Google Colaboratory for Quantifying Stomata in Images

Ángela Casado-García[1](✉), Jónathan Heras[1], and Alvaro Sanz-Sáez[2]

[1] Department of Mathematics and Computer Science, University of La Rioja, Logroño, Spain
{angela.casado,jonathan.heras}@unirioja.es

[2] Department of Crop, Soil, and Environmental Sciences, Auburn University, Auburn, USA
sanz@auburn.edu

Abstract. Stomata are pores in the epidermal tissue of plants formed by specialized cells called occlusive cells or guard cells. Analyzing the number and behavior of stomata is a task carried out by studying microscopic images, and that can serve, among other things, to better manage crops in agriculture. However, quantifying the number of stomata in an image is an expensive process since a stomata image might contain dozens of stomata. Therefore, it is interesting to automate such a detection process. This problem can be framed in the context of object detection, a task widely studied in computer vision. Currently, the best approaches to tackle object detection problems are based on deep learning techniques. Although these techniques are very successful, they might be difficult to use. In this work, we face this problem, specifically for the detection of stomata, by building a Jupyter notebook in Google Colaboratory that allows biologists to automatically detect stomata in their images.

Keywords: Object detection · YOLO · Stomata images · Google Colaboratory

1 Introduction

Stomata (singular "stoma") are pores on a plant leaf, see Fig. 1, that allow the exchange of gases, mainly CO_2 and water vapor, between the atmosphere and the plant. Stomata respond to changes in the environment and regulate the photosynthesis of plants, and thus their productivity. Due to these facts, scientists have studied the number, density, size, and behavior of stomata to model CO_2 dynamics in the atmosphere and predict future carbon and water cycles [10]; and, also, in crop breeding and crop management programs [3,11].

Partially supported by Ministerio de Industria, Economía y Competitividad, project MTM2017-88804-P; and Agencia de Desarrollo Económico de La Rioja, project 2017-I-IDD-00018. We also acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

R. Moreno-Díaz et al. (Eds.): EUROCAST 2019, LNCS 12014, pp. 231–238, 2020.
https://doi.org/10.1007/978-3-030-45096-0_29

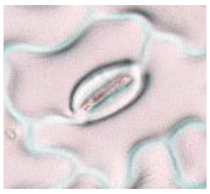


Fig. 1. Example of a stoma

In order to analyze stomata of plant leaves, plant biologists take microscopic images of leaves, and manually measure the stomata density, individual stomata opening, and morphological traits like the size and shape of the stomata guard cells (a pair of cells that surround each stoma). Those measurements are repeated over hundreds of images from different plant varieties. This is a tedious, error-prone, time-consuming and subjective task due to the large number of stomata in each image, see Fig. 2; but, it can be automatized by means of detection and imaging techniques. In this work, we present a first step towards such an automatization. Namely, using deep learning techniques, we have created a tool that automatically detects stomata in an image. The process to develop that kind of tool can be applied to other problems, and consists of two steps: training a detection model (see Sect. 2), and creating a graphical interface, in our case a Google colaboratory notebook, see Sect. 3.

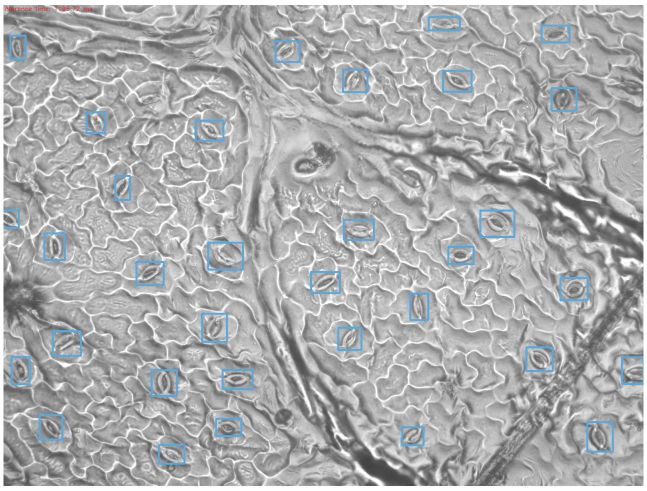


Fig. 2. Example of stomata detection

2 Training a Stomata Detection Model

In this section, we present the common workflow to create a deep learning-based object detector (see Fig. 3), the challenges that are faced on each stage of the workflow, the alternatives that can be employed to face those problems, and the concrete solutions given for the stomata case.

