

 Open access • Posted Content • DOI:10.1101/2021.08.20.457033

## **GinJinn2: Object detection and segmentation for ecology and evolution**

— [Source link](#) 

Tankred Ott, Ulrich Lautenschlager

**Institutions:** University of Regensburg

**Published on:** 20 Aug 2021 - bioRxiv (Cold Spring Harbor Laboratory)

**Topics:** Object detection and Feature extraction

Related papers:

- [The K-Space segmentation tool set](#)
- [A software framework for preprocessing and level set segmentation of medical image data](#)
- [A Review of Medical Image Segmentation: Methods and Available Software](#)
- [A Survey on Deep Learning Methods for Semantic Image Segmentation in Real-Time.](#)
- [PaddleSeg: A High-Efficient Development Toolkit for Image Segmentation.](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/ginjinn2-object-detection-and-segmentation-for-ecology-and-3ve2ot427p>

1 **GinJinn2: Object detection and segmentation for ecology and evolution**

2

3 **Authors**

4 Tankred Ott<sup>†,1</sup> & Ulrich Lautenschlager<sup>†,1</sup>

5

6 <sup>†</sup>These authors have contributed equally to this work and share first authorship. The  
7 order of their names in the author list was decided by coin toss.

8

9 <sup>1</sup>Evolutionary and Systematic Botany Group, Institute of Plant Biology, University of  
10 Regensburg, Universitätsstr. 31, D-93053 Regensburg, Germany.

11

12 **Corresponding author**

13 Ulrich Lautenschlager

14 Evolutionary and Systematic Botany Group

15 Institute of Plant Sciences

16 University of Regensburg

17 Universitätsstr. 31

18 D-93053 Regensburg, Germany

19 E-mail: [ulrich.lautenschlager@ur.de](mailto:ulrich.lautenschlager@ur.de)

20

21 **Running headline**

22 GinJinn2: Object detection and segmentation

23 **Abstract**

24 1. Proper collection and preparation of empirical data still represent one of the  
25 most important, but also expensive steps in ecological and  
26 evolutionary/systematic research. Modern machine learning approaches,  
27 however, have the potential to automate a variety of tasks, which until recently  
28 could only be performed manually. Unfortunately, the application of such  
29 methods by researchers outside the field is hampered by technical difficulties,  
30 some of which, we believe, can be avoided.

31 2. Here, we present GinJinn2, a user-friendly toolbox for deep learning-based  
32 object detection and instance segmentation on image data. Besides providing  
33 a convenient command-line interface to existing software libraries, it  
34 comprises several additional tools for data handling, pre- and postprocessing,  
35 and building advanced analysis pipelines.

36 3. We demonstrate the application of GinJinn2 for biological purposes using four  
37 exemplary analyses, namely the evaluation of seed mixtures, detection of  
38 insects on glue traps, segmentation of stomata, and extraction of leaf  
39 silhouettes from herbarium specimens.

40 4. GinJinn2 will enable users with a primary background in biology to apply deep  
41 learning-based methods for object detection and segmentation in order to  
42 automate feature extraction from image data.

43

44 **Keywords**

45 Automation, Computer Vision, Deep Learning, Instance Segmentation, Machine  
46 Learning, Morphometrics, Object Detection, Trait Extraction

## 47 **Introduction**

48 Conducting empirical studies in ecology and evolutionary/systematic biology typically  
49 requires substantial effort for proper data collection and preparation. The ability to  
50 automate time-consuming steps of this process, possibly along with further  
51 downstream analyses, for example, using programming languages like Python or R,  
52 can not only increase productivity, but also allow otherwise infeasible large-scale  
53 analyses. Recent advances in machine learning (ML), both on the soft- and  
54 hardware side, make it even possible to automate tasks that are difficult to solve by  
55 means of classically designed algorithms. Computer vision, in particular, has largely  
56 profited from deep learning, which increasingly influences even the more traditional  
57 branches of organismic biology. Species identification tools running on smartphone  
58 devices (for an overview, see Wäldchen & Mäder, 2018; Jones, 2020) are prominent  
59 examples for this trend. Beyond pure classification tasks, a technically even more  
60 challenging problem consists in localizing objects like cells, organs, or individuals on  
61 images. Specialized tools address this problem for various areas of application, such  
62 as crop or weed detection (e.g., Buddha et al., 2019; Afonso et al., 2020), detection  
63 of leaves and other plant organs on herbarium specimens (e.g., Ott et al., 2020;  
64 Weaver et al., 2020; Younis et al., 2020), stomata counting using microscopic leaf  
65 images (e.g., Fetter et al., 2019), animal counting using camera traps (Norouzzadeh  
66 et al., 2021), and many more.

