**Project Report**

**YOLO8 ANNOTATION TOOL**

**Master’s in computer science**

<https://github.com/RajeshRamadas/Yolo8-Annotation-Tool.git>

Student: Rajesh Kumar Ramadas

rajesh-kumar.ramadas@iu-study.org

Matriculation: 92125100

Course name                      Project: Computer Science Project

(DLMCSPCSP01)

University: International University of Applied Sciences

Juri-Gagarin-Ring 152 · D-99084 Erfurt

University Supervisor: Dr. Oezdemir Cetin

oezdemir.cetin@iu.org

Submission Date:  December 2024

Contents

[1. PROJECT OVERVIEW 3](#_Toc184369902)

[2. PROJECT GOALS 4](#_Toc184369903)

[3. FEATURES 4](#_Toc184369904)

[3.1. Image Loading and Navigation 4](#_Toc184369905)

[3.2. Bounding Box Drawing 4](#_Toc184369906)

[3.3. Annotation Management 4](#_Toc184369907)

[3.4. Image Settings 5](#_Toc184369908)

[3.5. Annotation Conversion 5](#_Toc184369909)

[3.6. Dataset Splitting 5](#_Toc184369910)

[3.7. PNG Conversion 6](#_Toc184369911)

[3.8. Logging 6](#_Toc184369912)

[4. TECHNOLOGICAL STACK 6](#_Toc184369913)

[5.1. YOLO Format 7](#_Toc184369914)

[Fig 5.1. Annotation data 7](#_Toc184369915)

[5.2. Pascal VOC Format 8](#_Toc184369916)

[5.2.1. Structure of Pascal VOC Format 8](#_Toc184369917)

[5.2.2. Example of Pascal VOC Format 9](#_Toc184369918)

[Fig 5.2. Pascal VOC format(sample) 9](#_Toc184369919)

[5.2.3. Advantages of Pascal VOC Format 9](#_Toc184369920)

[5.3. Key Differences Between YOLO and Pascal VOC Formats 9](#_Toc184369921)

[5.4. COCO Json Format 10](#_Toc184369922)

[Structure of a COCO JSON file 10](#_Toc184369923)

[Fig 5.2. COCO Json format(sample) 11](#_Toc184369924)

[6. TARGET USERS 11](#_Toc184369925)

[7. LIMITATIONS 12](#_Toc184369926)

[8. REPOSITORY & PREREQUISITES 13](#_Toc184369927)

[8.1. https://github.com/RajeshRamadas/Yolo8-Annotation-Tool.git 13](#_Toc184369928)

[8.2. Prerequisites 13](#_Toc184369929)

[8.3. Installation Steps & Execution 13](#_Toc184369930)

[8.3.1 Clone the Repository 13](#_Toc184369931)

[8.3.2. Install the Required Dependencies: 13](#_Toc184369932)

[8.3.4 Run the Application: 13](#_Toc184369934)

[8.4. Application 14](#_Toc184369936)

[Fig 8.4. Application snapshot 14](#_Toc184369938)

[9. STEPS TO USE APPLICATION. 14](#_Toc184369939)

[9.1. PNG Converter: 14](#_Toc184369940)

[9.2. Log Window: 15](#_Toc184369947)[\_Toc184369948](#_Toc184369948)

[9.3. Load Images: 15](#_Toc184369950)[\_Toc184369954](#_Toc184369954)

[9.4. Bounding box: 16](#_Toc184369956)

[9.5. Image Setting (Image Augmentation): 19](#_Toc184369975)

[9.6. Annotation setting: 20](#_Toc184369985)

[9.7. Annotation validation: 20](#_Toc184369991)

[9.8. Validation Overlapping Annotation: 21](#_Toc184369999)

[9.9. Training Dataset: 22](#_Toc184370005)[\_Toc184370007](#_Toc184370007)

[10. ANNOTATION VALIDATION WITH YOLO8 MODEL 26](#_Toc184370021)

[10.1. Training Yolo8 Model: 26](#_Toc184370022)

[11. CONCLUSION 29](#_Toc184370042)

[11.1. Key Strengths 29](#_Toc184370043)

[11.2. Future Directions 30](#_Toc184370044)

[12. BIBLIOGRAPHY 30](#_Toc184370045)

## 1. PROJECT OVERVIEW

The **Yolo8 Annotation Tool** is an intuitive Python-based application created to streamline the process of building annotated datasets for object detection models, specifically those that utilize the YOLOv8 framework. In the realm of object detection, having accurately labeled training data is paramount, yet achieving this can often be a daunting task filled with challenges such as time constraints and the potential for human error. This is where our tool steps in, offering a user-friendly solution that simplifies image annotation.

