

Semi automatic CVAT pipeline for data annotation

Julien Audet-Welke¹

²CERVO Brain Research Centre, Quebec City, Canada October 2024

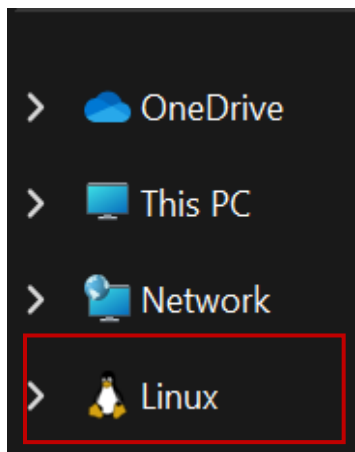
Overview

This package of scripts is designed to make adjusting tracks on a deep learning model (Precision track which is trained on animals as simple as possible. It streamlines the process of correcting automatically generated keypoints and bounding box annotations using CVAT (Computer Vision Annotation Tool). This program is an open-source data annotation tool that offers an SDK and API which were used to create the pipeline described in this document. This pipeline is only compatible with Windows computers and has not been developed for Mac or Linux.

Install Python

For the packages to function properly you will need to download Python which comes with pip install. Here is the link to install python: <https://www.python.org/downloads/>.

Install Windows Subsystem for Linux (WSL)



Some of the components of CVAT need Linux to run on your machine so you will need to install WSL which is available at the following link: <https://ubuntu.com/desktop/wsl>.


Once it is installed, if you go into your system folder you should see a little Linux icon should appear. It should look like the left image.

Next go to the start menu, search for WSL and double click the app to start it. Once the terminal opens, enter this command:

```
julien@LAPTOP-8QJM408V:~$ pip install ubuntu
```

This will install the Linux operating system that is required. Next you can close the terminal and begin the CVAT installation tutorial.

Important remark

For this entire document I am using a custom command prompt that will have a different visual appearance than the standard Windows command prompt. This visual overlay does not change the commands in any way. Simply enter the command I write starting from the two right pointing arrows: 