67 Despite the availability of increasingly convenient frameworks, adapting well-  
68 established ML methods to new areas of application typically requires an amount of  
69 technical knowledge that may discourage potential users. GinJinn2, whose core  
70 functionality is based on Detectron2 (Wu et al., 2019), aims at lowering this hurdle by  
71 providing an easy-to-use command-line interface to the latter, augmented by a  
72 number of utility functions, designed to help the user with building custom analysis  
73 pipelines. While its predecessor (Ott et al., 2020) focussed on extracting leaves from

74 digitized herbarium specimens, the presented program is a complete  
75 reimplementation, aiming at a wider scope of application. Unlike the former, it is not  
76 restricted to bounding-box object detection, but also incorporates functionality for  
77 instance segmentation, i.e., pixel-precise detection and classification of individual  
78 objects.

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

88

## 89 **Software**

### 90 **Overview**

91 GinJinn2 is a toolbox for deep learning-based bounding-box object detection and  
92 instance segmentation. As such, it provides functionality for model training,  
93 evaluation and application based on the Detectron2 framework, segmentation  
94 refinement based on CascadePSP (Cheng et al., 2020), a set of data pre- and  
95 postprocessing tools for handling annotated image datasets, and capabilities for data  
96 insight and visualization. GinJinn2 is not meant to be a replacement for existing  
97 frameworks like Detectron2 or the Tensorflow Object Detection API (Huang et al.,  
98 2017), but rather a toolkit enabling code-free access to deep learning-based object  
99 detection technologies. All of GinJinn2's functionality is accessible via an easy-to-use

100 command-line interface (CLI).

101

## 102 **Dataset splitting**

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

123

## 124 **Object detection and instance segmentation**

125 GinJinn2, by leveraging Detectron2's model zoo, offers several Faster R-CNN (Ren  
126 et al., 2015) and Mask R-CNN (He et al., 2017) models for bounding-box detection

127 and instance segmentation, respectively. These are used in a supervised manner,  
128 i.e., before being able to predict objects on new images in a meaningful way, their  
129 parameters (“weights”) have to be fitted to images with known object occurrences  
130 (“training”). While training such models *de novo* can be highly GPU-intensive, this  
131 process can be considerably abbreviated by starting from pretrained rather than  
132 randomly initialized weights (“transfer learning”). Accordingly, all available Detectron2  
133 models have already been trained on a large image dataset. Using those pretrained  
134 networks reduces the training time for new, custom datasets as well.

135

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

153 The quality of the final, trained model is best assessed based on a hitherto unused

154 dataset with known object occurrences. This can be done using *ginjinn evaluate*,  
155 which calculates COCO evaluation metrics for the specified test dataset.  
156 The *ginjinn predict* command allows applying a trained model to predict object  
157 occurrences for arbitrary images. Instance segmentations can optionally be refined  
158 using CascadePSP (Cheng et al., 2020); while slowing down the predictions, this  
159 may considerably improve the quality of the object outlines, especially in case of  
160 clear object boundaries. To facilitate the further use of the predictions, GinJinn2  
161 provides various output options: 1) visualization of the predictions on the original  
162 images, 2) writing a new COCO annotation file, and 3) saving a cropped image and,  
163 if applicable, segmentation mask for each predicted object.

164

### 165 **Further functionality**

166 GinJinn2 offers several utilities for data pre- and postprocessing:

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

171 Object annotations can be filtered by either category or size using *ginjinn utils*  
172 *filter\_cat* or *ginjinn utils filter\_size*, respectively. The latter command is also capable  
173 of removing only small disjunct fragments from existing objects.

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

178 Due to the limited spatial resolution of common object detection models, detecting or  
179 segmenting objects that are small in relation to the image size can be difficult. To



180 mitigate this problem, a sliding-window approach can be used to split the original  
181 images into smaller sub-images (*ginjinn utils sw\_split*), preserving annotated objects,  
182 if available. Conversely, predictions based on such fragmented images can be  
183 merged again (*ginjinn utils sw\_merge*) in order to generate an annotation of the  
184 original image.

