Understanding Pascal VOC and COCO Annotations for Object Detection

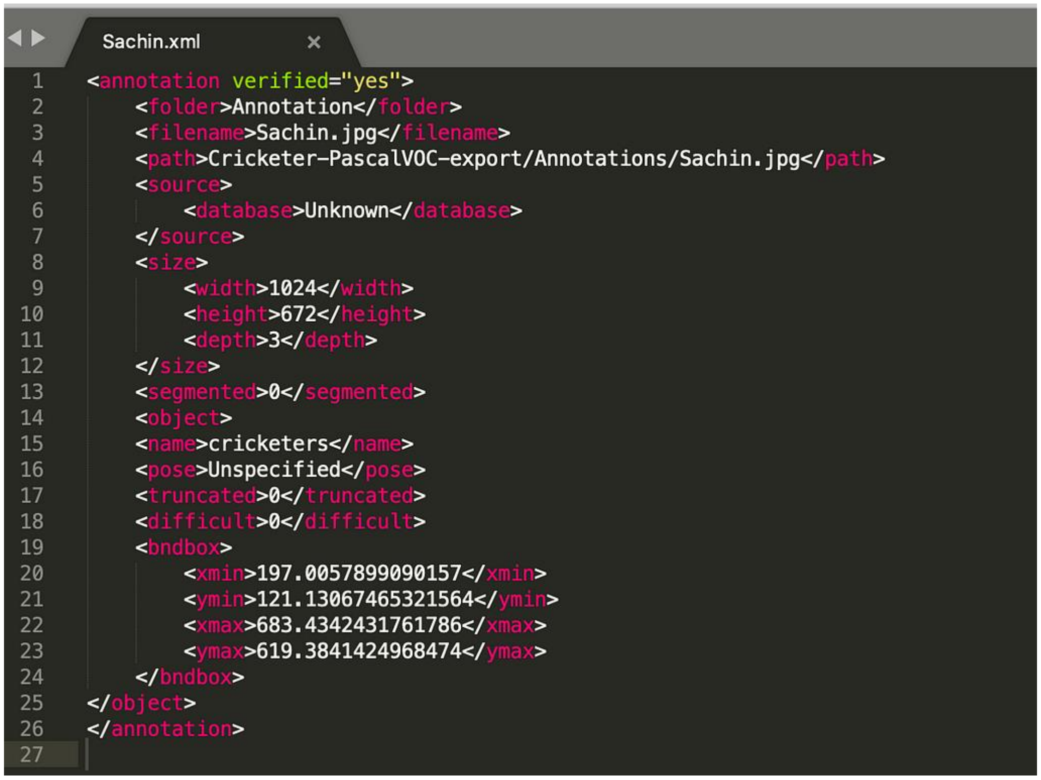
Pascal VOC (Visual Object Classes) is a format to store annotations for localizer or Object Detection datasets and is used by different annotation editors and tools to annotate, modify and train Machine Learning models. In PASCAL VOC format, for each image there is a xml annotation file containing image details, bounding box details, classes, rotation and other data.

Pascal VOC Annotations Structure:

Pascal VOC annotations are typically organized in the following structure:

Image Files: The dataset includes a collection of images, each associated with a unique filename.

Annotations: For each image, there is a corresponding annotation file in XML format. These XML files contain detailed information about the objects present in the image.



```xml
<annotation verified="yes">
    <folder>Annotation</folder>
    <filename>Sachin.jpg</filename>
    <path>Cricketer-PascalVOC-export/Annotations/Sachin.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>1024</width>
        <height>672</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
    <name>cricketers</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
        <xmin>197.0057899090157</xmin>
        <ymin>121.13067465321564</ymin>
        <xmax>683.4342431761786</xmax>
        <ymax>619.3841424968474</ymax>
    </bndbox>
</object>
</annotation>
```

Fig example of a Pascal VOC XML annotation for an image with one object

Let's break down the key components of a Pascal VOC annotation:

1. **Annotation XML File:**

Each annotation XML file corresponds to a specific image and contains the following elements:

<**annotation**>: The root element that encapsulates the entire annotation.

<**filename**>: Specifies the name of the image file.

<**size**>: Contains information about the image's size.

<**width**>: The width of the image in pixels.

<**height**>: The height of the image in pixels.

<**depth**>: The number of color channels (e.g., 3 for RGB).

2. Object Information:

Within each annotation XML, there is a section for each annotated object in the image. This section includes the following elements:

<object>: Represents an individual object in the image.

<name>: Specifies the object's class/category label (e.g., "car," "dog").

<pose>: Describes the pose or orientation of the object (e.g., "Frontal").

<truncated>: Indicates whether the object is partially visible or truncated (0 for non-truncated, 1 for truncated).

<difficult>: Flags objects that are difficult to detect (0 for not difficult, 1 for difficult).

<bndbox>: Defines the bounding box that surrounds the object.

<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.

COCO Annotations for Object Detection

COCO (Common Objects in Context) Annotations for Object Detection are a structured and widely used format for annotating and labeling objects within images for the purpose of training and evaluating object detection models in computer vision. COCO is a benchmark dataset and annotation format that has become a standard in the field. Here's a more detailed explanation

- **Bounding Boxes**: The primary annotation in COCO for object detection is the bounding box. A bounding box is a rectangular region drawn around an object within an image. It is defined by four coordinates: (x, y) for the top-left corner and (width, height) for the dimensions of the box. Each annotated object is associated with one or more bounding boxes.
- **Object Categories**: COCO annotations also include the labeling of object categories or classes. Each annotated object is assigned to a specific category or class, such as "car," "dog," "chair," etc. COCO provides a predefined set of 80 common object categories.
- **Annotation Format**: COCO annotations are typically stored in JSON (JavaScript Object Notation) format. JSON is a lightweight data interchange format that allows for easy organization and sharing of annotation data. This format includes information about bounding box coordinates, object categories, image metadata, and more.
- **Image Metadata**: Each image in the COCO dataset is associated with metadata, including an image ID, file name, image height, and image width. This metadata helps link annotations to the corresponding images.

Convert bounding boxes from Coco to Pascal VOC to Yolo and back

Converting bounding boxes from COCO (Common Objects in Context) format to Pascal VOC (Visual Object Classes) format and then to YOLO (You Only Look Once) format involves reshaping the data into different structures commonly used in object detection tasks.

What are bounding boxes?

Bounding boxes are rectangles used to surround objects in images. They are often used in object detection. There are multiple formats of bounding boxes (Coco, Yolo, Pascal). Each format uses its specific representation of bounding boxes coordinates. The following picture illustrates a bounding box.