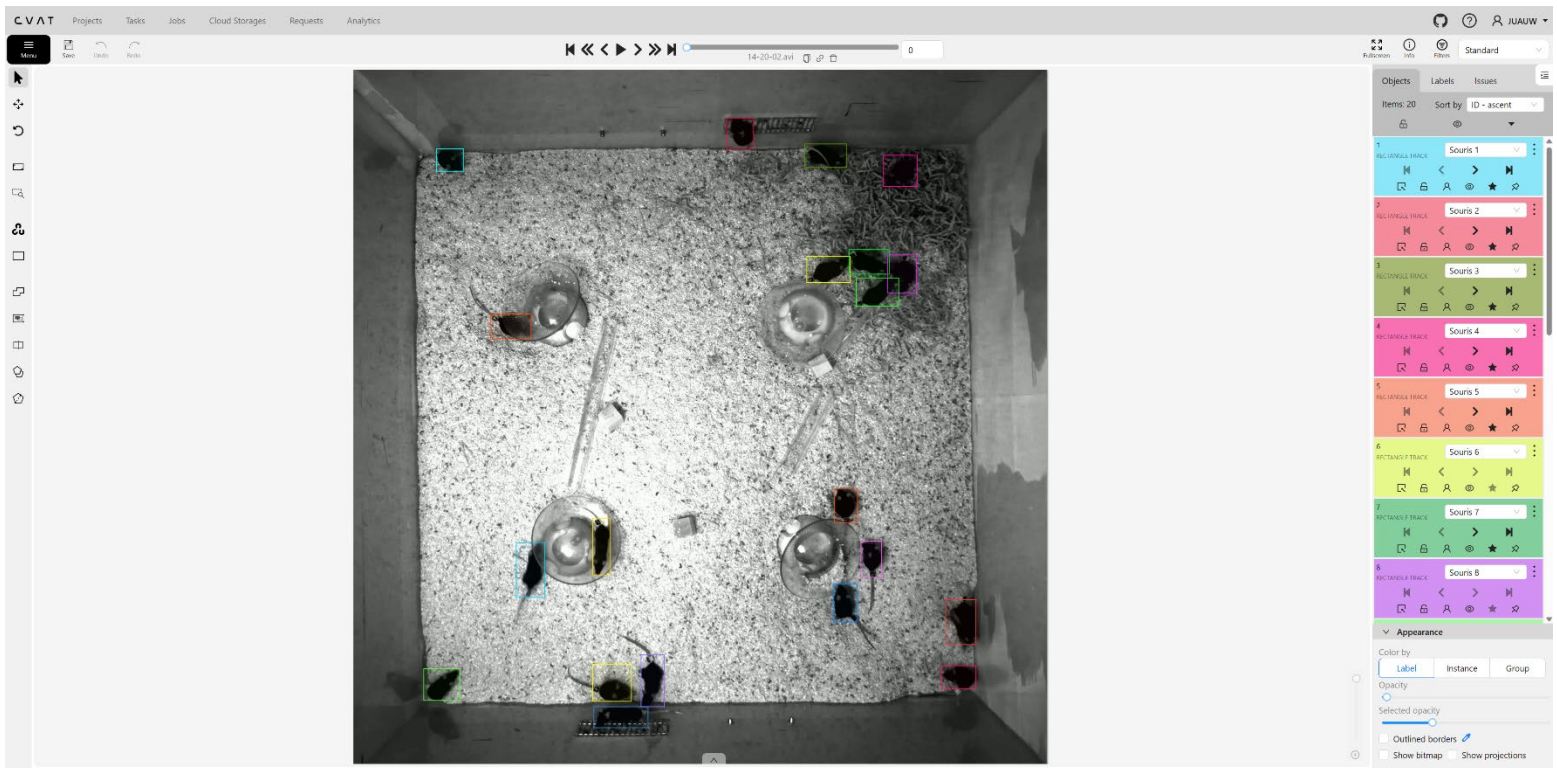
Local CVAT installation guide

YouTube tutorial: <https://www.youtube.com/watch?v=hyvNFuc06qQ>

Summary of the tutorial:

1. Install Git (file manager for code)
2. Install Docker (CVAT runs on a container)
3. Copy the CVAT repository on your machine (git clone)
4. Initialise docker
5. Install WSL
6. Install CVAT on docker
7. Create a username and password
8. Launch CVAT by typing localhost:8080 into a web browser