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

190

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

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

199

## 200 **Installation and usage**

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

206 <https://ginjinn2.readthedocs.io>.

207

## 208 **Example analyses**

### 209 **Seed counting**

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

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

227 After training, the AP50 was 98.6 and 98.90 for the validation and test dataset,  
228 respectively, which indicates that no overfitting occurred. The mean absolute error  
229 (MAE) of the class counts for the training dataset was 0.77 for *Arabidopsis* and 0.58  
230 for *Sedum*, meaning that on average, less than a single object per image was  
231 misclassified, missed, or falsely detected. The MAE of the seed proportions was  
232 0.01, i.e., only one percent deviation from the true seed proportions.

233

## 234 **Yellow-sticky-traps insect detection and counting**

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

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

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

252 The trained model achieved a validation and test AP50 of 90.12 and 92.4,  
253 respectively. The mean absolute error (MAE) of the instance counts was 1.67 for  
254 WF, 0.21 for NC, and 0.79 for MR at an average of 27.1, 1.67, and 7.41 annotated  
255 instances per image for the respective object categories. The former amounts to a  
256 relative counting error of 6% for WF, 12.5% for NC, and 10.6% for MR (weighted  
257 average: 7.24%).

258

## 259 **Stomata segmentation**

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

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

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

272

### 273 ***Leucanthemum* leaf segmentation**

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

283 For this purpose, the Botanic Garden and Botanical Museum Berlin provided us with  
284 303 digitized herbarium specimens from 12 different *Leucanthemum* Mill. (ox-eye  
285 daisy) species. Using CVAT, the specimen images were annotated with polygons of  
286 the single object category “leaf”. This category represents largely intact leaves, which

287 are a prerequisite for reliable morphometric analyses. The annotated images,  
288 comprising 950 “leaf” instances, were exported from CVAT as COCO dataset,  
289 flattened (*ginjinn utils flatten*) and split into training, validation, and test datasets.

290 A two-step pipeline (Figure 1B) was applied, consisting of 1) a Faster R-CNN  
291 bounding-box detection model that allows to extract individual leaves, and 2) a Mask  
292 R-CNN model to segment the leaves on those image parts. The Faster R-CNN was  
293 trained and evaluated on sliding-window crops (*ginjinn utils sw\_split*) of the three  
294 datasets. For the Mask R-CNN, sub-images (*ginjinn utils crop*) were cropped from  
295 the original annotated images, each containing a single annotated leaf. Based on  
296 those cropped datasets, the Mask R-CNN was trained and evaluated. In addition,  
297 segmentation refinement was applied to the predictions for the test dataset.

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

303

304 All used GinJinn2 commands and the corresponding project configuration files can  
305 be found in the Supporting Information (S1-S6).

306

## 307 **Discussion**

308 The GinJinn2 framework advances the original GinJinn by reimplementing its ideas  
309 on the basis of Detectron2, while also introducing new features like segmentation  
310 models including mask refinement, as well as several data pre- and postprocessing  
311 capabilities.

312 Based on four exemplary datasets we have shown applications of varying  
313 complexity. The seeds and yellow-sticky-traps analyses address multi-category

314 object counting problems using bounding-box detection. We were able to predict the  
315 seed ratios with an absolute error of only 1%, proving the potential of our software for  
316 the automation of such counting tasks. Considering the similar problem of counting  
317 insects on yellow glue traps, with an error of 7.2%, the accuracy of the trained model  
318 may appear less convincing. There are two likely causes for this difference in  
319 accuracy: 1) low contrast between objects (insects) and background (glue trap) and  
320 2) low quality of annotations. The latter could easily be solved by a more careful  
321 annotation scheme. Nevertheless, the achieved accuracy might be sufficient for  
322 practical applications, e.g., to measure the response to insecticide treatments or  
323 released beneficials in greenhouses.

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

334 Finally, the *Leucanthemum* analysis illustrates how to construct a pipeline consisting  
335 of sliding window-based bounding-box detection and subsequent segmentation for  
336 the extraction of high-quality leaf silhouettes from herbarium specimens. Here, the  
337 Faster R-CNN achieved an AP of 25.85. For potential morphometric analyses, we  
338 are not interested in extracting all leaves, but only largely intact ones, even at the  
339 cost of discarding viable instances. Therefore, the relatively low AP is sufficient. The  
340 Mask R-CNN, with an AP of 74.54 before refinement, was very successful at

341 segmenting the leaves inside the bounding boxes. This pipeline already allows to  
342 generate leaf outlines for downstream analyses like Elliptic Fourier Analysis or Leaf  
343 Dissection Index calculation (for an overview of such methods, see McLellan &  
344 Endler, 1998) with little manual effort.