Coco: Format [x_min, y_min, width, height]

The coordinates (x_min, y_min) are the top-left corner along with the width and height of the bounding box.

Pascal VOC: Format: [x_min, y_min, x_max, y_max]

x_min and y_min are coordinates of the top-left corner and x_max and y_max are coordinates of bottom-right corner of the bounding box.

Yolo: Format: [x_center, y_center, width, height]

x_center and y_center are the normalized coordinates of the center of the bounding box. The width and height are the normalized length. To convert YOLO in Coco or Pascal or vice versa it is important to have the size of the image to calculate the normalization.

COCO to Pascal VOC:

- COCO Format: COCO format includes information about each object's class label, bounding box coordinates (x, y, width, height), and other attributes in a JSON file.

- Pascal VOC Format: Pascal VOC format stores annotations in XML files. Each XML file corresponds to an image and contains information about object classes, bounding box coordinates, and image size.

Conversion Steps:

- Parse the COCO JSON file to extract object annotations.
- Create a corresponding XML file for each image in Pascal VOC format.
- Populate the XML file with object class labels, bounding box coordinates (in xmin, ymin, xmax, ymax format), and other information.
- Repeat this process for all images in the dataset.

Pascal VOC to YOLO:

- Pascal VOC Format: In Pascal VOC format, bounding box coordinates are specified as (xmin, ymin, xmax, ymax), and object class labels are in text format. This format is not directly compatible with YOLO.

- YOLO Format: YOLO format uses a text file for each image, containing object class IDs and normalized bounding box coordinates (center_x, center_y, width, height), all relative to the image dimensions.

Conversion Steps:

- Parse the Pascal VOC XML files to extract object annotations, including class labels and bounding box coordinates in (xmin, ymin, xmax, ymax) format.
- Normalize the bounding box coordinates by dividing them by the image width and height, resulting in (center_x, center_y, width, height) format.
- Map Pascal VOC class labels to unique class IDs (starting from 0).
- Create a YOLO-formatted text file for each image, including class IDs and normalized bounding box coordinates.

YOLO to Pascal VOC (Optional, for Backward Conversion):

- YOLO Format: YOLO format stores annotations in text files with class IDs and normalized bounding box coordinates (center_x, center_y, width, height).
- Pascal VOC Format: Pascal VOC format uses XML files with object class labels and bounding box coordinates in (xmin, ymin, xmax, ymax) format.

Conversion Steps (Optional, for Backward Conversion):

- Parse the YOLO text files to extract class IDs and normalized bounding box coordinates.
- Denormalize the bounding box coordinates by multiplying them by the image width and height, converting them back to (xmin, ymin, xmax, ymax) format.
- Map YOLO class IDs back to class labels.
- Create Pascal VOC XML files with class labels and bounding box coordinates.

What is CVAT?

CVAT stands for "Computer Vision Annotation Tool." It is an open-source software application designed to assist with the annotation and labeling of images and videos for computer vision and machine learning tasks. CVAT provides a user-friendly interface for annotators to draw bounding boxes, polygons, key points, and other annotations on images and video frames. It is a valuable tool for creating labeled datasets that can be used to train and evaluate computer vision algorithms, including object detection, image segmentation, and facial recognition models.

CVAT offers features for collaborative annotation, making it possible for multiple annotators to work on the same dataset simultaneously. It also supports various annotation formats, making it compatible with different machine learning frameworks and libraries. CVAT has gained popularity in the computer vision community as a powerful and versatile annotation tool for research and development projects.

CVAT is used for labeling data for solving computer vision tasks such as:

- Image Classification
- Object Detection
- Object Tracking
- Image Segmentation
- Pose Estimation
-

Image classification

Image classification in CVAT involves using the CVAT tool to label or categorize images into different classes or groups. This process helps organize and prepare data for training machine learning models to automatically recognize and classify objects or patterns within images. Users upload images, define categories, assign labels to images, review and correct annotations, and then export the labeled data for training and evaluation of image classification algorithms.

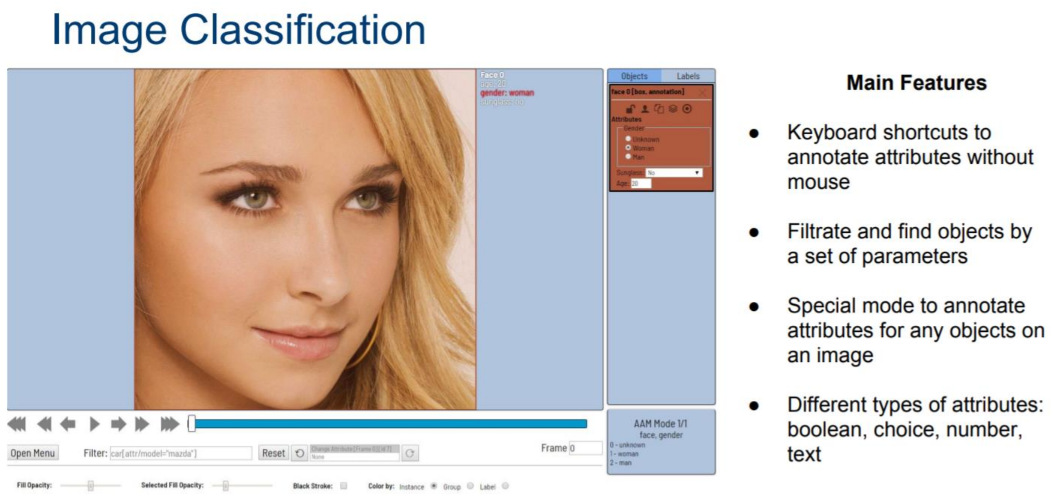


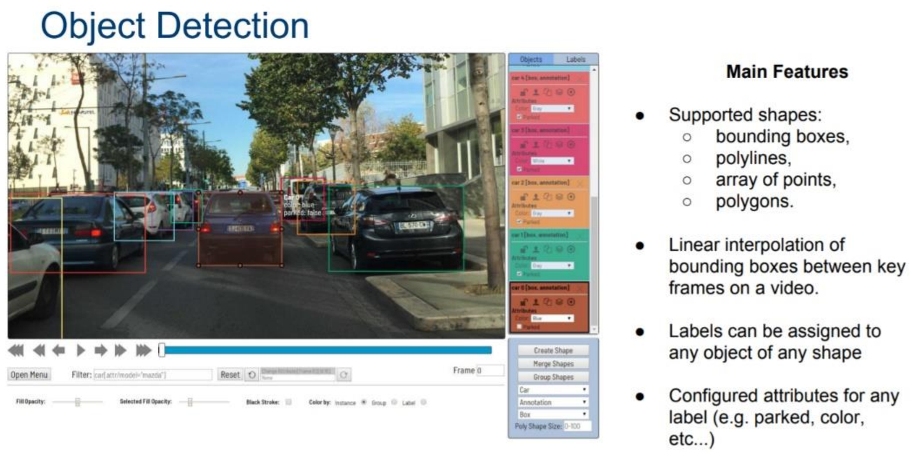
Image Classification

Main Features

- Keyboard shortcuts to annotate attributes without mouse
- Filtrate and find objects by a set of parameters
- Special mode to annotate attributes for any objects on an image
- Different types of attributes: boolean, choice, number, text

