements. At the moment, GinJinn2 is only available for Unix-like operating systems with access to an NVidia GPU while Windows support may become available with forthcoming updates to the Windows Subsystem for Linux (WSL). Moreover, there is only one meta-architecture for each of the two detection tasks available, namely Faster R-CNN and Mask R-CNN. These, however, are among the most successful architectures for general-purpose object detection and segmentation. The integration of additional model architectures may be part of future versions.

We are confident that GinJinn2 will enable users, even those without programming experience, to apply deep learning-based methods for object detection and segmentation as part of their analysis pipelines. Advanced users may utilize GinJinn2 as a tool for rapid prototyping.

ACKNOWLEDGEMENTS

First and foremost, we thank Christoph Oberprieler (Regensburg), who acquired the funding, for enabling this work and his comments on the manuscript. We thank Robert Vogt (Berlin) and Sergey Rosbakh (Regensburg) for providing digital images of *Leucanthemum* specimens and seed mixtures respectively. We also thank David Dilcher (Bloomington) for granting us permission to use the microscopic image shown in Figure 2a. The help of Maximilian Schall and Sebastian Segieth (both Regensburg), who annotated many of the *Leucanthemum* and seeds images, is much appreciated. We thank Tanja Wenzel for her support in designing the workflow diagrams. We also thank Agnes Scheunert (Regensburg) for her comments on the manuscript. The comments of Sam Church and two anonymous reviewers, who helped to significantly improve this contribution, are much appreciated. This work was supported by a Grant (OB 155/13-1) of the German Research Foundation (DFG) in the frame of the Priority Programme SPP 1991 'Taxon-omics—New Approaches for Discovering and Naming Biodiversity' to Christoph Oberprieler. Open Access funding enabled and organized by Projekt DEAL.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHORS' CONTRIBUTIONS

T.O. and U.L. envisioned the present work, implemented the software, carried out the analyses and wrote the manuscript. Both authors approved the final version of the manuscript. We further note that U.L. and T.O. contributed equally to this work. The order of their names in the author list was decided by coin toss.

DATA AVAILABILITY STATEMENT

GinJinn2's source code and manual are freely available at GitHub (https://github.com/AGOberprieler/GinJinn2) and archived at Zenodo (Lautenschlager & Ott, 2021). For all presented example applications, the used GinJinn2 commands and project configuration files are deposited at Zenodo as well (https://doi.org/10.5281/zenodo.5747582). The annotated Seeds, Yellow-sticky-traps and *Leucanthemum* datasets are hosted by the German Federation for Biological Data (GFBio; https://doi.org/10.34656/k53b-9c27.1, https://doi.org/10.34656/41pk-rn18.1 and https://doi.org/10.34656/skvz-cs62.1). The images used for the Stomata analysis are hosted by the Cuticle Database (Barclay et al., 2012); the corresponding annotations (https://doi.org/10.5281/zenodo.5718606) and a Python3 script (https://doi.org/10.5281/zenodo.5747611) to prepare the images from the Cuticle Database are archived at Zenodo.

ORCID

Tankred Ott https://orcid.org/0000-0001-6748-0510
Ulrich Lautenschlager https://orcid.org/0000-0003-1886-2277

REFERENCES

Afonso, M., Fonteijn, H., Fiorentin, F. S., Lensink, D., Mooij, M., Faber, N., Polder, G., & Wehrens, R. (2020). Tomato fruit detection and counting in greenhouses using deep learning. *Frontiers in Plant Science*, *11*, 1759. https://doi.org/10.3389/fpls.2020.571299

Barclay, R. S., Wilf, P., Dilcher, D. L., & McElwain, J. C. (2012). *The cuticle database project. The Earth and Environmental Systems Institute of Pennsylvania State University*. Retrieved from http://cuticledb.eesi.psu.edu

Buddha, K., Nelson, H., Zermas, D., & Papanikolopoulos, N. (2019). Weed detection and classification in high altitude aerial images for robot-based precision agriculture. *2019 27th Mediterranean Conference on Control and Automation (MED)*, 280–285. https://doi.org/10.1109/MED.2019.8798582

Carrasco, M., Toledo, P. A., Velázquez, R., & Bruno, O. M. (2020). Automatic stomatal segmentation based on Delaunay-Rayleigh frequency distance. *Plants*, *9*(11), 1613. https://doi.org/10.3390/plants9111613

Casado-García, A., del-Canto, A., Sanz-Saez, A., Pérez-López, U., Bilbao-Kareaga, A., Fritschi, F. B., Miranda-Apodaca, J., Muñoz-Rueda, A., Sillero-Martínez, A., Yoldi-Achalandabaso, A., Lacuesta, M., & Heras, J. (2020). LabelStoma: A tool for stomata detection based on the YOLO algorithm. *Computers and Electronics in Agriculture*, *178*, 105751. https://doi.org/10.1016/j.compag.2020.105751

Cheng, H. K., Chung, J., Tai, Y.-W., & Tang, C.-K. (2020). CascadePSP: Toward class-agnostic and very high-resolution segmentation via global and local refinement. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8887–8896. https://doi.org/10.1109/CVPR42600.2020.00891