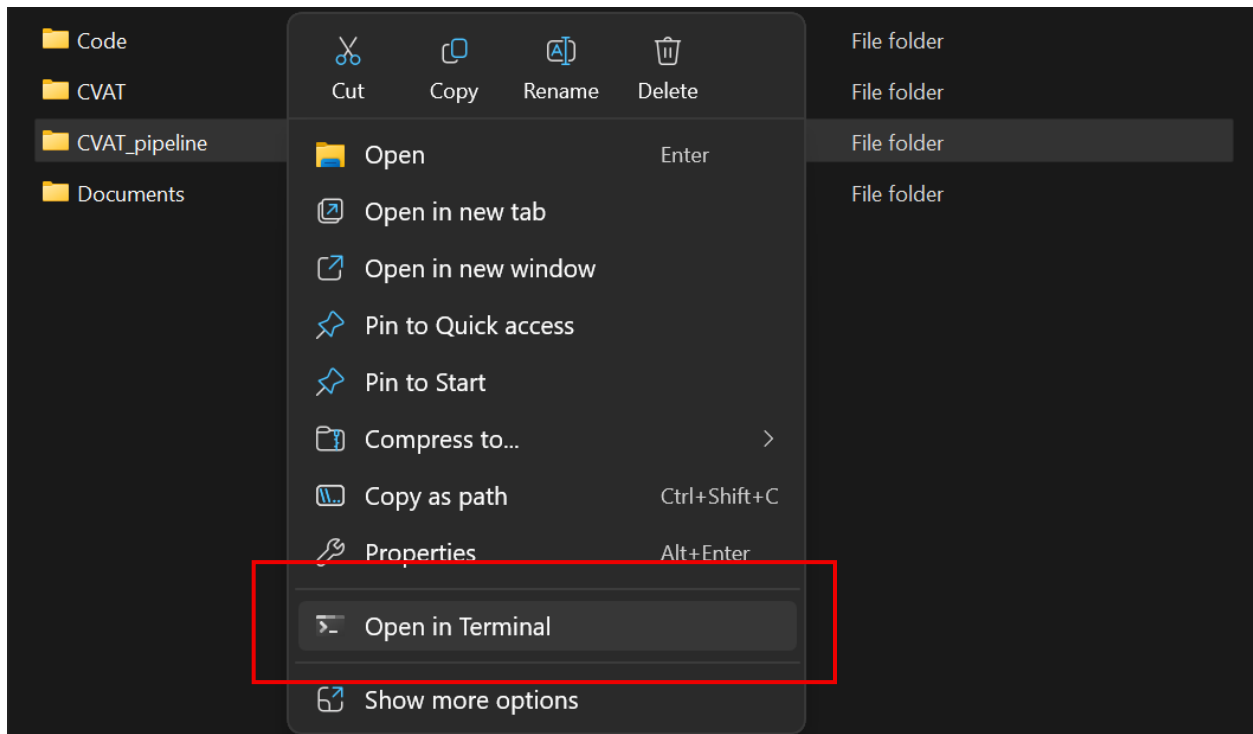
CVAT work environment example



Link to the GitHub repository: https://github.com/juauw/CVAT_pipeline.git

Step by step guide

Once you have CVAT up and running locally you can download the GitHub repository that was mentioned above. The instructions are the same as the tutorial to install CVAT. The first step is to open your terminal in the copied CVAT folder (CVAT_pipeline). Here is what that should look like:



This will open the terminal in the correct folder and should allow all the scripts to work. Next step to running the pipeline is to install all the dependencies and packages of the environment. You can do this by writing the following in your terminal (opened in the CVAT_pipeline folder):

```
>> pip install -r requirements.txt
```

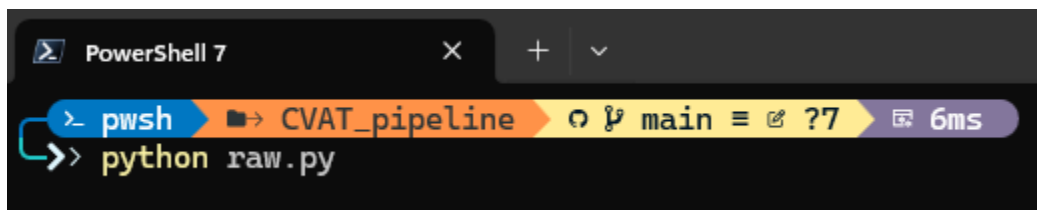
Afterwards you can go ahead with starting the workflow. It splits into different parts depending on what your input data is. In all cases, you will need to generate an SVG file. An SVG file is an image file that contains the shape of a given skeleton and all the names of each point.

Here are the steps to generate the SVG file:

1. You will need a `metadata.py` file that contains the skeleton information for your animal
2. Next you will execute the `raw.py` script directly or via `auto_task_SDK.py`
3. The script will generate a file named `skeleton.svg` which you will input into CVAT later

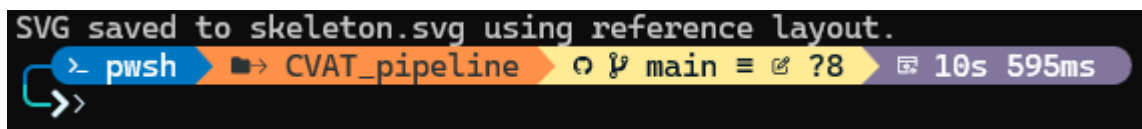
Details about **raw.py**

This script loads the `metadata.py` file and uses it as a reference to build an SVG image. By default, it uses mouse keypoints extracted from the metadata, but it can be expanded to fit any animal. Here is how to execute it directly:



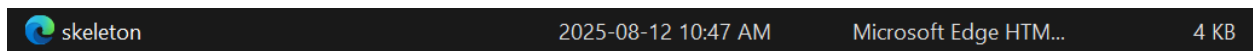
```
PowerShell 7
> pwsh -> CVAT_pipeline -> main ?7 6ms
>> python raw.py
```

If the command is successful, the command line should display this:



```
SVG saved to skeleton.svg using reference layout.
> pwsh -> CVAT_pipeline -> main ?8 10s 595ms
>>
```

And it should look like this in your folder browser:



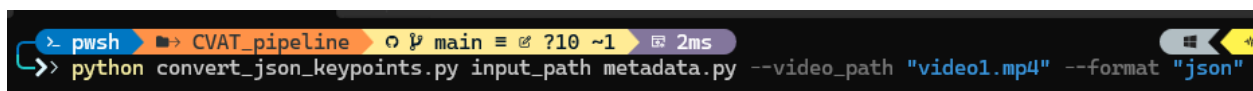
The SVG will be generated into the same folder as the script. This SVG will be put into a field in CVAT once the **auto_task_SDK.py** script or the **converted_json_bbox.py** script has been executed.