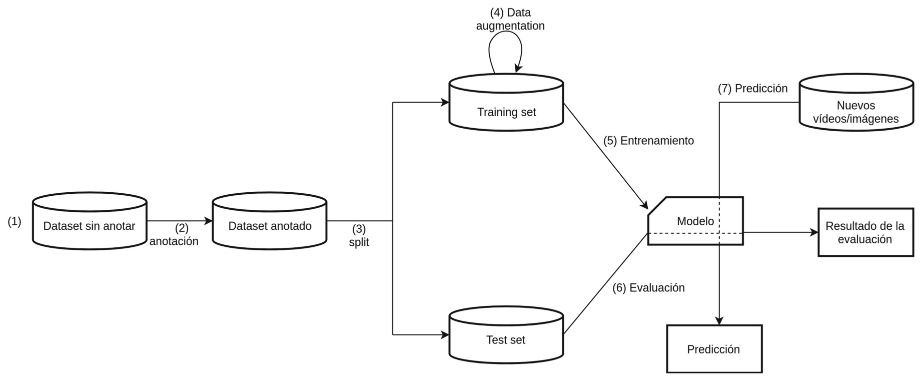


Fig. 3. Workflow of object detection projects

Nowadays, there are several algorithms and libraries that allow the creation of object detection models. Although each of them is different, the process to build a model is independent and always consists of the same 6 steps: dataset acquisition, dataset annotation, dataset split, dataset augmentation, model training and model evaluation.

Dataset Acquisition. The starting point of any image detection project is the acquisition of a dataset of images. This is an important step since the dataset must be representative of what the model will later find when deployed in the real world. Different alternatives can be used to obtain a dataset of images. For example, using open datasets, web scrapping images with the appropriate license, using images provided by a client, or using special purpose devices to capture the images. In our case, stomata images are acquired with a microscope, and this kind of image in not usually freely available. Therefore, we created our own dataset of images. Namely, our dataset consists of 500 stomata images of soy plants captured with a Leica microscope.

Dataset Annotation. The second step of the process consists in annotating the dataset; that is, indicating the position of the objects in the images. This is a time-consuming stage that might require the intervention of expert users. To simplify this task, there are graphical interactive tools, like LabelImg [22] or

YOLO mark [1], that help in the generation of the annotation files. Unfortunately, there is not a standard annotation format and, in fact, the format varies among object detection frameworks and annotation tools. In our case, the images were annotated using LabelImg by expert biologists.

Dataset Split. After annotating the images, and, As in any other machine learning project, it is instrumental to split the dataset obtained in the previous steps into two independent sets: a training set—that will be employed to train the object detector—and a test set—that will be employed to evaluate the model. Common split sizes for training and testing set include 66.6%/33.3%, 75%/25%, and 90%/10%. In our case, the dataset was split into a training set containing the 75% of the images (500 images), and a testing set containing the other 25% of the images (100 images).

Dataset Augmentation. Deep learning methods are data demanding and require a considerable amount of images. In the case of small datasets (for example, in the case of biomedical images that are difficult to obtain), it is posible to apply data-augmentation [20], a technique that consists in generating new training samples from the original dataset by applying image transformations (for instance, applying flips, rotations, filters or adding noise). This approach has been applied in image classification problems [20,21], and there are several libraries implementing this method (for instance, Augmentor [2] or Imgaug [12]). However, those libraries cannot be applied in the context of object detection since they do not alter the annotation (note that if we flip an image, the position, and hence the annotation, of the objects change). In our case, we employed CLODSA [9] a library that not only generates new images by applying data-augmentation but also creates the corresponding annotations. By employing this library, the training dataset of 500 images was augmented using flips, rotations, filters or adding noise, obtaining a final dataset of 4500 images.

Model Training. Currently there are two types of deep learning algorithms used to build object detection models: two-phase and one-phase algorithms. The two-phase detection algorithms have two components: the first is responsible for proposing regions of interest that are used by the second component to decide the position of the objects. These algorithms are usually accurate but slow at runtime, so they can not be used in real-time applications. Examples of algorithms of this type are R-CNN [8], Fast R-CNN [7] and Faster R-CNN [17]. The second type of detection algorithms are the one-phase algorithms, that are trained from start to end to make the detections. These algorithms are fast but not so precise as two-phase algorithms, although most of these algorithms are introducing improvements that increase their accuracy without affecting their response times, see Table 1. Among these algorithms we can find SSD [15], RetinaNet [14] or YOLO [16]. Recently the YOLO algorithm, a one-phase algorithm, has reached a precision comparable to that of two-phase algorithms, but keeping the processing in real time. This is why we have chosen this algorithm for our work. The YOLO algorithm was trained during 200.000 iterations using the default settings of this algorithm.