Built with **PyQt6**, the tool features a graphical interface designed for users of all backgrounds—from those just starting their journey in machine learning to seasoned data scientists. It allows users to easily annotate images by drawing bounding boxes around objects. This capability is crucial for defining regions of interest (ROI) in object detection workflows, as these bounding boxes are integral to training machine learning models to recognize and localize various objects in images.

To enhance the annotation process, the Yolo8 Annotation Tool includes a variety of features aimed at improving efficiency. Users can take advantage of batch processing, undo/redo options, and automatic dataset splitting, all of which significantly reduce the time and effort required for dataset preparation. Moreover, the tool incorporates data augmentation techniques, such as resizing and adjusting brightness/contrast, to increase the diversity of the training dataset—an essential factor for improving model performance.

Additionally, the tool provides built-in options for converting image formats, ensuring consistency throughout the annotation process. While the current version supports only single-category annotations and is specifically optimized for YOLOv8, its solid foundation allows for potential future enhancements, which could expand its capabilities to meet more complex annotation needs and accommodate additional machine learning frameworks.

## 2. PROJECT GOALS

The primary goals of the Yolo8 Annotation Tool are:

* **User-Friendly Annotation:** To provide an easy-to-use platform that enhances productivity while minimizing errors during the dataset preparation process.
* **High-Quality Data Production:** To ensure the generation of high-quality annotated datasets that can significantly improve the performance of machine learning models, particularly for object detection tasks.
* **Simplified Dataset Management:** To offer features that streamline the image annotation process and facilitate efficient management of datasets.

## 3. FEATURES

### 3.1. Image Loading and Navigation

The Yolo8 Annotation Tool allows users to easily manage and navigate their image datasets:

* **Load Images:** Users can select a directory containing their image files, allowing the tool to load these images for annotation. This feature enables quick access to large datasets without the need to open each image manually.
* **Navigate Images:** Navigation controls like "Previous" and "Next" buttons help users move seamlessly between loaded images, making it easy to browse the dataset and concentrate on specific images without interruptions.

### 3.2. Bounding Box Drawing

Creating bounding boxes around objects is a core functionality of the tool:

* **Drawing:** Users can annotate objects in images through a simple click-and-drag action. This intuitive functionality makes it accessible for both beginners and experienced annotators.
* **Bounding Box Details:** After drawing a bounding box, users receive detailed information about its dimensions and coordinates, which is crucial for maintaining accuracy in the annotation process.

### 3.3. Annotation Management

Managing annotations is straightforward with the Yolo8 Annotation Tool:

* **Save Annotations:** Users can save their drawn bounding boxes in a YOLO-compatible format, ensuring the dataset is ready for training without requiring additional formatting steps.
* **Load Annotations:** Previously saved annotations can be loaded, which will automatically display on the corresponding image, making it easy to review or modify existing work.
* **Delete Annotations:** Users have the option to delete an image’s annotations if necessary, offering flexibility in dataset management.
* **Validate Annotations:** The tool can check if each image has a corresponding annotation file, alerting users to any missing annotations, thus ensuring data integrity.

### 3.4. Image Settings

The tool provides several image manipulation options to optimize images for annotation:

* **Adjust Dimensions:** Users can modify image width and height, allowing for customization according to annotation requirements.
* **Rotate Image:** The rotation slider enables users to rotate images at any desired angle, facilitating annotation from different perspectives.
* **Adjust Brightness and Contrast:** Sliders for brightness and contrast adjustments help ensure that objects are clearly visible, regardless of lighting conditions.

### 3.5. Annotation Conversion

The tool includes a feature for converting annotation formats:

* **Convert to VOC XML:** Users can convert YOLO format annotations to PASCAL VOC XML format. This flexibility is valuable for users transitioning between different machine learning models**.**
* **Convert to COCO JSON:** Users can convert YOLO format annotations to COCO format.

### 3.6. Dataset Splitting

Splitting datasets becomes a breeze with this tool:

* **Training Dataset: Users can divide their datasets into training, validation, and test sets based on specified ratios. This automated process ensures proper organization for model training.**

### 3.7. PNG Conversion

Consistency in image format is crucial, and the tool offers conversion functionality:

* **Convert to PNG:** Users can convert images from various formats into PNG format, simplifying preprocessing steps for machine learning projects.