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

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

360

## 361 **Acknowledgements**

362 First and foremost, we would like to thank Christoph Oberprieler (Regensburg), who  
363 acquired the funding, for enabling this work and his comments on the manuscript.

364 We thank Robert Vogt (Berlin) and Sergey Rosbakh (Regensburg) for providing  
365 digital images of *Leucanthemum* specimens and seed mixtures, respectively. We  
366 would also like to thank David Dilcher (Bloomington) for granting us permission to  
367 use the microscopic image shown in Figure 2A. The help of Maximilian Schall and

368 Sebastian Segieth (both Regensburg), who annotated many of the *Leucanthemum*  
369 and seeds images, is much appreciated. We thank Tanja Wenzel for her support in  
370 designing the workflow diagrams. We also thank Agnes Scheunert (Regensburg) for  
371 her comments on the manuscript.

372 This work was supported by a Grant (OB 155/13-1) of the German Research  
373 Foundation (DFG) in the frame of the Priority Programme SPP 1991 “Taxon-omics –  
374 New Approaches for Discovering and Naming Biodiversity” to Christoph Oberprieler).

375

### 376 **Author contributions**

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

381

### 382 **Data availability**

383 GinJinn2’s source code and manual are freely available at GitHub  
384 (<https://github.com/AGOberprieler/GinJinn2>). The annotated Seeds, Yellow-sticky-  
385 traps and *Leucanthemum* datasets are hosted by the German Federation for  
386 Biological Data (GfBio; [Link A](#), [Link B](#), [Link C](#); will be supplied as soon as available).

387 The images used for the Stomata analysis are hosted by the Cuticle Database  
388 (Barclay et al., 2012), a Python script for splitting the images is provided in the  
389 supporting information (S7); the corresponding annotations are hosted by GfBio ([Link](#)  
390 [D](#); will be supplied as soon as available).

391

### 392 **References**

393 Afonso, M., Fonteijn, H., Fiorentin, F. S., Lensink, D., Mooij, M., Faber, N., Polder,  
394 G., & Wehrens, R. (2020). Tomato fruit detection and counting in greenhouses



395 using deep learning. *Frontiers in Plant Science*, 11.  
396 <https://doi.org/10.3389/fpls.2020.571299>

397 Barclay, R. S., Wilf, P., Dilcher, D. L., & McElwain, J. C. (2012). The cuticle database  
398 project. The Earth and Environmental Systems Institute of Pennsylvania State  
399 University [Version 1.1, 10th May 2012]. <http://cuticledb.eesi.psu.edu>

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

404 Carrasco, M., Toledo, P. A., Velázquez, R., & Bruno, O. M. (2020). Automatic  
405 stomatal segmentation based on Delaunay-Rayleigh frequency distance. *Plants*,  
406 9(11). <https://doi.org/10.3390/plants9111613>

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

412 Cheng, H. K., Chung, J., Tai, Y.-W., & Tang, C.-K. (2020). CascadePSP: Toward  
413 class-agnostic and very high-resolution segmentation via global and local  
414 refinement. *ArXiv:2005.02551 [Cs]*. <http://arxiv.org/abs/2005.02551> [preprint]

