The Journal of Open Source Software

Ipyannotator: the infinitely hackable annotation framework

Ítalo Epifânio[1], **Oleksandr Pysarenko**[1], **and Immanuel Bayer**[1]

1 Palaimon GmbH

DOI: 10.21105/joss.04480

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @csadorf
- @matthewfeickert

Submitted: 13 April 2022
Published: 24 August 2022

Summary

Annotation is a task that associates semantic tags to a digital representation (image, video, text, and others). This process is important for supervised machine learning (ML) approaches, since the model learns, and successively improves, from data examples in form of input-output pairs.

The variety of digital representations makes it difficult to develop a tool that is flexible and meets all requirements for machine learning applications in different projects. Ipyannotator is an open-source tool that allows manual annotation in multiple data formats, enabling researchers/users to explore, create, and improve their datasets without leaving their own development environment, and empowering them to extend and customize the annotation process.

Statement of need

Many breakthroughs in machine learning (ML) applications such as image classification, text understanding, and recommender systems belong to the class of supervised machine learning. These ML methods often require an extensive basis of annotated data from which the model learns. The amount and quality of the annotated data is essential to generate a model that yields accurate prediction (Wong et al., 2015).

The variety of annotation taks, data formats, and the respective visualization of data is enormous. When dealing with multiple domains of supervised ML, the large variety of applications is a challenging task, especially considering that the tooling is potentially not flexible enough, which imposes limitations to the user.

Ipyannotator is a library developed to provide a solution to this problem. Ipyannotator can be used to visualize, create, and improve datasets, providing a flexible solution that can be extended and customized by the user within the Jupyter development environment.

Jupyter is one of the most popular tools for data science (Wang et al., 2019) and is currently in used in more than 7850000 public repositories on GitHub (Parente, 2022). Ipyannotator is a tool developed to be used with Jupyter Notebooks, allowing researchers and developers to integrate the library into their ML projects quickly and easily. The solution, however, is not limited to research and developement teams. Since the Ipyannotator runs also on a web server, it can by used for annotation purposes by any user.

Lahtinen et al. (2021) developed an annotator as a browser plugin, Dutta & Zisserman (2019) built an annotator as a program that runs on the browser, and Bernal et al. (2019) developed a desktop application for domain-specific medical image annotation. These previous annotation tools are restricted to fixed data types and a specific type of annotation. Ipyannotator was designed to be executed in Jupyter notebooks and allows users to change its iterations on runtime, allowing extensions and customization, creating an "infinitely hackable annotation framework" [1].

Infinitely hackable annotation framework

Ipyannotator uses Python libraries, such as Ipywidgets (Widgets, 2022), Ipycanvas (Renou, 2022), and Ipyevents (Craig, 2022), that abstract the HTML and Javascript interactions, allowing developers to design UI interactions and elements using Python. Python's dynamic nature allows users to modify classes or modules at runtime, and due to the libraries mentioned, users can change the default behavior of Ipyannotator's UI, hacking the library. Browser interaction such as mouse moving and HTML elements such as dropdowns are some of the examples that can be changed at runtime when using Ipyannotator.

Being integrated with Jupyter notebooks makes Ipyannotator usage easy to modify at different abstraction levels. The data science team can change the library's behavior while writing their own scripts on a platform and programming language that they already know.

In addition to the ability to change Ipyannotator's browser interaction using Python, the library also provides a flexible API that enables adding custom annotators. With a custom pair of input and output classes, the user can create and register a new annotator while reusing the library resources.

A simple but flexible API to define annotation tasks

Ipyannotator provides a simple API (application programming interface), which is based on three steps that describe general tasks in the data annotation process. These are denoted as the explore, create, and improve phases.

These three steps in conjuction with domain-specific annotation types define the inputs and outputs of the annotation process, providing a very flexible and extendable API to set up annotation tasks.

The following code examples illustrate the main actions around which the Ipyannotator API is built. Please keep in mind that Ipyannotator aims to be flexible enought that these generic aspects can be extented to much complexer and domain-specific tasks and interfaces. An exemplary application, a standard image classification task, is used in the following.

```
from pathlib import Path
from ipyannotator.base import Settings
from ipyannotator.annotator import Annotator
from ipyannotator.mltypes import InputImage, OutputImageLabel

input = InputImage(image_dir='images', image_width=200, image_height=200)
output = OutputImageLabel(label_dir='labels', label_width=30, label_height=30)
settings = Settings(project_path=Path('data'))

annotator = Annotator(input, output, settings)
```

Explore

The explore step provides the ability to visualize and navigate through images and datasets. Figure 1 displays an example of an image classification task using one of Ipyannotator's built-in artificial demonstration datasets. The dataset consists of images showing simple shapes in

[1] "infinitely hackable" is one of the key design principles used. Even through using very thin Python wrappers, e.g., Renou (2022) for the JavaScript Web Canvas API, ipyannotator still has to obey the finite limitations of the Python language.

different colors. The annotation task is to assign to each image the color that is closest to the provided labels.