At this point in the workflow, it is important to know if you want to annotate keypoints or bounding boxes in CVAT. If you want to annotate keypoints you need to execute the **convert_json_keypoints.py** script directly. Here is how these two scripts work and how to

execute them. Note that if you don't have any annotations and are annotating from scratch you can skip this script and go directly to **auto_task_SDK.py**

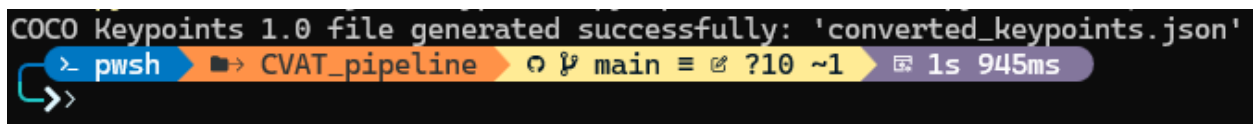
Details about **convert_json_keypoints.py**

This script takes a file named **metadata.py** and a **.csv** or **.json** input file and produces a file named **converted_keypoints.json** or **converted_keypoints.xml** which is compatible with the CVAT keypoints import module. Here is how to execute the script directly:



```
> pwsh -> CVAT_pipeline -> main ?10 ~1 2ms
>> python convert_json_keypoints.py input_path metadata.py --video_path "video1.mp4" --format "json"
```

For this command the mandatory arguments are the **input_path** and **metadata.py**. The **input_path** refers to the name of the **.csv** or **.json** annotation file which should be placed in the same folder as this script. The **--video_path** optional argument needs to be used only when the input format is **CSV**. The argument should be the name of the video corresponding to the **CSV** annotation file and it's used to calculate the resolution of the frame which is needed to create the correct annotations. The video should also be in the same folder as the **convert_json_keypoints.py** script. The **-format** optional argument is used to select the desired output format. The options are **xml** or **json**. Here is what the console should display if the script ran correctly (for the example above):



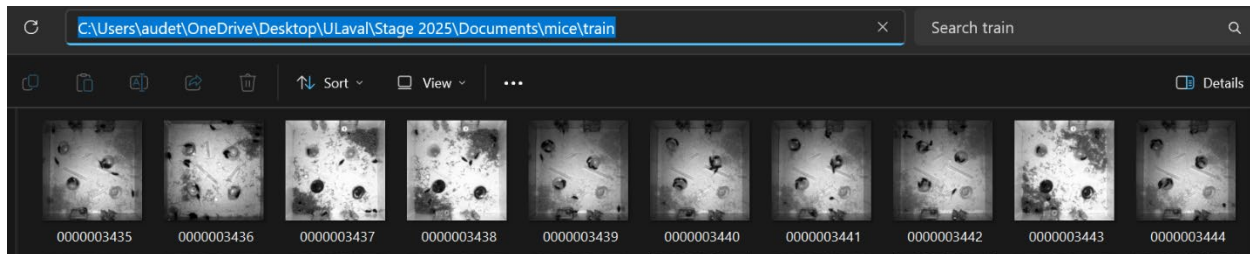
```
COCO Keypoints 1.0 file generated successfully: 'converted_keypoints.json'
> pwsh -> CVAT_pipeline -> main ?10 ~1 1s 945ms
>>
```

The script will output a file named **converted_keypoints.json**, this file will be uploaded manually into a CVAT task once the **auto_task_SDK.py** script has been executed. The alternate **xml** output file is used only if there is a chance that the tracks have switched in the annotations and there needs to be a verification completed since **COCO** keypoints do not support persistent tracks in CVAT.

