{DATA SCIENCE INTERNSHIP PROJECT REPORT FORMAT}

AERIAL IMAGE ANALYSIS FOR SMART CITY DEPLOYMENT

(General Purpose Image/Video Annotation and Analysis for Large Scale Computer Vision)

Vaibhav Mahato,

CSE2021021 and RCC INSTITUTE OF INFORMATION TECHNOLOGY

Project Guide / Mentor Name: Dr. SUJOY BISWAS

Period of Internship: 14th Jan 2025 - 30th April 2025

Report submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata.

Introduction: Aerial Image Analysis for Smart City Development

- **Leveraging Computer Vision for Smart City Planning**
- Aerial imagery provides a high-level spatial overview essential for modern urban development.
- By integrating computer vision techniques with annotation tools like **Label Studio**, large-scale image data can be analyzed efficiently to derive actionable urban insights.

  Urban Littering Detection (Polygon Annotation)
- Identifying and mapping urban litter zones helps municipalities pinpoint areas of poor waste management.
- **Computer vision** models trained on **polygon-annotated** litter regions enable automatic recognition of waste accumulation hotspots in future datasets.
- These insights can drive smart sanitation planning, optimize waste collection routes, and enhance civic monitoring systems.

  Rooftop Tank Identification (Bounding Box / Polygon Annotation)
- Rooftop tanks are crucial indicators of urban water infrastructure, especially in high-density zones.
- By annotating rooftop tanks using **bounding boxes** or **polygons**, models can learn to detect water storage units across residential and commercial zones.
- This supports smart water management initiatives, including monitoring capacity distribution and planning for decentralized water systems.

  Waterbody Segmentation (Polygon Annotation)
- Urban waterbodies (lakes, ponds, reservoirs) play a vital role in flood control, biodiversity, and climate regulation.

- Through **polygon-based segmentation**, computer vision can delineate these regions and monitor their shrinkage, contamination, or encroachment over time.
- Smart city systems can integrate this data to inform green infrastructure planning and ecological restoration.

Outcome for Smart City Ecosystems

- This multi-layered aerial image analysis pipeline contributes to a **data-driven urban governance model**.
- It empowers planners with spatial intelligence to build **cleaner**, **safer**, and **resource-optimized** cities.
- By correlating environmental factors with visual indicators, the approach supports **sustainable urban growth** aligned with smart city objectives.

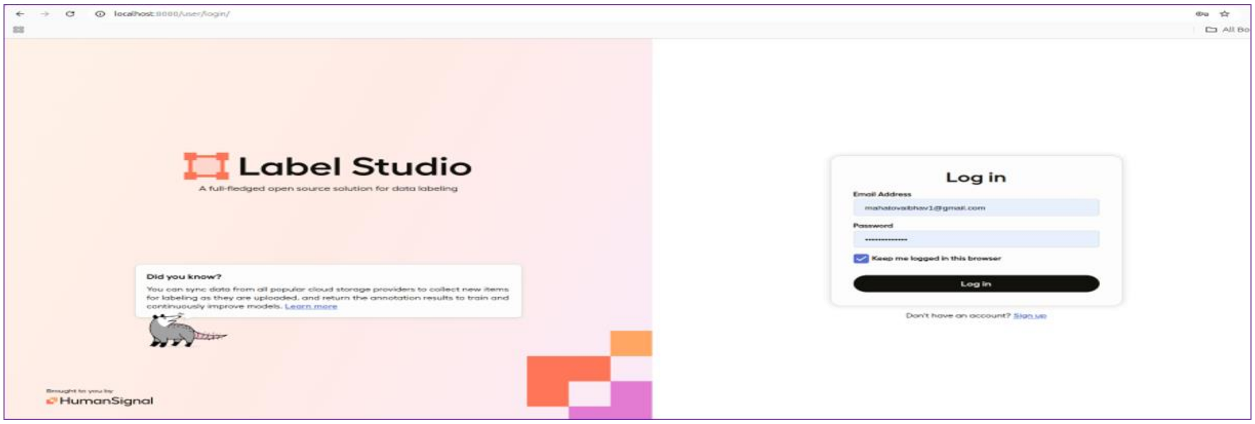
The objective of this project is to harness aerial imagery and computer vision techniques to generate actionable insights for smart city planning and environmental management. Through the annotation of urban features using Label Studio, we aim to achieve the following:

- **Detect and monitor urban littering patterns** using polygon annotations to identify areas with high waste accumulation, supporting better waste disposal planning and public sanitation strategies.
- **Map and analyze rooftop water tanks** via bounding boxes or polygons to assess the distribution of water storage infrastructure, which aids in evaluating urban water supply and potential rooftop utility.
- **Accurately delineate natural and man-made waterbodies** using polygon annotations to monitor their size, health, and seasonal changes, contributing to effective water resource management and flood risk assessment.
- **Illustrate the power of aerial image analysis combined with computer vision** in automating urban feature recognition, significantly reducing manual monitoring time and improving data-driven decision-making for city administrators.
- **Build a foundation for training machine learning models** that can perform real-time detection of critical urban features, enabling proactive urban planning and smarter city services. These models can be further used in real estate financing , other other uses in capital market.

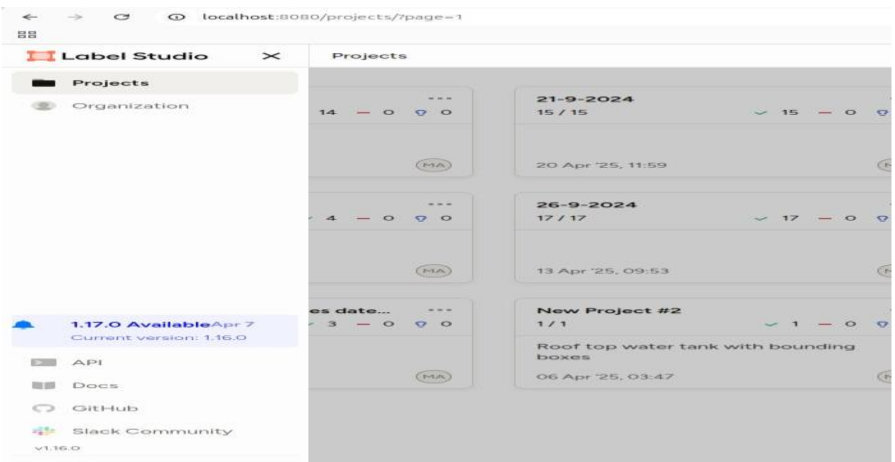
  Hypothesis: Automated detection of specific urban features (litter, tanks, waterbodies) from aerial images can achieve accuracy levels comparable to manual annotation, with significantly reduced effort and time.