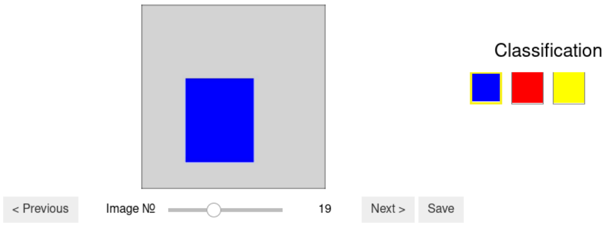


Figure 1: Explore step for image classification task

Create

In the create step new annotation datasets can be created by the user. Figure 2 presents an example of an image classfication task, allowing the user to manually select multiple options and save the results in the dataset.

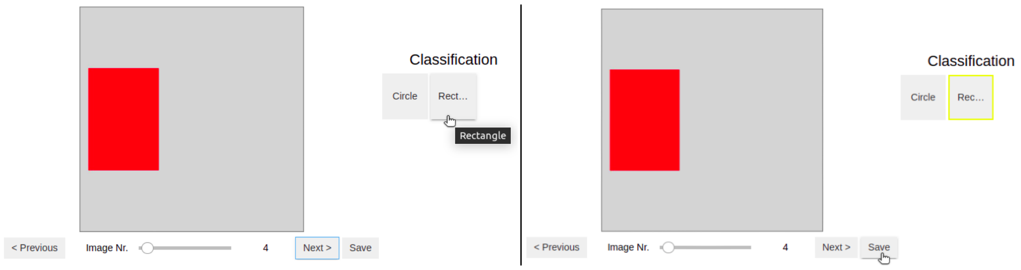
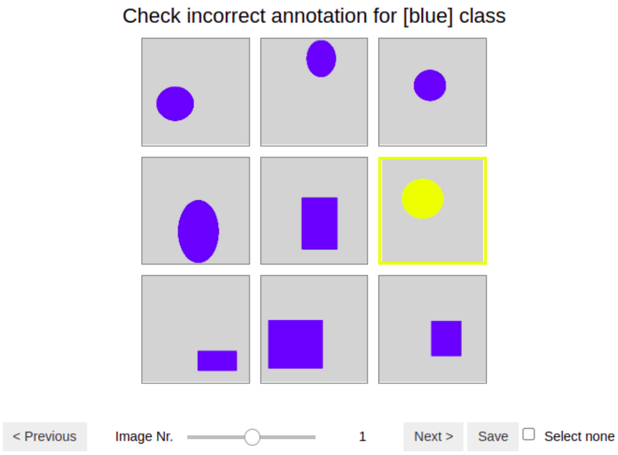


Figure 2: Create step for image classification task

Improve

The improve step enables the user to refine datasets and thereby improve the annotation quality. Figure 3 displays the usage of improve, allowing users to iterate over every class and identify incorrect results.

Inspecting large batches of pre-annotated data allows a user to very quickly improve the quality of datasets that were generated by an insufficient ML model. It also allows a domain expert to verify and improve the annotation work of less specialized annotators.



Check incorrect annotation for [blue] class

Figure 3: Improve step for image classification task

Key Design Decisions

Jupyter Notebook all the way down

Jupyter Notebooks are used by many researcher relaying on open source software to create and document their work. Ipyannotator not only runs directly in Jupyter Notebooks but is also developed as a collection of notebooks. This collection constitues a library, user documentation, and executable tutorials. This workflow is enabled by the innovative fastai library (Fastai, 2022) that turns Jupyter Notebook into a literate programming environment.

For the development of the user interface (UI), the Ipywidget library (Widgets, 2022) was used to build a graphical user interface (GUI) in Jupyter Notebook. Furthermore, the voila library (Dashboards, 2022), which uses Jupyter Notebook as a web-app, was also incorporated in the Ipyannotator project to create the GUI for an easy to access web application.