Details about `auto_task_SDK.py`

Supplemental video: <https://www.youtube.com/watch?v=88TVX58GHlc>. This script is central to the keypoints annotation pipeline. It creates a task in CVAT and automatically imports the images associated with the task. The annotations must be uploaded manually because of a limitation in CVAT. This script also executes `raw.py` automatically. Here is how to use the program:

1. You need to enter your username and password into the command line
2. You can choose to enter a task name, otherwise it will be "Keypoint annotation"
3. Select the folder where your images et videos are and copy its path:



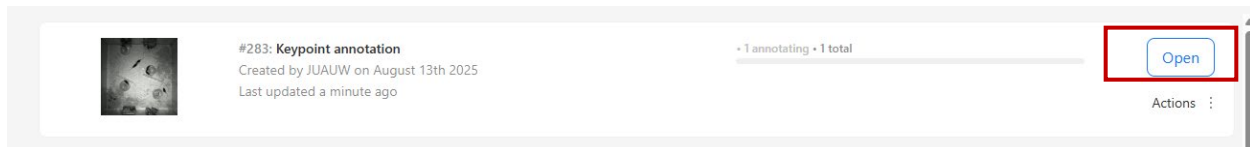
Here is what the input into the command line should look like:

```
pwsh -> CVAT_pipeline -> main 75 2ms 100% 13,11:20
>> python auto_task_SDK.py --username "JUAUW" --password "Yetisb130^^" --folder "C:\\Users\\audet\\OneDrive\\Desktop\\ULaval\\Stage 2025\\CVAT_pipeline"
```

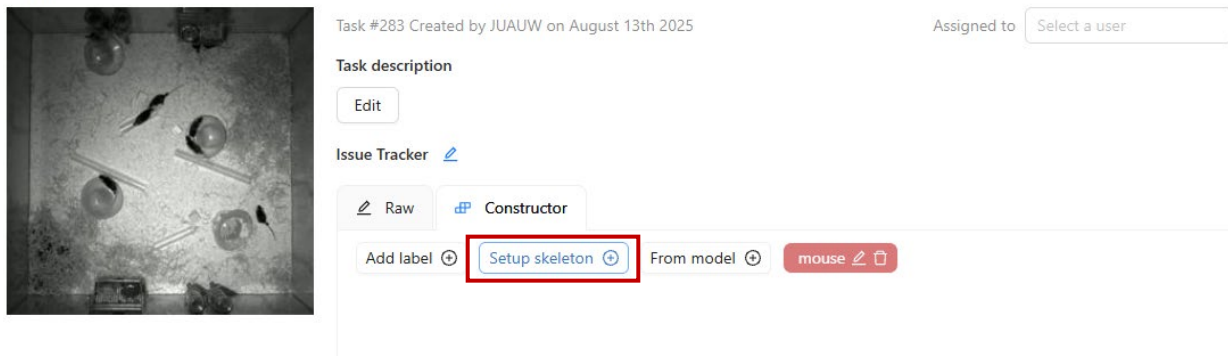
If the command has run successfully, it should display this result:

```
Running raw.py...
SVG saved to skeleton.svg using reference layout.
Connected to CVAT server.
Extracting frames from video: video1.mp4
Uploading 1301 extracted frames.
Uploading data: 100% | 719M/719M [01:08<00:00, 11.1MB/s]
Task 'Keypoint annotation' created with ID: 283
```

4. Once this step is complete you will find the task in the CVAT UI where you will have to upload the SVG and the annotations manually. Here are the steps shown visually:

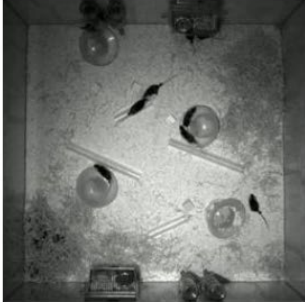


Keypoint annotation



5. You will need to delete the label and create a new one, it needs to keep the same name.
Next you need to upload the SVG:

Keypoint annotation [🔗](#)



Task #283 Created by JUAW on August 13th 2025


Assigned to

Task description

Issue Tracker [🔗](#)