### 3.8. Logging

To enhance user experience, the tool includes a logging system:

* **Log Messages:** A dedicated log window provides real-time feedback, keeping users informed of actions taken and any issues that may arise during the annotation process.

## 4. TECHNOLOGICAL STACK

The Yolo8 Annotation Tool leverages a robust technological stack:

* 1. **Programming Language: Python**  
     Chosen for its simplicity and strong compatibility with machine learning libraries, Python facilitates ease of development and integration with YOLOv8 models. (python , n.d.)
  2. **GUI Framework: PyQt6**  
     PyQt6 allows for the creation of a cross-platform, user-friendly interface. Its extensive set of widgets enables smooth navigation and customization. (ultralytics , n.d.)
  3. **Image Processing Libraries: OpenCV or PIL**
* **OpenCV:** Offers fast image manipulation functions ideal for large datasets. (opencv, n.d.)
* **PIL:** Provides a simpler alternative for basic image processing and format conversion tasks. (python-pillow, n.d.)

1. **ANNOTATION FORMAT: YOLOV8-COMPATIBLE**

Annotations are stored in a format optimized for object detection tasks, ensuring smooth integration with YOLOv8 models for training.

YOLO (You Only Look Once), Pascal VOC (Visual Object Classes) and COCO Json are widely used annotation formats in the field of computer vision, specifically for object detection tasks. Each format has its unique structure and use cases. Below is a detailed explanation of both formats.

### 5.1. YOLO Format

The YOLO format is used primarily for training models in the YOLO object detection framework. This format is characterized by its simplicity and efficiency in representing bounding boxes for detected objects.

#### **5.1.1. Structure of YOLO Format**

1. **Annotation File**: Each image has a corresponding text file with the same name as the image but with a .txt extension. For example, for an image named image1.jpg, the annotation file would be image1.txt.
2. **Line Format**: Each line in the text file represents one annotated object and contains the following information:

* **class\_id**: An integer that denotes the category of the object. The class IDs start from 0.
* **x\_center**: The X coordinate of the center of the bounding box, normalized by the width of the image (a value between 0 and 1).
* **y\_center**: The Y coordinate of the center of the bounding box, normalized by the height of the image (a value between 0 and 1).
* **width**: The width of the bounding box, normalized by the width of the image (a value between 0 and 1).
* **height**: The height of the bounding box, normalized by the height of the image (a value between 0 and 1).

#### **5.1.2. Example of YOLO Format**

For an image image1.jpg, the corresponding annotation file image1.txt may look like this:



## Fig 5.1. **Annotation data**

In this example:

* The first line indicates an object of class ID 0 (the first category) centered at (0.5, 0.5) with a width of 0.2 and a height of 0.3 of the image dimensions.
* The second line indicates an object of class ID 1, centered at (0.75, 0.25) with a width of 0.15 and a height of 0.2.

#### **5.1.2. Advantages of YOLO Format**

* **Efficiency**: The format is compact and easy to parse, making it efficient for model training.
* **Real-Time Processing**: YOLO is designed for real-time object detection, and the format aligns well with this goal by providing quick access to bounding box information.
* **Normalization**: Using normalized coordinates makes the annotations independent of image size, simplifying the training process with images of varying dimensions.

### 5.2. Pascal VOC Format

The Pascal VOC format is an older but still widely used annotation format for object detection tasks. It was originally developed for the Pascal Visual Object Classes Challenge, which aimed to evaluate and benchmark the performance of object detection algorithms.

### Structure of Pascal VOC Format

1. **Annotation Files**: Each image has a corresponding XML file that contains detailed information about the objects present in the image. The naming convention is similar to YOLO, with the XML file sharing the same name as the image but with an .xml extension (e.g., image1.xml for image1.jpg).
2. **XML Structure**: The XML file contains several key elements:

* **filename**: The name of the image file.
* **size**: Dimensions of the image, including width, height, and depth (number of channels).
* **object**: Each object is represented within its own <object> tag, which contains:
  + - **name**: The class label of the object.
    - **bndbox**: The bounding box coordinates, including:
      * **xmin**: The x-coordinate of the top-left corner of the bounding box.
      * **ymin**: The y-coordinate of the top-left corner of the bounding box.
      * **xmax**: The x-coordinate of the bottom-right corner of the bounding box.
      * **ymax**: The y-coordinate of the bottom-right corner of the bounding box.