Object detection

Object detection in CVAT, the Computer Vision Annotation Tool, is a workflow for identifying and annotating specific objects within images or videos. It involves creating a new annotation task, uploading the media to be analyzed, and defining the object classes of interest, such as cars, pedestrians, or animals. Using the CVAT interface, users draw bounding boxes around these objects and assign appropriate class labels. After thorough review and correction, the annotated data can be exported for training object detection models, enabling them to recognize and locate objects in new visual data. This process is crucial for building systems that can automatically detect and track objects in various applications, including surveillance, autonomous vehicles, and image analysis.



Object Detection

Main Features

- Supported shapes:
  - bounding boxes,
  - polylines,
  - array of points,
  - polygons.
- Linear interpolation of bounding boxes between key frames on a video.
- Labels can be assigned to any object of any shape
- Configured attributes for any label (e.g. parked, color, etc...)

Object Tracking

Object tracking in CVAT, the Computer Vision Annotation Tool, is a technique used to follow and monitor the movement of specific objects within videos. This process involves creating annotation tasks in CVAT, where users annotate the initial location of objects in the first frame. As the video progresses, CVAT provides tools to track and annotate the same objects in subsequent frames, ensuring that the objects' positions are accurately recorded over time. This enables the creation of labeled datasets for training tracking algorithms, which can then be used to automatically track objects in new videos. Object tracking in CVAT is valuable in applications such as surveillance, video analysis, and autonomous navigation, where monitoring and understanding the movement of objects is essential.

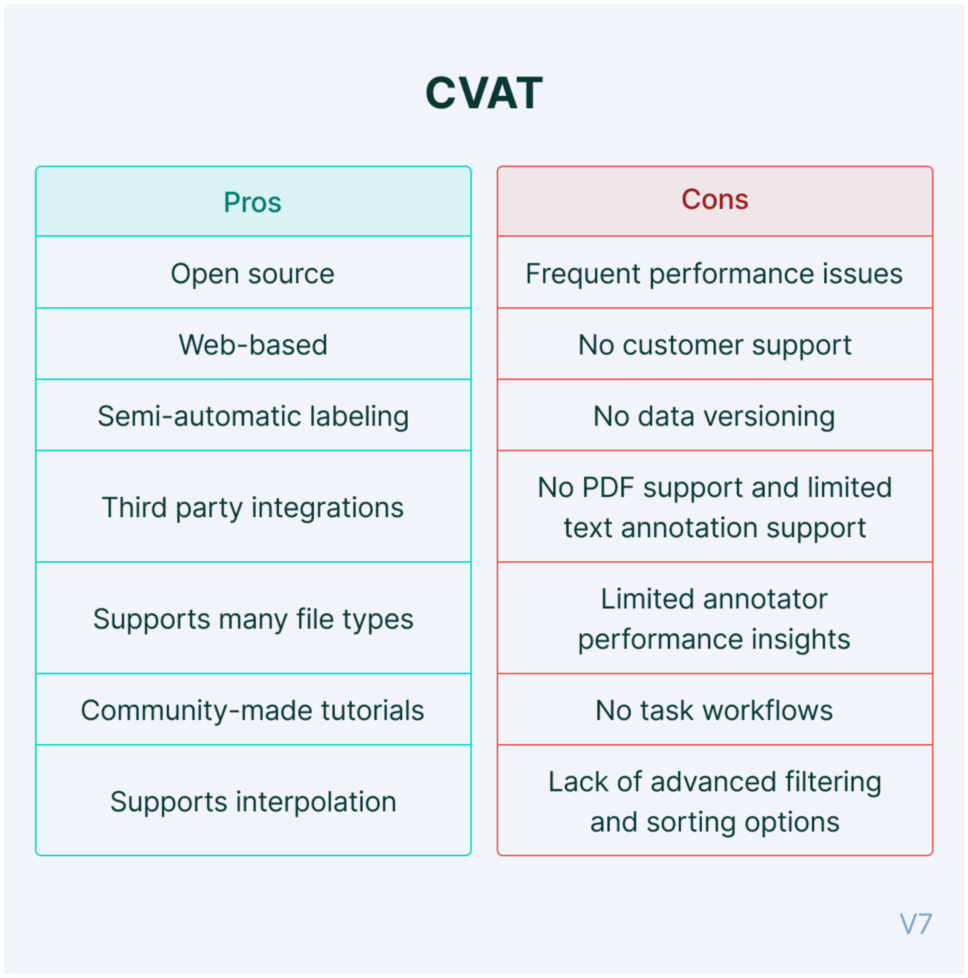
Image segmentation

Image segmentation in CVAT, the Computer Vision Annotation Tool, is a process of dividing an image into distinct regions or segments, each corresponding to a different object or part of the scene. Users typically define the object classes or categories they want to segment, and then they use CVAT's annotation tools to create precise outlines or masks around the objects in the image. These annotations indicate which pixels belong to each object or segment. Image segmentation is essential for various computer vision tasks, such as object recognition, medical image analysis, and scene understanding. CVAT streamlines the annotation process, making it easier to create high-quality labeled datasets for training and evaluating segmentation models.

Pose Estimation

Pose estimation in CVAT, the Computer Vision Annotation Tool, is the process of determining the spatial position and orientation of objects or people within images or videos. This technique is commonly used to understand and analyze the posture or position of individuals or objects in various applications, including robotics, sports analysis, and human-computer interaction. To perform pose estimation in CVAT, users create annotation tasks and annotate key points or joints on the objects or individuals of interest. These key points define the object's or person's pose, capturing information about their body or limb positions. The annotated data can be used to train pose estimation models, which can then predict the pose of objects or people in new images or video frames. CVAT simplifies the annotation process, making it easier to generate labeled datasets for training and evaluating pose estimation algorithms.