Architecture

Ipyannotator's architecture consists of three main systems components that comprise the user interface (UI), the server, and the data storage. These components are targeted at two different user types. A non-code architecture, which is schematically illustrated in Figure 4, is included for non-technical annotators. The setup for a wide range of technically experienced annotators, schematically shown in Figure 5, targets users typically involved in research projects, e.g., data scientists, domain experts, and software developers.

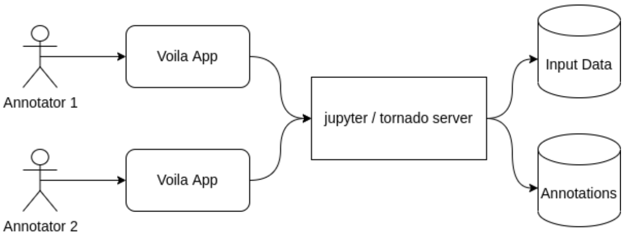


Figure 4: Non-technical user architecture

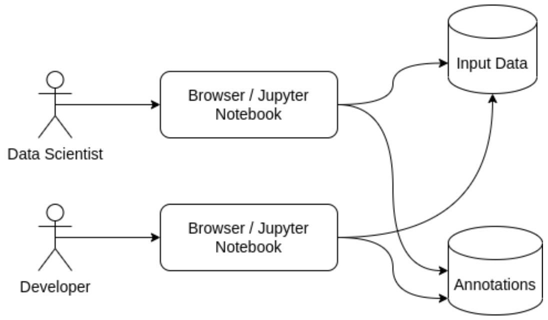


Figure 5: Technical user architecture

For the technical user, multiple tutorials are provided, demonstrating Ipyannotator's utilization. The tutorials make it easier for new users get started and adapt the notebooks to their tasks. They also demonstrate the annotation workflow and different features.

Usage

Currently, Ipyannotator is used for supervised ML projects by the devloper Palaimon GmbH. Multiple tutorials and use cases have been tested and published to improve and validate the usage of Ipyannotator on other real world projects.

Acknowledgements

The authors acknowledge the financial support by the Federal Ministry for Digital and Transport of Germany under the program mFUND for the project OS-VAT (project number 19F2160A).

References

Bernal, J., Histace, A., Masana, M., Angermann, Q., Sánchez-Montes, C., Rodríguez de Miguel, C., Hammami, M., García-Rodríguez, A., Córdova, H., Romain, O., & others. (2019). GTCreator: A flexible annotation tool for image-based datasets. *International Journal of Computer Assisted Radiology and Surgery*, *14*(2), 191–201. https://doi.org/10.1007/s11548-018-1864-x

Craig, M. (2022). *Ipycanvas* (Version 0.12.0) [Computer software]. https://github.com/mwcraig/ipyevents

Dashboards, V. (2022). *Voila* (Version 0.3.4) [Computer software]. https://github.com/voila-dashboards/voila

Dutta, A., & Zisserman, A. (2019). The VIA annotation software for images, audio and video. *Proceedings of the 27th ACM International Conference on Multimedia*, 2276–2279. https://doi.org/10.1145/3343031.3350535

Fastai. (2022). *Nbdev* (Version 1.2.4) [Computer software]. https://github.com/fastai/nbdev

Lahtinen, T., Turtiainen, H., & Costin, A. (2021). BRIMA: Low-overhead BRowser-only IMage annotation tool (preprint). *2021 IEEE International Conference on Image Processing (ICIP)*, 2633–2637. https://doi.org/10.1109/icip42928.2021.9506683

Parente, P. (2022). Nbestimate. In *GitHub repository*. https://github.com/parente/nbestimate; GitHub.

Renou, M. (2022). *Ipycanvas* (Version 0.12.0) [Computer software]. https://github.com/martinRenou/ipycanvas

Wang, A. Y., Mittal, A., Brooks, C., & Oney, S. (2019). How data scientists use computational notebooks for real-time collaboration. *Proc. ACM Hum.-Comput. Interact.*, *3*(CSCW). https://doi.org/10.1145/3359141

Widgets, J. (2022). *Ipywidgets* (Version 7.6.3) [Computer software]. https://github.com/jupyter-widgets/ipywidgets

Wong, Y.-S., Chu, H.-K., & Mitra, N. J. (2015). SmartAnnotator an interactive tool for annotating indoor RGBD images. *Computer Graphics Forum*, *34*, 447–457. https://doi.org/10.1111/cgf.12574