Table 1. Precision and detection time of algorithms in the Pascal Voc dataset [6]

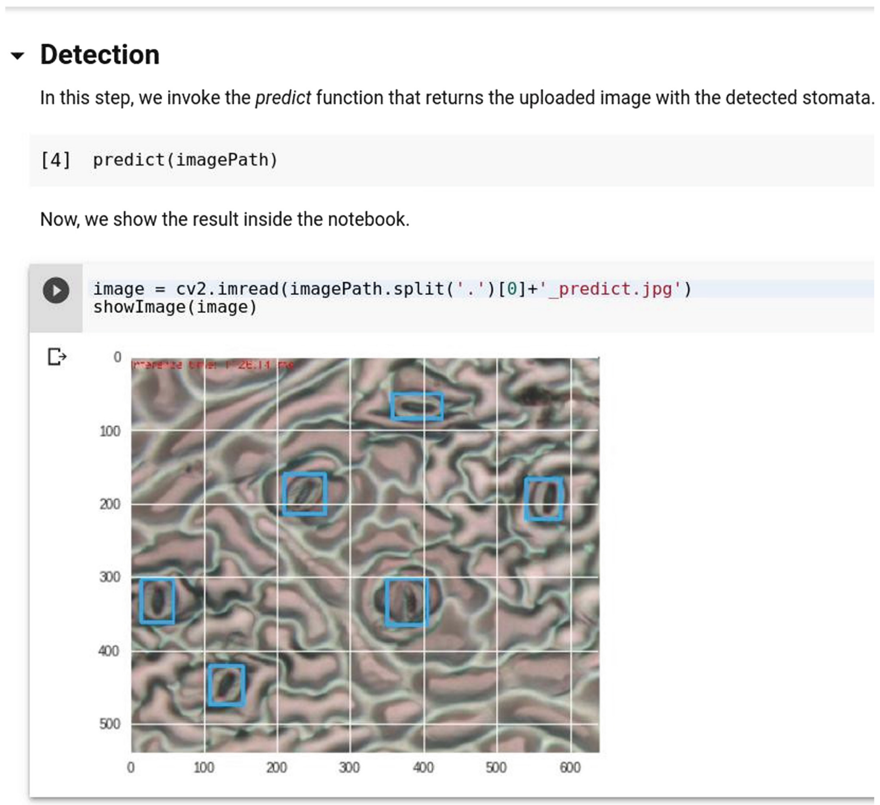
Detection framework	mAP	FPS
Faster RCNN - VGG16	73.2	7
Faster RCNN - ResNet	76.4	5
YOLO v1	63.4	45
SSD 500	76.8	19
YOLO (416×416 image size)	76.8	67
YOLO (480×480 image size)	77.8	59

Model Evaluation. Finally, after training a model, we need to evaluate it to assess its performance. Namely, the images in the testing set, are presented to the model and ask it to detect the objects in those images. Those detections are compared to the ground-truth provided by the annotations of the testing set, and the result of the comparison is evaluated using metrics such as the IoU [18], the mAP [18], the precision, the recall or the F1-score [6]. A problem that might arise during the evaluation is the overfitting of the object detection model; that is, the model can detect objects on images from the training dataset, but it cannot detect objects on any other image. In order to avoid this problem, early stopping [19] can be applied by comparing the results obtained by the models after different numbers of iterations. Using this technique, in our stomata detection model, we built a model with a mAP of 90.91% and a precision of 98% in the testing set.

The complete workflow to train the stomata detection model is available at github https://github.com/ancasag/YOLONotebooks; and such a workflow can be generalized to other problems as explained in [4]. Finally, the model is ready to be employed in images that neither belong to the training set nor to the testing set.

3 A Google Colaboratory Notebook

Using a detection model with new images is usually as simple as invoking a command with the path of the image (and, probably, some additional parameters). However, this requires the installation of several libraries and the usage of a command line interface; and, this might be challenging for non-expert users like biologists. Therefore, it is important to create simple and intuitive interfaces that might be employed by different kinds of users; otherwise, they will not be able to take advantage of the object detection model.