Pros & Cons of CVAT



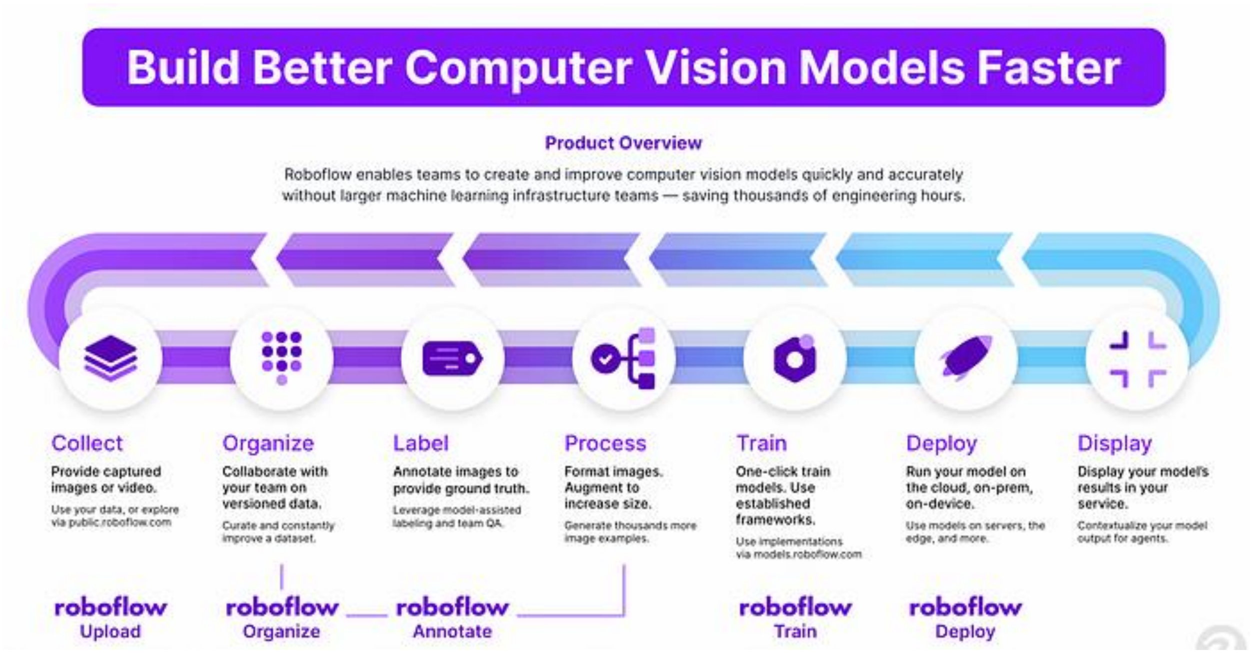
CVAT

Pros	Cons
Open source	Frequent performance issues
Web-based	No customer support
Semi-automatic labeling	No data versioning
Third party integrations	No PDF support and limited text annotation support
Supports many file types	Limited annotator performance insights
Community-made tutorials	No task workflows
Supports interpolation	Lack of advanced filtering and sorting options

Introduction to Roboflow

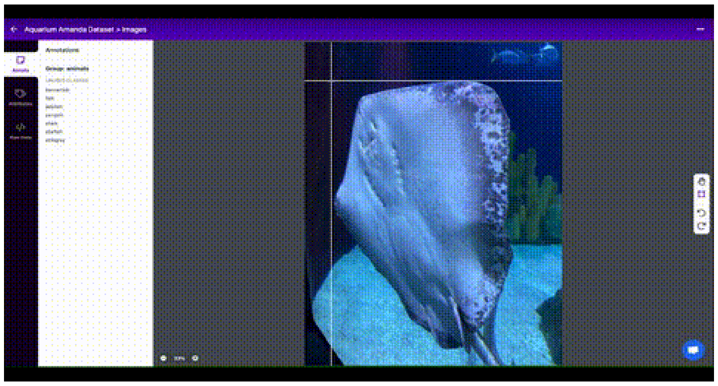
Roboflow is a platform that simplifies the process of developing computer vision applications. It provides tools and services to help developers and data scientists efficiently manage, preprocess, annotate, train, and deploy machine learning models for tasks like object detection, image segmentation, and more. With Roboflow, users can import and manage their image and video datasets, annotate objects within images, train custom models, and deploy those models for real-world applications. It offers features like data augmentation, model evaluation, and version control to streamline the computer vision workflow. Roboflow is a valuable resource for individuals and teams looking to leverage machine learning for various visual recognition tasks. This tool can do the following

- Dataset annotation
- Dataset Preprocessing
- Merge Project/Datasets
- Dataset Health check
- Dataset Export
- Train the model


Build Better Computer Vision Models Faster
Product Overview
Roboflow enables teams to create and improve computer vision models quickly and accurately without larger machine learning infrastructure teams — saving thousands of engineering hours.
Collect
Provide captured images or video.
Use your data, or explore via public.roboflow.com
Organize
Collaborate with your team on versioned data.
Curate and constantly improve a dataset.
Label
Annotate images to provide ground truth.
Leverage model-assisted labeling and team QA.
Process
Format images. Augment to increase size.
Generate thousands more image examples.
Train
One-click train models. Use established frameworks.
Use implementations via models.roboflow.com
Deploy
Run your model on the cloud, on-prem, on-device.
Use models on servers, the edge, and more.
Display
Display your model's results in your service.
Contextualize your model output for agents.
roboflow Upload
roboflow Organize
roboflow Annotate
roboflow Train
roboflow Deploy

Dataset annotation in Roboflow involves the process of adding labels or annotations to your images or videos to provide information about the objects or regions of interest within the data. Here's a brief overview of dataset annotation in Roboflow:

- **Data Import:** Start by uploading our image or video dataset to the Roboflow platform. Roboflow supports various data formats, making it easy to get your data into the system.
- **Annotation Tools:** Roboflow provides annotation tools that allow us to label objects, key points, or segments within your images or frames. We can draw bounding boxes around objects, mark key points on bodies, or outline segments in images, depending on the specific annotation needs of your project.
- **Labeling Objects:** Use the annotation tools to label objects of interest within our images. For example, if you're working on object detection, we can draw bounding boxes around cars, people, or any other objects we want to identify.
- **Annotation Quality Control:** Roboflow offers features to ensure the quality and accuracy of annotations. We can review and edit annotations to correct any errors or inconsistencies in the labeling.
- **Class Definitions:** Define the classes or categories that your objects belong to. For instance, if you're annotating animals, you can create classes like "cat," "dog," and "bird."
- **Metadata:** Add metadata and attributes to your annotations, such as object IDs, confidence scores, or any additional information that's relevant to your specific use case.
- **Export Annotations:** Once your dataset is fully annotated, you can export the annotations in various formats, such as COCO, Pascal VOC, or YOLO, making it compatible with different machine learning frameworks and models.