### Example of Pascal VOC Format

Here’s an example of how an XML file (image1.xml) may look:



## Fig 5.2. **Pascal VOC format(sample)**

In this example:

* The image is named image1.jpg and has a size of 800x600.
* There are two objects: a "car" and a "person," each with specified bounding boxes defined by their corners.

### Advantages of Pascal VOC Format

* **Rich Metadata**: The XML structure allows for additional metadata to be included, such as image dimensions and depth.
* **Multi-Class Support**: Each XML file can contain multiple objects of different classes, making it suitable for complex datasets.
* **Standardized**: Being a widely used format, it is supported by many libraries and frameworks, making it easier to integrate into different workflows.

### 5.3. Key Differences Between YOLO and Pascal VOC Formats

| **Feature** | **YOLO Format** | **Pascal VOC Format** |
| --- | --- | --- |
| **File Type** | .txt files for each image | .xml files for each image |
| **Bounding Box Representation** | Normalized coordinates (center x, center y, width, height) | Absolute pixel coordinates (xmin, ymin, xmax, ymax) |
| **Class ID** | Integer values starting from 0 | Class names (strings) |
| **Multi-Class Support** | Each image file can contain multiple annotations | Supports multiple objects per image in one file |
| **Efficiency** | More compact and faster to process | More verbose due to XML structure |
| **Image Size Independence** | Yes, annotations are normalized | No, annotations are absolute coordinates |

### COCO Json Format

The COCO JSON format is a widely used annotation format for object detection, segmentation, and key point detection tasks, originally developed for the [COCO (Common Objects in Context)](https://cocodataset.org/) dataset. It organizes annotations into a JSON file containing various fields for images, annotations, categories, and other metadata.

### Structure of a COCO JSON file

A COCO JSON file contains the following top-level keys:

1. info: General information about the dataset.
2. licenses: Information about the dataset's licensing.
3. images: Metadata about the images in the dataset.
4. annotations: The actual annotations (bounding boxes, segmentation masks, or key points).
5. categories: Class label definitions for the dataset.

Here’s an example of how a coco Json file (image1.xml) may look:



### Fig 5.2. **COCO Json format(sample)**

## 6. TARGET USERS

The Yolo8 Annotation Tool is tailored for three main user groups:

* 1. **Data Scientists:**  
     Data scientists require high-quality datasets for model training. The tool provides:
  + **Streamlined Annotation:** An intuitive GUI reduces manual effort.
  + **Data Augmentation:** Options for resizing and brightness adjustments enhance dataset variability.
  + **YOLOv8 Compatibility:** Generates datasets ready for model training.
  1. **Researchers in Computer Vision:**  
     Researchers need reliable tools for creating datasets. The tool offers:
  + **Customizable Annotations: Precise bounding box placement for high-quality data.**
  + **Ease of Use: A user-friendly interface accommodates varying technical backgrounds.**
  + **Experimental Flexibility: Supports dataset splitting and format conversion.**
  1. **Machine Learning Practitioners:**  
     Practitioners focused on object detection benefit from:
  + **YOLOv8 Integration: Direct labeling for training YOLO models.**
  + **High-Quality Data: Ensures precise annotations with validation features.**
  + **Dataset Management: Automates the splitting of datasets into training, validation,** and test sets.

## 7. LIMITATIONS

While the Yolo8 Annotation Tool offers a variety of powerful features, it also has limitations:

**7.1. Single Category Annotation:**  
The tool currently supports only one object category per project. This can restrict flexibility for users dealing with datasets containing multiple object classes, resulting in:

* + **Workflow Constraints:** Users must manage separate projects for different categories, leading to inefficiencies.
  + **Reduced Data Richness:** This limitation may prevent users from generating diverse datasets essential for robust model training.
  1. **Best suited for wide display screen:**

**Annotation tool works best on wider screen than a laptop screen.**

* 1. **Application support for windows machine only.**

**exe file only supports windows machine.**

* 1. **Single image format support**:

Tool currently supports only one images format(.png) (consider using PNG CONVERTER)

### 8. REPOSITORY & PREREQUISITES

### 8.1. <https://github.com/RajeshRamadas/Yolo8-Annotation-Tool.git>

### 8.2. Prerequisites

**Before installing the Yolo8 Annotation Tool, make sure you have the following installed on your system:**

* + **Python 3.9+**: **Ensure Python is installed. You can download it from** [python.org](https://www.python.org/downloads/).
  + **pip**: **Python package manager, usually installed with Python**.