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

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

- 421 Huang J., Rathod V., Sun C., Zhu M., Korattikara A., Fathi A., Fischer I., Wojna Z.,  
422 Song Y., Guadarrama S., Murphy K. (2017). Speed/accuracy trade-offs for  
423 modern convolutional object detectors. *2017 IEEE Conference on Computer  
424 Vision and Pattern Recognition (CVPR)*. <https://doi.org/10.1109/CVPR.2017.351>
- 425 Jones, H. G. (2020). What plant is that? Tests of automated image recognition apps  
426 for plant identification on plants from the British flora. *AoB PLANTS*, 12(6).  
427 <https://doi.org/10.1093/aobpla/plaa052>
- 428 Li, K., Huang, J., Song, W., Wang, J., Lv, S., & Wang, X. (2019). Automatic  
429 segmentation and measurement methods of living stomata of plants based on the  
430 CV model. *Plant Methods*, 15(1), 67. <https://doi.org/10.1186/s13007-019-0453-5>
- 431 Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., &  
432 Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In D. Fleet, T.  
433 Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp.  
434 740–755). Springer International Publishing. [https://doi.org/10.1007/978-3-319-10602-1\\_48](https://doi.org/10.1007/978-3-319-10602-1_48)
- 435
- 436 McLellan, T., & Endler, J. A. (1998). The relative success of some methods for  
437 measuring and describing the shape of complex objects. *Systematic Biology*,  
438 47(2), 264–281. <https://doi.org/10.1080/106351598260914>
- 439 Meeus, S., Van den Bulcke, J., & Wyffels, F. (2020). From leaf to label: A robust  
440 automated workflow for stomata detection. *Ecology and Evolution*, 10(17), 9178–  
441 9191. <https://doi.org/10.1002/ece3.6571>
- 442 Nieuwenhuizen, A. T., Hemming, J., & Suh, H. K. (2018). Detection and classification  
443 of insects on stick-traps in a tomato crop using Faster R-CNN. *The Netherlands  