Fetter, K. C., Eberhardt, S., Barclay, R. S., Wing, S., & Keller, S. R. (2019). StomataCounter: A neural network for automatic stomata identification and counting. *New Phytologist*, *223*(3), 1671–1681. https://doi.org/10.1111/nph.15892

Gómez-de-Mariscal, E., García-López-de-Haro, C., Ouyang, W., Donati, L., Lundberg, E., Unser, M., Muñoz-Barrutia, A., & Sage, D. (2021). DeepImageJ: A user-friendly environment to run deep learning models in ImageJ. *Nature Methods*, *18*(10), 1192–1195. https://doi.org/10.1038/s41592-021-01262-9

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *IEEE International Conference on Computer Vision (ICCV)*, 2980–2988. https://doi.org/10.1109/ICCV.2017.322

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. (2017). Speed/accuracy trade-offs for modern convolutional object detectors.

2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/CVPR.2017.351

Jones, H. G. (2020). What plant is that? Tests of automated image recognition apps for plant identification on plants from the British flora. *AoB PLANTS*, *12*(6), plaa052. https://doi.org/10.1093/aobpla/plaa052

Lautenschlager, U., & Ott, T. (2021). AGOberprieler/GinJinn2: (v1.0.2). *Zenodo*, https://doi.org/10.5281/zenodo.5718521

Li, K., Huang, J., Song, W., Wang, J., Lv, S., & Wang, X. (2019). Automatic segmentation and measurement methods of living stomata of plants based on the CV model. *Plant Methods*, *15*(1), 67. https://doi.org/10.1186/s13007-019-0453-5

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740–755). Springer International Publishing. https://doi.org/10.1007/978-3-319-10602-1_48

McLellan, T., & Endler, J. A. (1998). The relative success of some methods for measuring and describing the shape of complex objects. *Systematic Biology*, *47*(2), 264–281. https://doi.org/10.1080/106351598260914

Meeus, S., Van den Bulcke, J., & Wyffels, F. (2020). From leaf to label: A robust automated workflow for stomata detection. *Ecology and Evolution*, *10*(17), 9178–9191. https://doi.org/10.1002/ece3.6571

Nieuwenhuizen, A. T., Hemming, J., & Suh, H. K. (2018). Detection and classification of insects on stick-traps in a tomato crop using Faster R-CNN. *The Netherlands Conference on Computer Vision*. Retrieved from https://edepot.wur.nl/463457

Norouzzadeh, M. S., Morris, D., Beery, S., Joshi, N., Jojic, N., & Clune, J. (2021). A deep active learning system for species identification and counting in camera trap images. *Methods in Ecology and Evolution*, *12*(1), 150–161. https://doi.org/10.1111/2041-210X.13504

Ott, T., Palm, C., Vogt, R., & Oberprieler, C. (2020). GinJinn: An object-detection pipeline for automated feature extraction from herbarium specimens. *Applications in Plant Sciences*, *8*(6), e11351. https://doi.org/10.1002/aps3.11351

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28 (NIPS 2015)*. Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Schneider, C. A., Rasband, W. S., & Eliceiri, K. W. (2012). NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, *9*(7), 671–675. https://doi.org/10.1038/nmeth.2089

Schroeder, A. B., Dobson, E. T. A., Rueden, C. T., Tomancak, P., Jug, F., & Eliceiri, K. W. (2020). The ImageJ ecosystem: Open-source software for image visualization, processing, and analysis. *Protein Science*, *30*(1), 234–249. https://doi.org/10.1002/pro.3993

Song, W., Li, J., Li, K., Chen, J., & Huang, J. (2020). An automatic method for stomatal pore detection and measurement in microscope images of plant leaf based on a convolutional neural network model. *Forests*, *11*(9), 954. https://doi.org/10.3390/f11090954

Toda, Y., Toh, S., Bourdais, G., Robatzek, S., Maclean, D., & Kinoshita, T. (2018). DeepStomata: Facial recognition technology for automated stomatal aperture measurement. *BioRxiv*, 365098. https://doi.org/10.1101/365098 [preprint]

Wäldchen, J., & Mäder, P. (2018). Machine learning for image based species identification. *Methods in Ecology and Evolution*, *9*(11), 2216–2225. https://doi.org/10.1111/2041-210X.13075

Weaver, W. N., Ng, J., & Laport, R. G. (2020). LeafMachine: Using machine learning to automate leaf trait extraction from digitized herbarium specimens. *Applications in Plant Sciences*, *8*(6), e11367. https://doi.org/10.1002/aps3.11367

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). *Detectron2*. Retrieved from https://github.com/facebookresearch/detectron2

Younis, S., Schmidt, M., Weiland, C., Dressler, S., Seeger, B., & Hickler, T. (2020). Detection and annotation of plant organs from digitised herbarium scans using deep learning. *Biodiversity Data Journal*, *8*, e57090. https://doi.org/10.3897/BDJ.8.e57090

SUPPORTING INFORMATION

Additional supporting information may be found in the online version of the article at the publisher's website.

How to cite this article: Ott, T., & Lautenschlager, U. (2022). GinJinn2: Object detection and segmentation for ecology and evolution. *Methods in Ecology and Evolution*, 13, 603–610. https://doi.org/10.1111/2041-210X.13787