### 8.3. Installation Steps & Execution

### 8.3.1 ****Clone the Repository****

### ****8.3.2. Install the Required Dependencies:****

### 

### ****8.3.4 Run the Application:****

### 

### ****8.4. Application****

## 

## Fig 8.4. **Application snapshot**

## 9. STEPS TO USE APPLICATION.

## 9.1. PNG Converter:

## 

## Fig 9.1.1 **Use the option to convert images from other formats to <.png> format.**

## 

## Fig 9.1.2. **A folder named <converted\_PNG> has been created, and the .png images have been placed inside it.**

## 

## Fig 9.1.3. **PNG generated files as snapshot.**

## 9.2. Log Window:

## 

## Fig 9.2.1. **Log for the PNG converter is available as snapshot.**

## 9.3. Load Images:

## **Select a folder that contains only <.png> format images.**

## **Once the images are loaded, the IMAGES FILES section will display a list of the loaded images.**

## **The display area will show the image selected from the IMAGES FILES list.**

## 

## Fig 9.3.1. **Highlighted section is displaying area and Image files.**

## 9.4. Bounding box:

## **Select the top-left corner and drag downwards to create a bounding box.**

## **Enter the annotation ID, then click the "Save Annotation" button.**

## **Once the "Save Annotation" button is clicked, a resave pop-up window will appear.**

## **After the image is saved, a .txt file will be generated in the same folder as the image.**

## 

## Fig 9.4.1. **Bounding box snapshot**

## 

## Fig 9.4.2. **Update the annotation ID or class.**

## 

## Fig 9.4.3. **Save annotation triggers option to resave image.**

## 

## Fig 9.4.4. **Pop-up window to resave image.**

## 

## Fig 9.4.5. **log window displays a record of the saved images.**

## 

## Fig 9.4.6. **Annotation file generated with .txt extension.**

## 

## Fig 9.4.7. **yolo8 format annotation**

## 9.5. Image Setting (Image Augmentation):

## **Set the image size by adjusting the height and width.**

## **Rotate the image with a 360-degree rotation option.**

## **Adjust the brightness and contrast as needed.**

## **Random, Flip Image and Color Jitter**

## **To save the changes, click "Save Image" (please note that auto-save is not supported).**

## 

## Fig 9.5.1. **Image setting with rotation and save image snapshot.**

## 

## Fig 9.5.2. **Saved image.**

## 9.6. Annotation setting:

## ****Load Annotation**: After an annotation is created, the bounding box can be displayed by selecting this option, provided the annotation file exists.**

## ****Delete Annotation:** The annotation file can be deleted using this option, but only if the annotation file is present.**

## ****Redo and Undo:** The annotation bounding box can be redone or undone as needed.**

## 

## 

## Fig 10.6.1. **Annotation setting as highlighted.**

## 9.7. Annotation validation:

## ****Validate Annotation:** Once all bounding boxes are updated, this option allows you to verify whether all annotation files have been generated. When the button is clicked, a pop-up window will display a list of files that are missing annotations. If all annotations are generated, no files will be listed.**

## 

## Fig 9.7.1. **pop-up window shows list of files annotation not done yet.**

## 

## Fig 9.7.2. **Annotation files for respective images**

## 

## Fig 9.7.3. **Pop-up window for validate annotation.**

## 9.8. Validation Overlapping Annotation:

## **“Validate overlap annotation.” Provides option to verify overlapping bounding box.**

## **Intersection Over Union (IoU) method is used to validate overlapping bounding box.**

## 

## 

## Fig 9.8.1. **Overlapping annotation validation**

## 9.9. Training Dataset:

## **Option offers a feature to easily split the dataset into training, validation, and testing sets, facilitating the process of preparing the dataset for deep learning models.**