Dataset Preprocessing

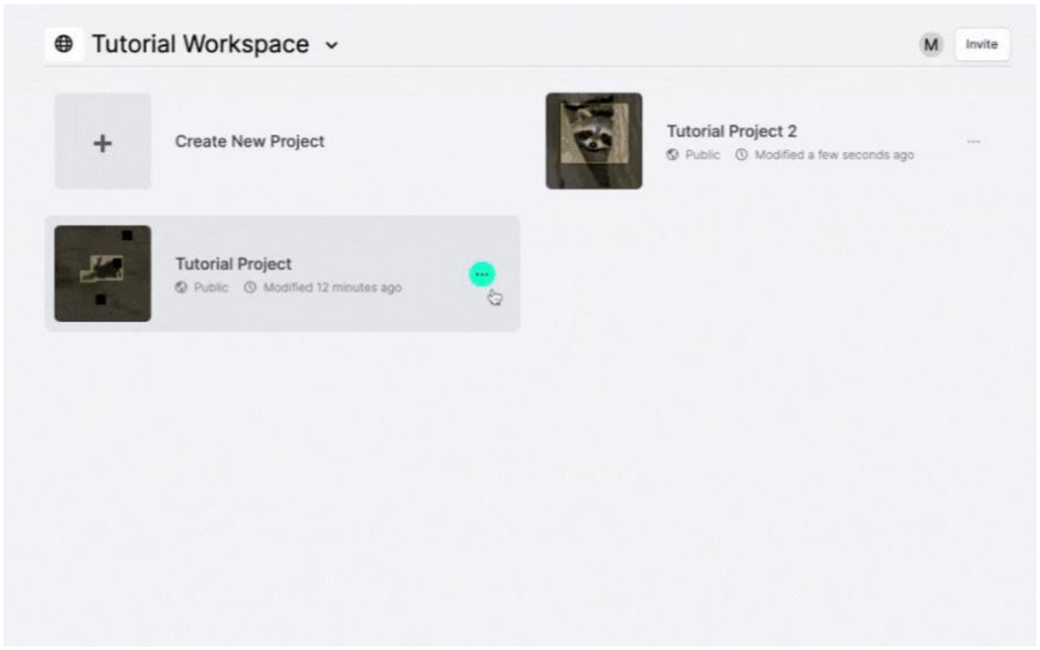
Before fitting the model, we can preprocess the dataset to get good training, validation, and testing results and to avoid overfitting the model. The following preprocessing can be applied by using this tool.

- Resizing.
- converting to grayscale.
- Augmentation.

Merge Project/Datasets:

This tool also provides the ability to store the different versions of the annotated dataset.

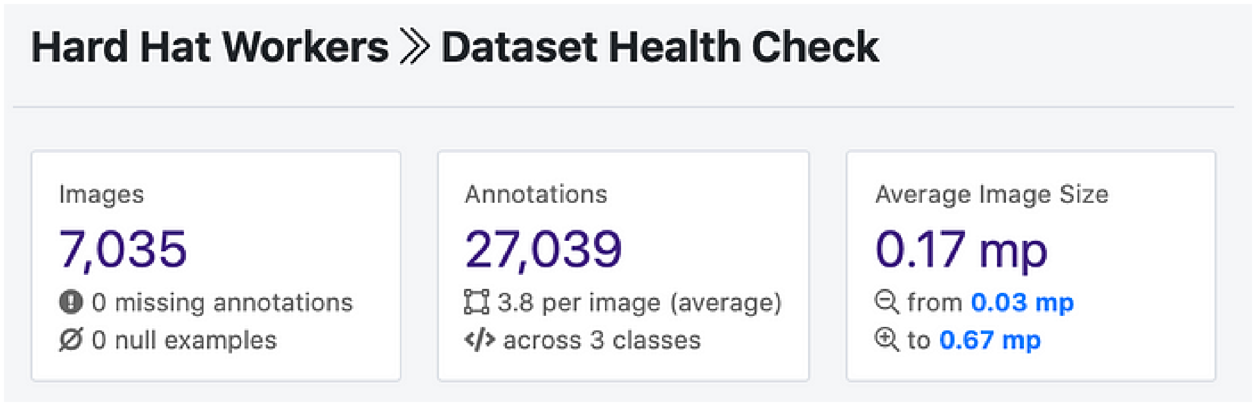
We can merge these versions into one dataset.



Dataset Health check:

We can visualize the dataset health, which can impact the accuracy of the model. We can visualize the following parameters of the dataset.

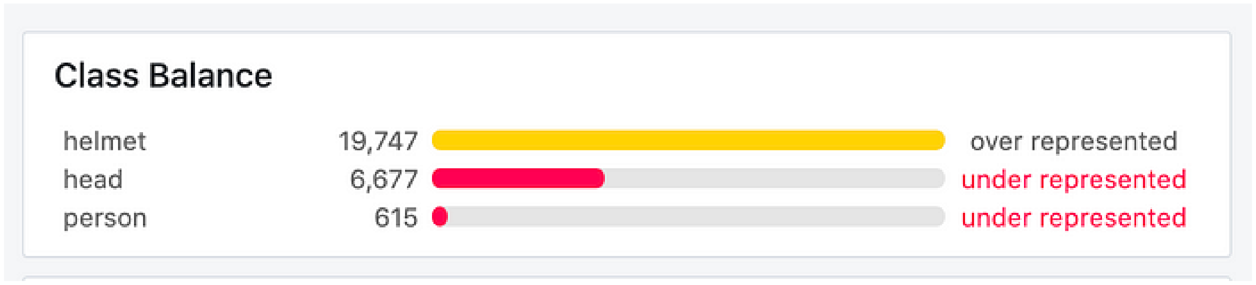
Number of images and annotations



Hard Hat Workers ≫ Dataset Health Check

Images	Annotations	Average Image Size
7,035	27,039	0.17 mp
0 missing annotations	3.8 per image (average)	from 0.03 mp
0 null examples	across 3 classes	to 0.67 mp

- Class Balancing



Class Balance

helmet	19,747	over represented
head	6,677	under represented
person	615	under represented

Train

- Roboflow offers an AutoML product called Roboflow Train.
- It is the easiest way to train and deploy a state-of-the-art object detection model on your custom dataset.
- It's one click. When our model has finished training you can see the following metrics on the dataset version page
  1. mean average precision.
  2. Precision
  3. recall, and more

How to Train YOLOv8 Object Detection on a Custom Dataset

YOLOv8 is the latest installment in the highly influential family of models that use the YOLO (You Only Look Once) architecture. YOLOv8 was developed by Ultralytics, a team known for its work on YOLOv3 and YOLOv5.