Detection

In this step, we invoke the *predict* function that returns the uploaded image with the detected stomata.

```
[4]  predict(imagePath)
```

Now, we show the result inside the notebook.

```
image = cv2.imread(imagePath.split('.')[0]+'_predict.jpg')
showImage(image)
```

Fig. 4. Notebook to automatically detect stomata in images

To disseminate our detection model, we have created a Jupyter notebook, see Fig. 4, that allows users to detect stomata in their images. Jupyter notebooks [13] are documents for publishing code, results and explanations in a form that is both readable and executable; and, they have been widely adopted across multiple disciplines, both for its usefulness in keeping a record of data analyses, and also for allowing reproducibility. The drawback of Jupyter notebooks is that they require the installation of several libraries. Such a problem has been overcome in our case by providing our notebook in Google Colaboratory [5], a free Jupyter notebook environment that requires no setup and runs entirely in the cloud avoiding the installation of libraries in the local computer. The notebook is available at https://github.com/ancasag/Stomata.

4 Conclusions and Further Work

In this work, we have developed a tool that allows biologists to automatically detect stomata in images. To build this tool, it has been necessary to create a object detection model using the YOLO algorithm. In addition, since using models directly in the framework that provides YOLO models might be difficult for biologist we have developed a simple-to-use interface by means of a Jupyter notebook available in Google Colaboratory.

Several tasks remain as further work. First of all, we are planning to train a model with stomata images from several varieties of plants since the current model was built only soy plants. Moreover, we will develop a tool that allows users to interact with the result produced by the detection model; and not only obtain stomata density, but also other useful information.

References

1. Alexey, A.B.: YOLO mark (2018). https://github.com/AlexeyAB/Yolomark
2. Bloice, M.D., Stocker, C., Holzinger, A.: Augmentor: an image augmentation library for machine learning. J. Open Source Softw. **2**, 432 (2017)
3. Buttery, B.R., Tan, C.S., Buzzell, R.I., Gaynor, J.D., MacTavish, D.C.: Stomatal numbers of soybean and response to water stress. Plant Soil **149**(2), 283–288 (1993). https://doi.org/10.1007/BF00016619
4. Casado-García, A., Heras, J.: Guiding the creation of deep learning-based object detectorss. In: Proceedings of the XVIII Conferencia de la Asociacion Española para la Inteligencia Artificial (CAEPIA 2018), session DEEPL 2018 (2018)
5. Colaboratory Team: Google Colaboratory (2017). https://colab.research.google.com
6. Everingham, M., et al.: The pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. **111**(1), 98–136 (2015)
7. Girshick, R.: Fast R-CNN. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015)
8. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
9. Heras, J., et al.: CLoDSA: an open-source image augmentation library for object classification, localization, detection and semantic segmentation (2018). https://github.com/joheras/CLoDSA
10. Hetherington, A.M., Woodward, F.I.: The role of stomata in sensing and driving environmental change. Nature **424**(6951), 901–908 (2003)
11. Hughes, J., et al.: Reducing stomatal density in barley improves drought tolerance without impacting on yield. Plant Physiol. **174**(2), 776–787 (2017)
12. Jung, A.: Imgaug: a library for image augmentation in machine learning experiments (2017). https://github.com/aleju/imgaug
13. Kluyver, T., et al.: Jupyter notebooks – a publishing format for reproducible computational workflows. In: Proceedings of the 20th International Conference on Electronic Publishing, pp. 87–90. IOS Press (2016)
14. Lin, T., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. IEEE Trans. Pattern Anal. Mach. Intell. **39**, 2980–2988 (2017). abs/1708.02002

15. Liu, W., et al.: SSD: single shot MultiBox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
16. Redmon, J., Farhadi, A.: YOLOv3: An Incremental Improvement. CoRR abs/1804.02767 (2018). http://arxiv.org/abs/1804.02767
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, vol. 28, pp. 91–99 (2015)
18. Rosebrock, A.: Deep Learning for Computer Vision with Python. PyImageSearch (2018). https://www.pyimagesearch.com/
19. Sarle, W.S.: Stopped training and other remedies for overfitting. In: Proceedings of the 27th Symposium on the Interface of Computing Science and Statistics, pp. 352–360 (1995)
20. Simard, P., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR 2003), vol. 2, pp. 958–964 (2003)
21. Simard, P., Victorri, B., LeCun, Y., Denker, J.S.: Tangent prop - a formalism for specifying selected invariances in an adaptive network. In: Proceedings of the 4th International Conference on Neural Information Processing Systems (NIPS 1991). Advances in Neural Information Processing Systems, vol. 4, pp. 895–903 (1992)
22. Tzutalin, D.: LabelImg (2015). https://github.com/tzutalin/labelImg