444 Conference on Computer Vision*. <https://edepot.wur.nl/463457>
- 445 Norouzzadeh, M. S., Morris, D., Beery, S., Joshi, N., Jovic, N., & Clune, J. (2021). A  
446 deep active learning system for species identification and counting in camera trap

447 images. *Methods in Ecology and Evolution*, 12(1), 150–161.  
448 <https://doi.org/10.1111/2041-210X.13504>

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

452 Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T.,  
453 Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z.,  
454 Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S.  
455 (2019). PyTorch: An imperative style, high-performance deep learning library. In  
456 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett  
457 (Eds.), *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*.  
458 Curran Associates, Inc. [http://papers.neurips.cc/paper/9015-pytorch-an-](http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf)  
459 [imperative-style-high-performance-deep-learning-library.pdf](http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf)

460 Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time  
461 object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee,  
462 M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing*  
463 *Systems 28 (NIPS 2015)*. Curran Associates, Inc.  
464 <https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38>  
465 [046-Paper.pdf](https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38)

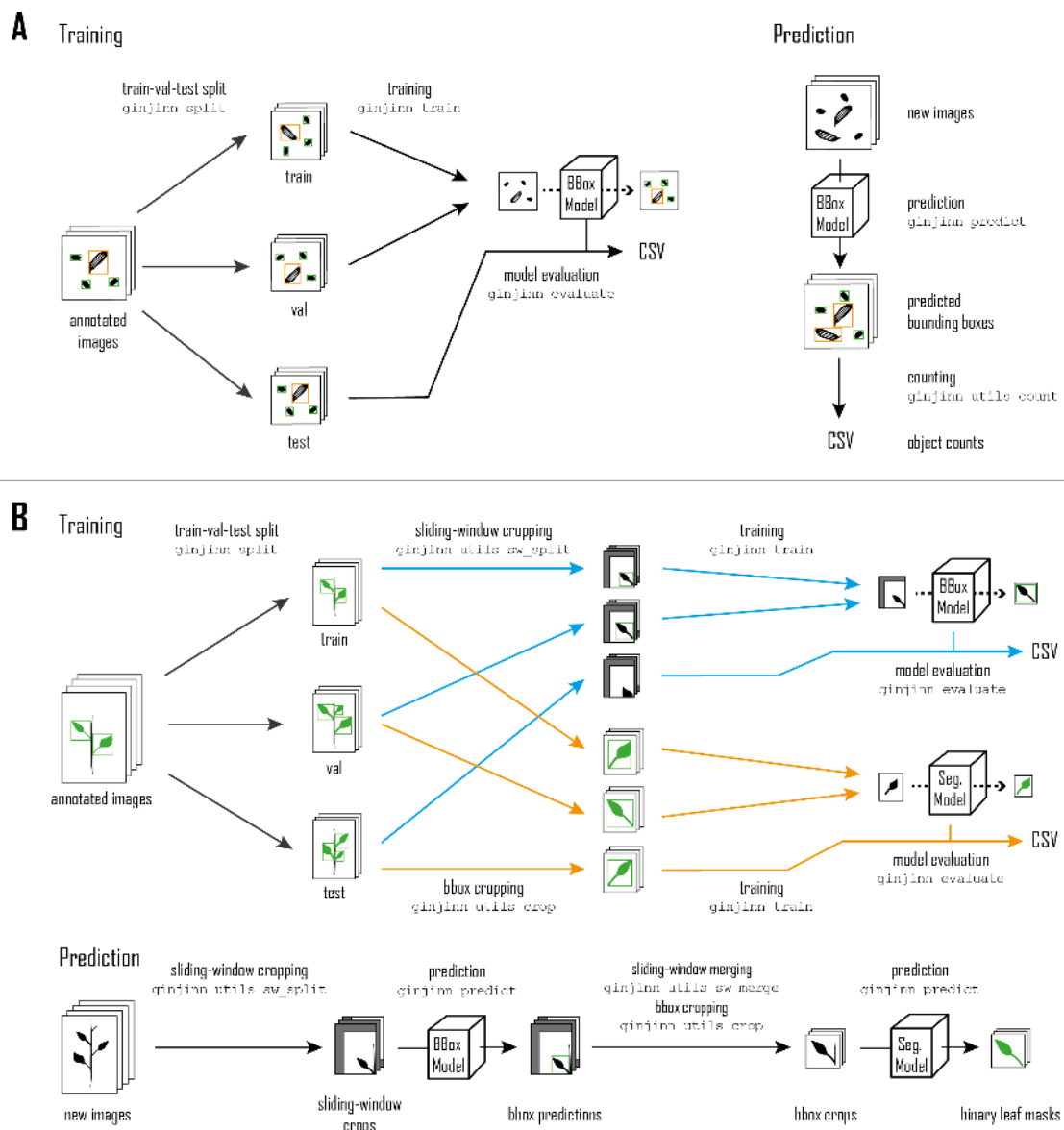
466 Song, W., Li, J., Li, K., Chen, J., & Huang, J. (2020). An automatic method for  
467 stomatal pore detection and measurement in microscope images of plant leaf  
468 based on a convolutional neural network model. *Forests*, 11(9), 954.  
469 <https://doi.org/10.3390/f11090954>