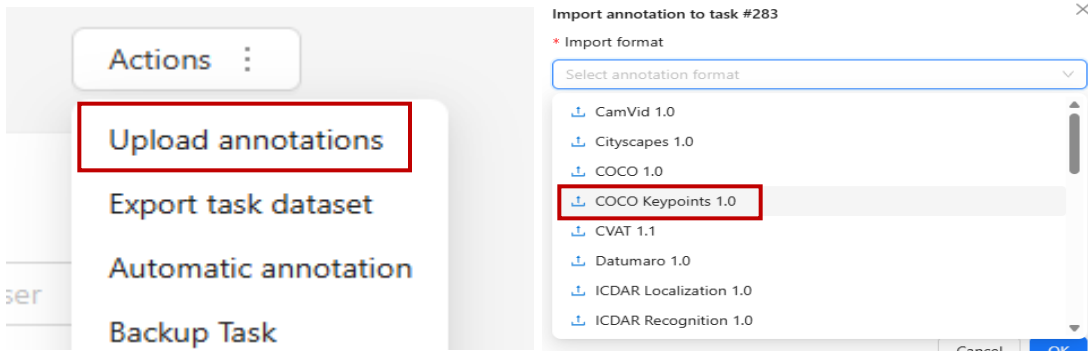
Note: If you labelled your task differently the task will not be named “Keypoint annotation”

☒

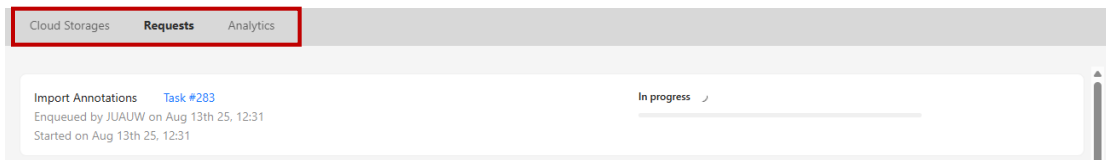


Note: If you are starting annotation from scratch, you can go back to tasks and click on the Job hyperlink to directly get started with annotations.

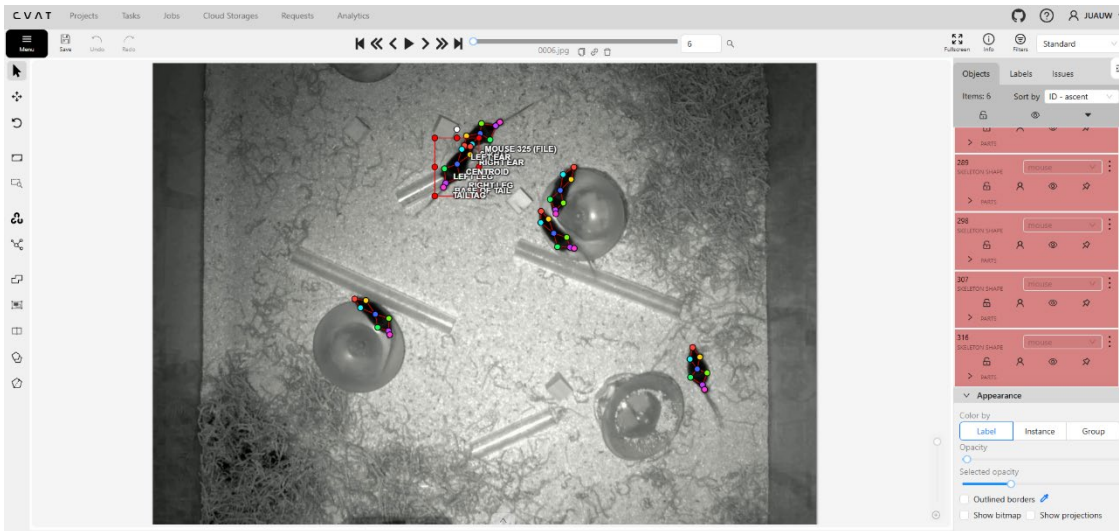
- A skeleton will show up once you upload the SVG that is in the CVAT_pipeline folder. Once this is done you can click continue and go back to the task. Next you will upload the file named **converted_keypoints.json**:



- This is the page where the annotations will upload:



- Once the annotations are uploaded the interface should look like this:



At this stage you can annotate keypoints from scratch or modify existing keypoints, so they are more precise. Once the annotation is completed you will need to export the annotations manually since the CVAT SDK does not currently support exports.

How to export keypoints or bounding boxes

CVAT pipeline overview