The steps to train a YOLOv8 object detection model on custom data are:

1. **Install YOLOv8 from pip**
2. **Create a custom dataset with labelled images**
3. **Export your dataset for use with YOLOv8**
4. **Use the yolo command line utility to run train a model**
5. **Run inference with the YOLO command line application**

How to Install YOLOv8

YOLOv8 can be installed in two ways: from the source and via pip. This is because it is the first iteration of YOLO to have an official package.

From pip (recommended)

To install YOLOv8 from pip, use the following command:

```
pip install ultralytics==8.0.0
or
pip install ultralytics
```

From source

We can also install the model from the source on GitHub using these commands:

```
git clone https://github.com/ultralytics/ultralytics
cd ultralytics
pip install -e ultralytics
```

The New YOLOv8 API

The developers of YOLOv8 decided to break away from the standard YOLO project design: separate train.py, detect.py, val.py, and export.py scripts. In the short term it will probably cause some confusion while in the long term, it is a fantastic decision.

This pattern has been around since YOLOv3, and every YOLO iteration has replicated it. It was relatively simple to understand but notoriously challenging to deploy especially in real-time processing and tracking scenarios. The new approach is much more flexible because it allows YOLOv8 to be used independently through the terminal, as well as being part of a complex computer vision application.

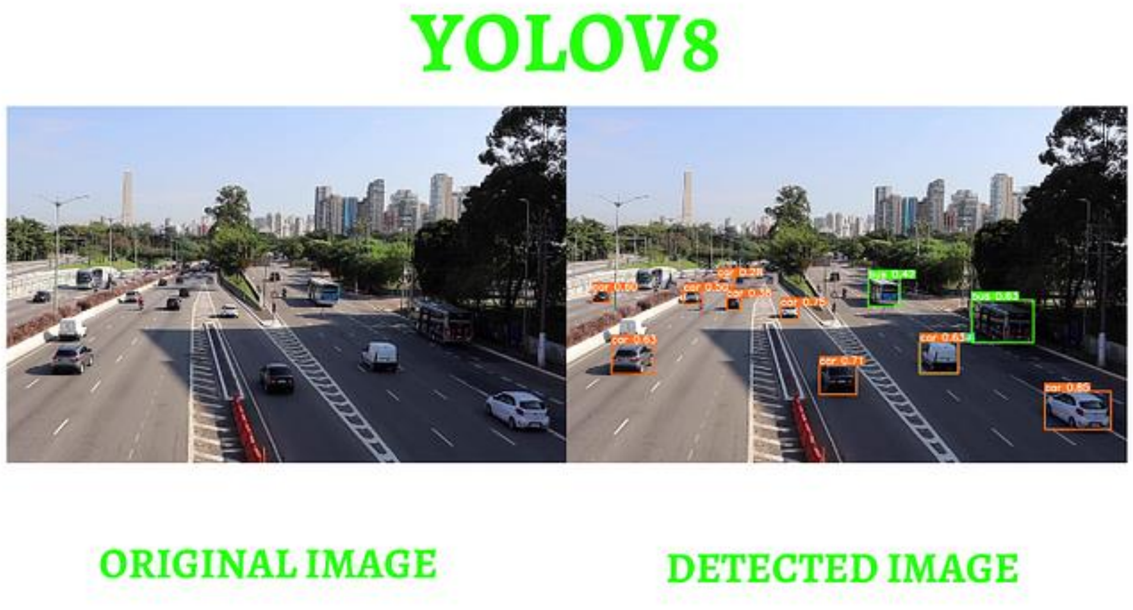
Pretrained Object Detection

#for image

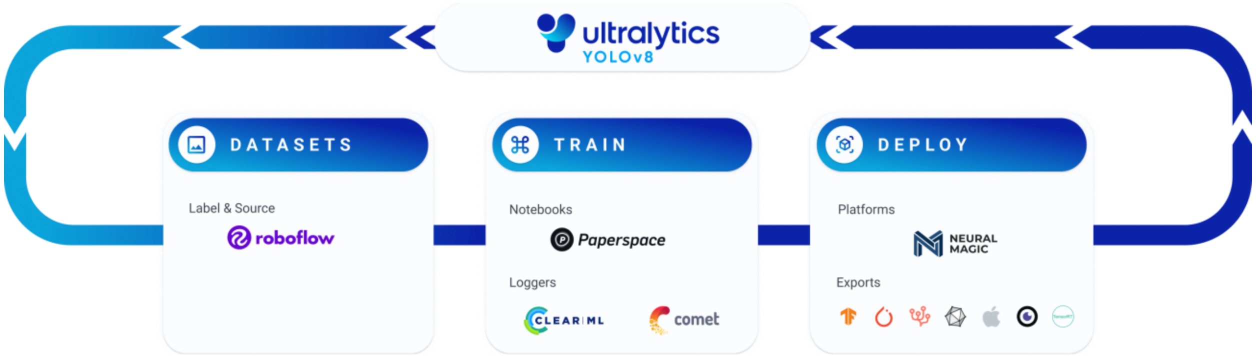
```
yolo task=detect mode=predict model=yolov8n.pt source="test.png"
```

#for video

```
yolo task=detect mode=predict model=yolov8n.pt source="test.mp4"
```



Model Training with Ultralytics YOLO



Training a deep learning model involves feeding it data and adjusting its parameters so that it can make accurate predictions. Train mode in Ultralytics YOLOv8 is engineered for effective and efficient training of object detection models, fully utilizing modern hardware capabilities.

Why Choose Ultralytics YOLO for Training?

Here are some compelling reasons to opt for YOLOv8's Train mode:

Efficiency: Make the most out of your hardware, whether you're on a single-GPU setup or scaling across multiple GPUs.

Versatility: Train on custom datasets in addition to readily available ones like COCO, VOC, and ImageNet.

User-Friendly: Simple yet powerful CLI and Python interfaces for a straightforward training experience.

Hyperparameter Flexibility: A broad range of customizable hyperparameters to fine-tune model performance.