## **Enter the training, validation, and testing split ratio.**

## 

## Fig 9.9.1. **Pop-up window for splitting dataset in training, validation, and testing.**

## 

## Fig 9.9.2. **Pop-up window for splitting dataset is successful.**

## 

## Fig 9.9.3. **Splitting dataset into folders (training, validation, and testing)**

## 

## Fig 9.9.4. **Splitting dataset into folders for test images and labels**

## 

## Fig 9.9.5. VOC XML format converter from yolo8 annotation

## 

## Fig 9.9.5. COCO JSON format converter from yolo8 annotation

## 

## Fig 9.9.6. COCO JSON format and VOC XML format generated.

## 10. ANNOTATION VALIDATION WITH YOLO8 MODEL

## 10.1. Training Yolo8 Model:

## 10.1.1. Dataset: YOLO models expect data in the COCO or YOLO format. A typical YOLO directory structure for custom datasets is:

## 

## Fig 10.1.1.directory structure for custom datasets.

## 10.1.2. Data Configuration: Create a data.yaml file specifying paths and class names.

## 

## Fig 10.1.2. Data Configuration with data.yaml file.

## 

## 10.1.3. Training YOLOv8 Model: Python code for training a YOLOv8 model with custom data.

## 

## Fig 10.1.3. YOLOv8 train model.



**Fig 10.1.4.** YOLOv8 model training image



## Fig 10.1.5. YOLOv8 model validation image

## 10.1.4. Evaluate the Model: After training, you can evaluate the model’s performance using.

## 

## Fig 10.1.6. YOLOv8 model validation code

## 

## Fig 10.1.7. YOLOv8 model validation with limited dataset and CPU.

**Precision (B):** 0.0047

* Indicates a very low precision, meaning there are many false positives relative to true positives.

**Recall (B):** 0.7778

* This shows that the model is correctly identifying about 78% of actual alpacas, which is a significant improvement.

**mAP@0.5 (B):** 0.3075

* This indicates a reasonably good performance in terms of mean Average Precision at an IoU threshold of 0.5, suggesting that the model is better at correctly localizing detections.

**mAP@0.5:0.95 (B):** 0.1226

* This value indicates that there is still room for improvement in the model's performance across a range of IoU thresholds, especially for higher precision.

**Fitness Score:** 0.1411

* The fitness score combines various metrics, and improvements in precision and mAP will enhance this score.

## 10.1.5. Prediction: After training and evaluation, model need to be tested with sample image.

## 

## Fig 10.1.5. YOLOv8 prediction

## ****Note:**** Training a custom dataset from scratch requires substantial GPU resources and a large dataset. Due to resource constraints (CPU-only environment and limited dataset), we conducted training to verify that the YOLOv8 model (YOLOv8.pt) is capable of detecting and validating the annotations.

## 10.1.6. GitHub link sample training using annotation tool.

## [Yolo8-Annotation-Tool/02-Implementation/demo/train\_yolo8 at main · RajeshRamadas/Yolo8-Annotation-Tool](https://github.com/RajeshRamadas/Yolo8-Annotation-Tool/tree/main/02-Implementation/demo/train_yolo8)

## 11. CONCLUSION

The Yolo8 Annotation Tool stands out as a powerful and user-friendly application designed to streamline the process of creating annotated datasets for object detection tasks. With its intuitive graphical user interface (GUI), the tool caters to a diverse range of users, including data scientists, computer vision researchers, and machine learning practitioners. Its key features, such as batch processing, bounding box annotation, data augmentation options, and dataset management capabilities, significantly enhance productivity and efficiency in preparing datasets for training machine learning models.

### 11.1. Key Strengths

The strengths of the Yolo8 Annotation Tool lie in its focus on usability, accuracy, and efficiency. The tool’s ability to simplify the annotation process, along with its features for managing datasets and converting annotations, makes it a valuable resource for anyone involved in developing object detection models.

### 11.2. Future Directions

While the current version serves its purpose effectively, there is room for improvement. Future updates could expand the tool’s capabilities by supporting multi-category annotations and incorporating additional formats for annotations. By continuously enhancing the tool, we can better meet the evolving needs of users and keep pace with advancements in machine learning and computer vision.

Overall, the Yolo8 Annotation Tool is a significant step forward in facilitating the creation of high-quality datasets, helping users produce better-performing models and pushing the boundaries of what is possible in the field of object detection.

## 12. BIBLIOGRAPHY

*opencv*. (n.d.). Retrieved from https://opencv.org/: https://opencv.org/

*python* . (n.d.). Retrieved from https://www.python.org/: https://www.python.org/

*python-pillow*. (n.d.). Retrieved from https://python-pillow.org/: https://python-pillow.org/

*ultralytics* . (n.d.). Retrieved from https://pypi.org/project/: https://pypi.org/project/ultralytics/