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

- 473 Wäldchen, J., & Mäder, P. (2018). Machine learning for image based species  
474 identification. *Methods in Ecology and Evolution*, 9(11), 2216–2225.  
475 <https://doi.org/10.1111/2041-210X.13075>
- 476 Weaver, W. N., Ng, J., & Laport, R. G. (2020). LeafMachine: Using machine learning  
477 to automate leaf trait extraction from digitized herbarium specimens. *Applications*  
478 *in Plant Sciences*, 8(6), e11367. <https://doi.org/10.1002/aps3.11367>
- 479 Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2.  
480 <https://github.com/facebookresearch/detectron2>
- 481 Younis, S., Schmidt, M., Weiland, C., Dressler, S., Seeger, B., & Hickler, T. (2020).  
482 Detection and annotation of plant organs from digitised herbarium scans using deep  
483 learning. *Biodiversity Data Journal*, 8, e57090. <https://doi.org/10.3897/BDJ.8.e57090>

484

485 **Figures**

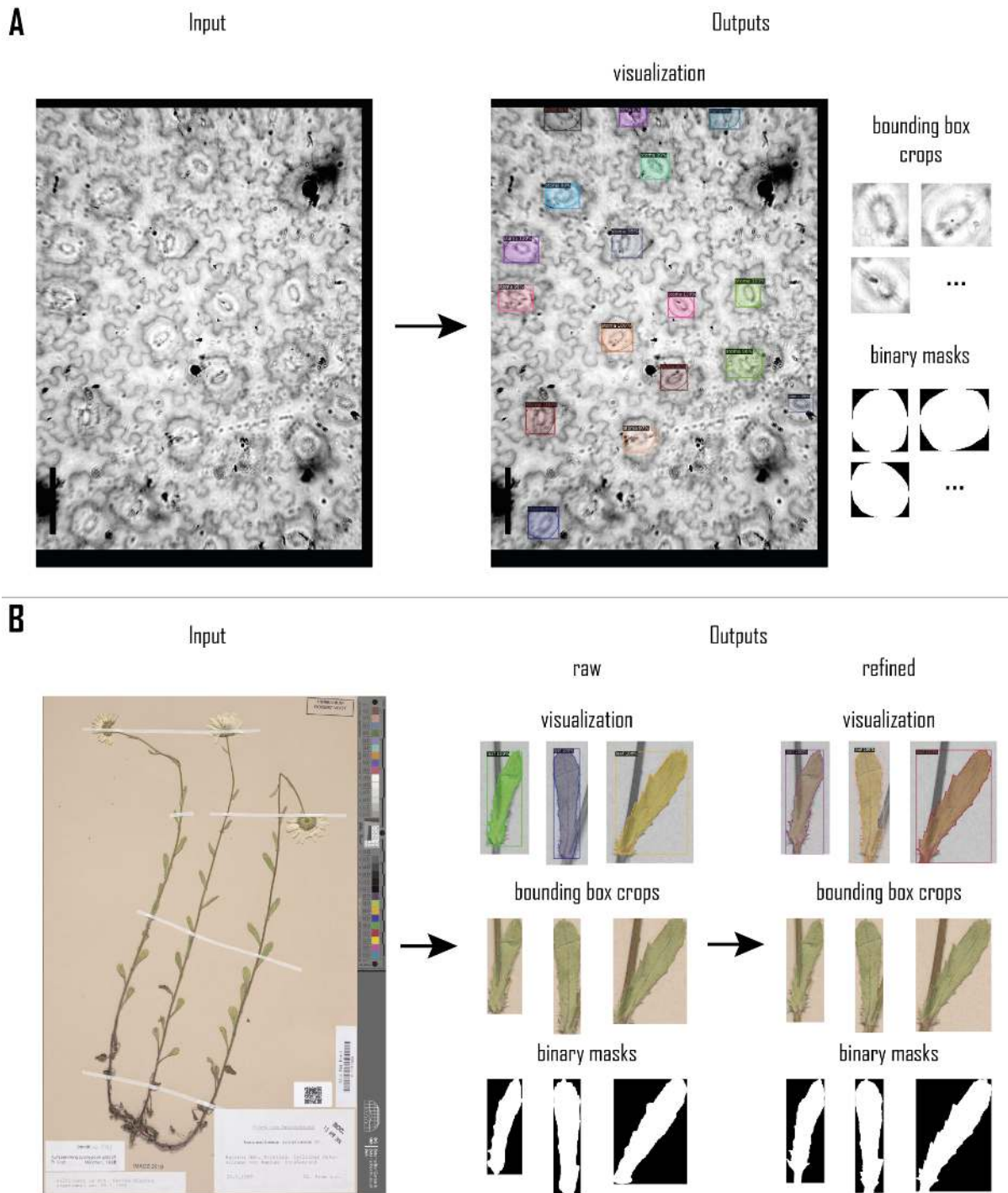


486

487 Figure 1. Seeds (A) and *Leucanthemum* (B) analysis workflows. The Seeds dataset  
 488 is split into training, validation, and test datasets, which are used to train and  
 489 evaluate a bounding-box model (A, Training). The trained model is applied to new  
 490 data for seed counting (A, Prediction). The *Leucanthemum* dataset is also split into  
 491 training, validation, and test datasets, but the workflow comprises training and  
 492 evaluation of two separate models (B, Training). The blue branch refers to a  
 493 bounding-box model for the detection of leaves on sliding-window crops of the split  
 494 dataset. The orange branch depicts the training and evaluation of an instance  
 495 segmentation model on padded bounding boxes cropped from the split datasets.

496 Leaf segmentations for new data are predicted by combining both models (**B**,  
497 Prediction).

498



499

500 Figure 2. Exemplary outputs from the Stomata (**A**) and *Leucanthemum* (**B**) analyses.

501 **A** depicts a single input image along with corresponding predictions by the stomata  
502 model, showing different output formats. Similarly, **B** shows an input image and  
503 corresponding predictions for the *Leucanthemum* pipeline, before and after  
504 segmentation refinement.

505

## 506 **Supporting information**

507 commands.pdf:

508 Appendix S1. GinJinn2 commands of exemplary analyses.

509 seeds.yaml:

510 Appendix S2. GinJinn2 configuration file for the seeds analysis.

511 stickytraps.yaml:

512 Appendix S3. GinJinn2 configuration file for the yellow-sticky-traps analysis.

513 stomata.yaml:

514 Appendix S4. GinJinn2 configuration file for the stomata analysis.

515 leucanthemum\_bbox.yaml:

516 Appendix S5. GinJinn2 configuration file for the *Leucanthmum* analysis (bounding-  
517 box detection).

518 leucanthemum\_segmentation.yaml:

519 Appendix S6. GinJinn2 configuration file for the *Leucanthmum* analysis (instance  
520 segmentation).

521 split\_image.py:

522 Appendix S7. Image splitting script.