Key Features of Train Mode

The following are some notable features of YOLOv8's Train mode:

1. Automatic Dataset Download: Standard datasets like COCO, VOC, and ImageNet are downloaded automatically on first use.

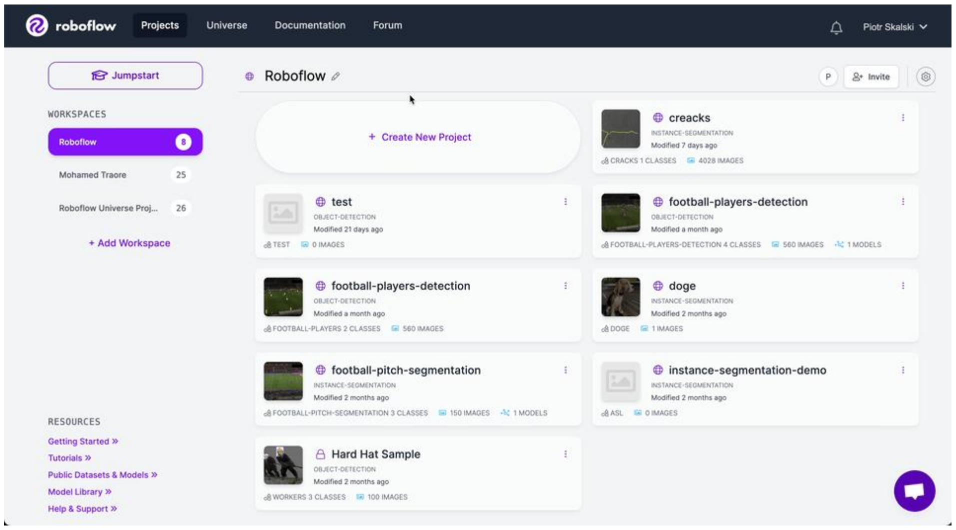
2. Multi-GPU Support: Scale your training efforts seamlessly across multiple GPUs to expedite the process.
3. Hyperparameter Configuration: The option to modify hyperparameters through YAML configuration files or CLI arguments.
4. Visualization and Monitoring: Real-time tracking of training metrics and visualization of the learning process for better insights.

Train YOLOv8 on Custom Data

Ultralytics YOLOv8 is the latest version of the YOLO (You Only Look Once) object detection and image segmentation model developed by Ultralytics. The YOLOv8 model is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and image segmentation tasks. It can be trained on large datasets and is capable of running on a variety of hardware platforms, from CPUs to GPUs.

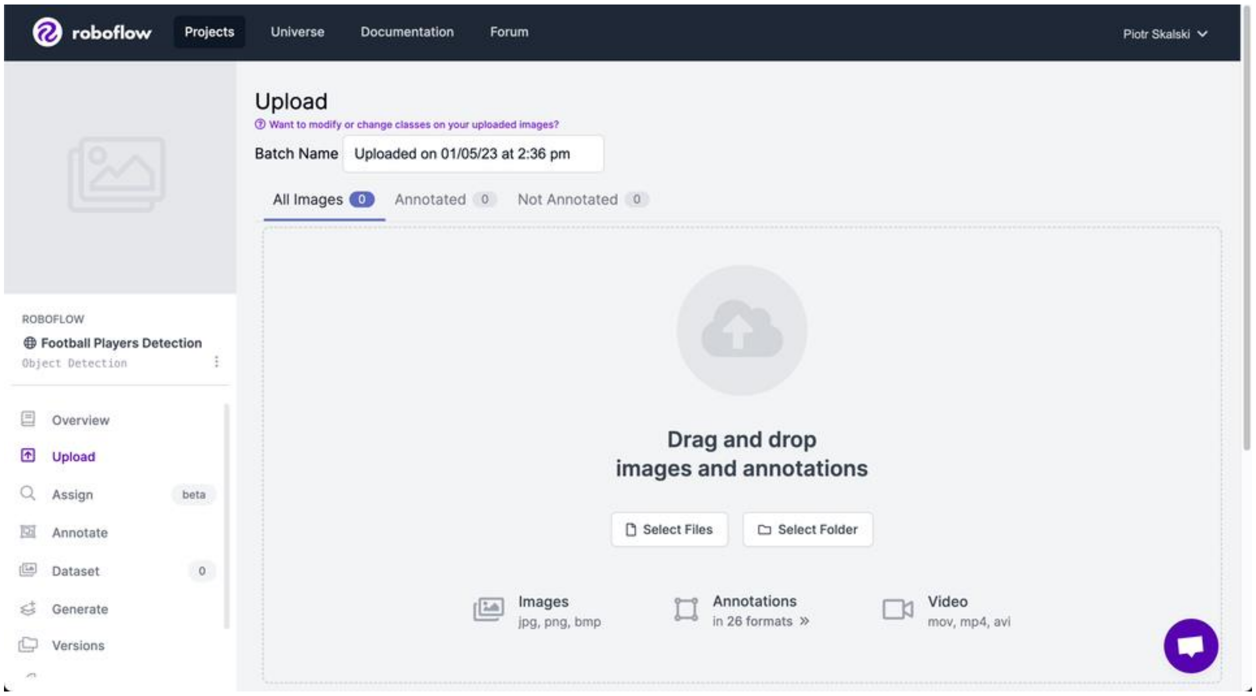
Step 1: Creating project

Before you start, we need to create a Roboflow account. Once we do that, we can create a new project in the Roboflow dashboard. Keep in mind to choose the right project type. In our case, Object Detection.

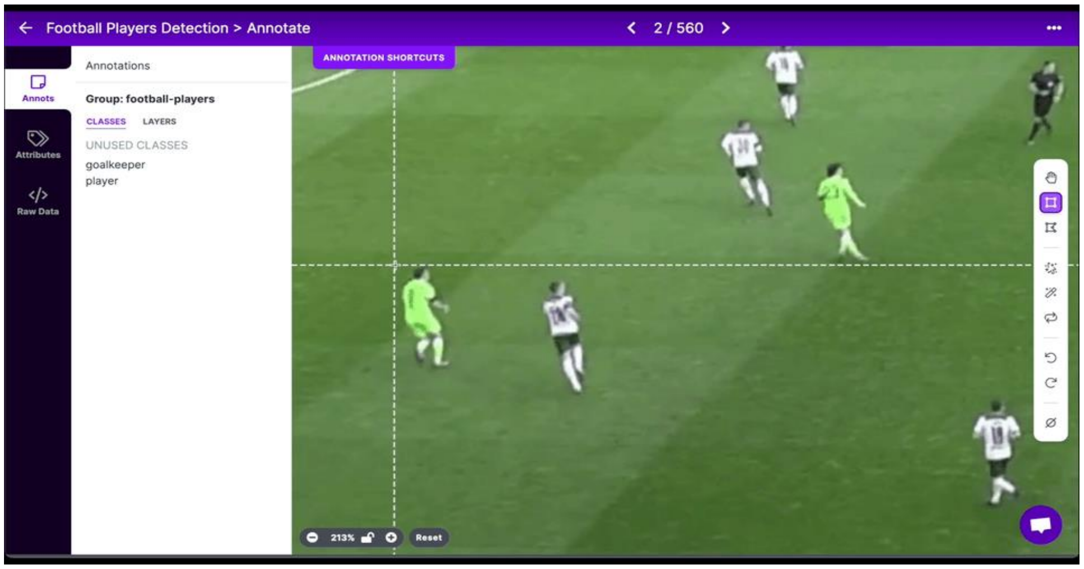


Step 2: Uploading images

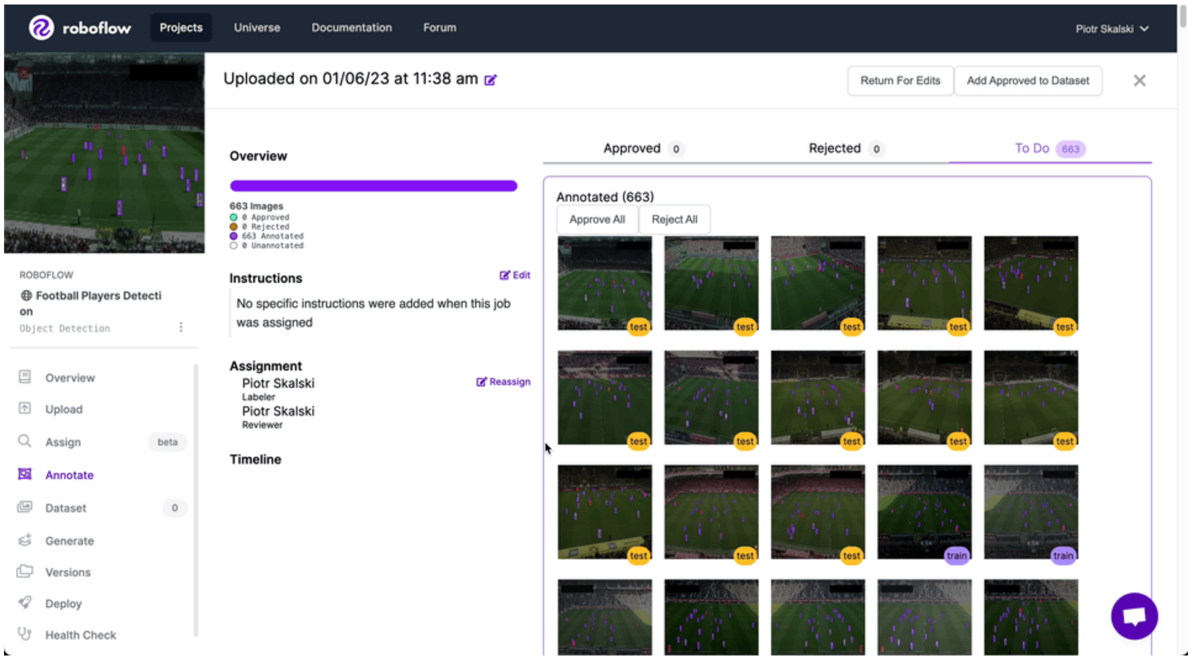
Next, add the data to newly created project. We can do it via API or through our web interface. If you drag and drop a directory with a dataset in a supported format, the Roboflow dashboard will automatically read the images and annotations together.



Step 3: Labeling



Step 4: Generate new dataset version



Uploaded on 01/06/23 at 11:38 am

Overview

663 Images
0 Approved
0 Rejected
663 Annotated
0 Unannotated

Instructions

No specific instructions were added when this job was assigned

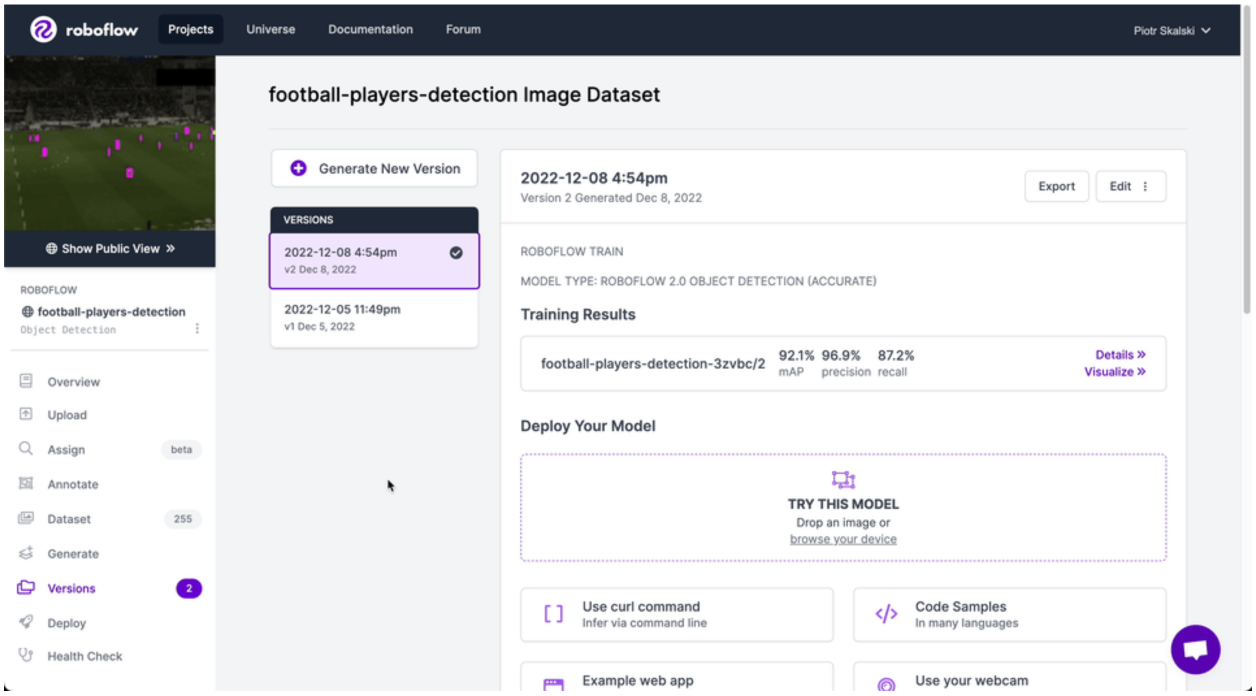
Assignment

Piotr Skalski
Labeler
Piotr Skalski
Reviewer

Timeline

Annotated (663)

Step 5: Exporting dataset



football-players-detection Image Dataset

2022-12-08 4:54pm
v2 Dec 8, 2022

2022-12-05 11:49pm
v1 Dec 5, 2022

2022-12-08 4:54pm
Version 2 Generated Dec 8, 2022

ROBOFLOW TRAIN

MODEL TYPE: ROBOFLOW 2.0 OBJECT DETECTION (ACCURATE)

Training Results

	mAP	precision	recall
football-players-detection-3zvbc/2	92.1%	96.9%	87.2%

Deploy Your Model

Use curl command
Infer via command line

Code Samples
In many languages

Example web app

Use your webcam