METHODOLOGY

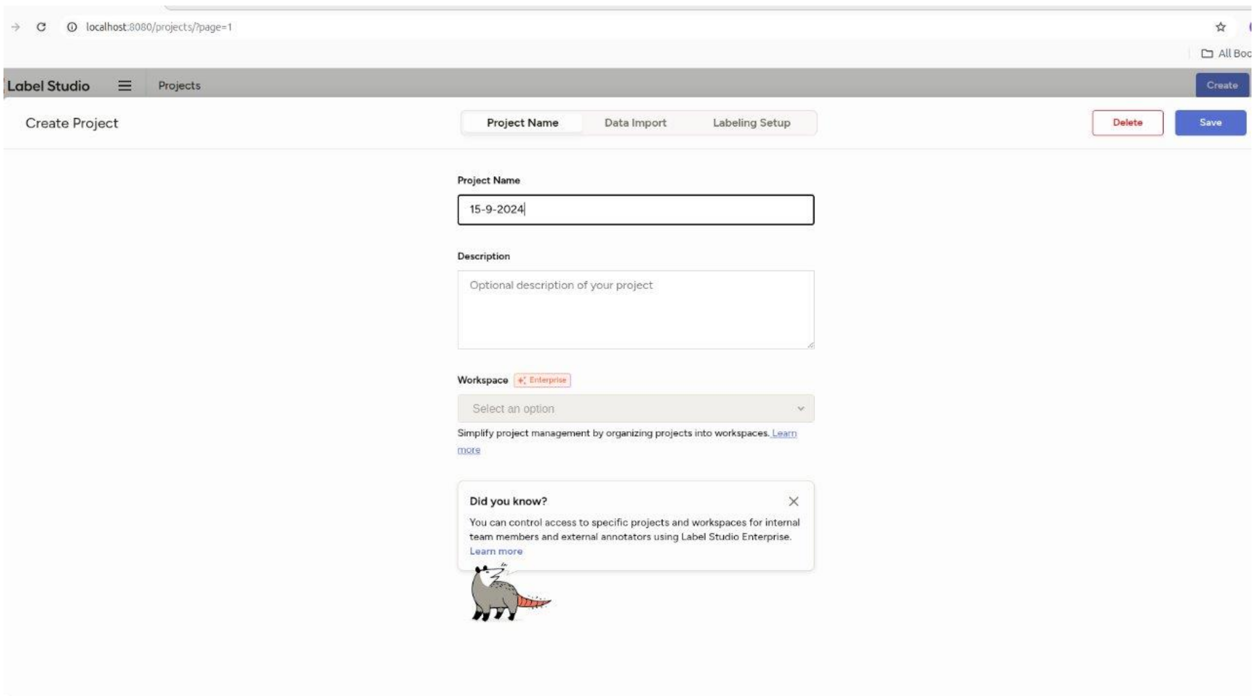
I have been provided with lot of aerial images i.e, high-resolution aerial imagery from drones, satellites, or mapping services. Then I uploaded them at label studio Ensure that label studio is running locally which was installed in your system.



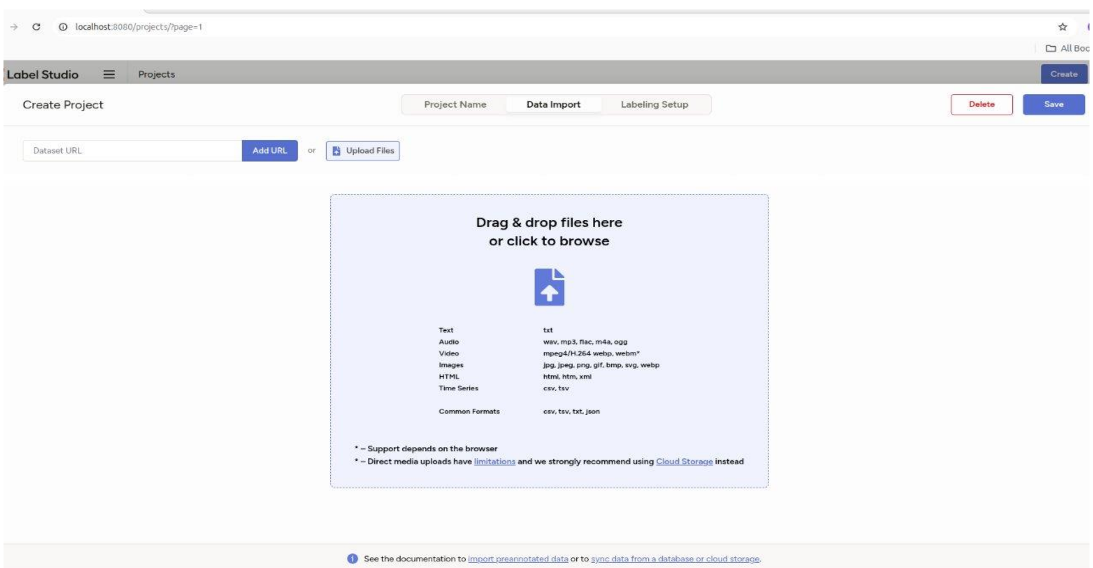
After login , we go to the projects section by clicking on the top left corner three vertical lines.



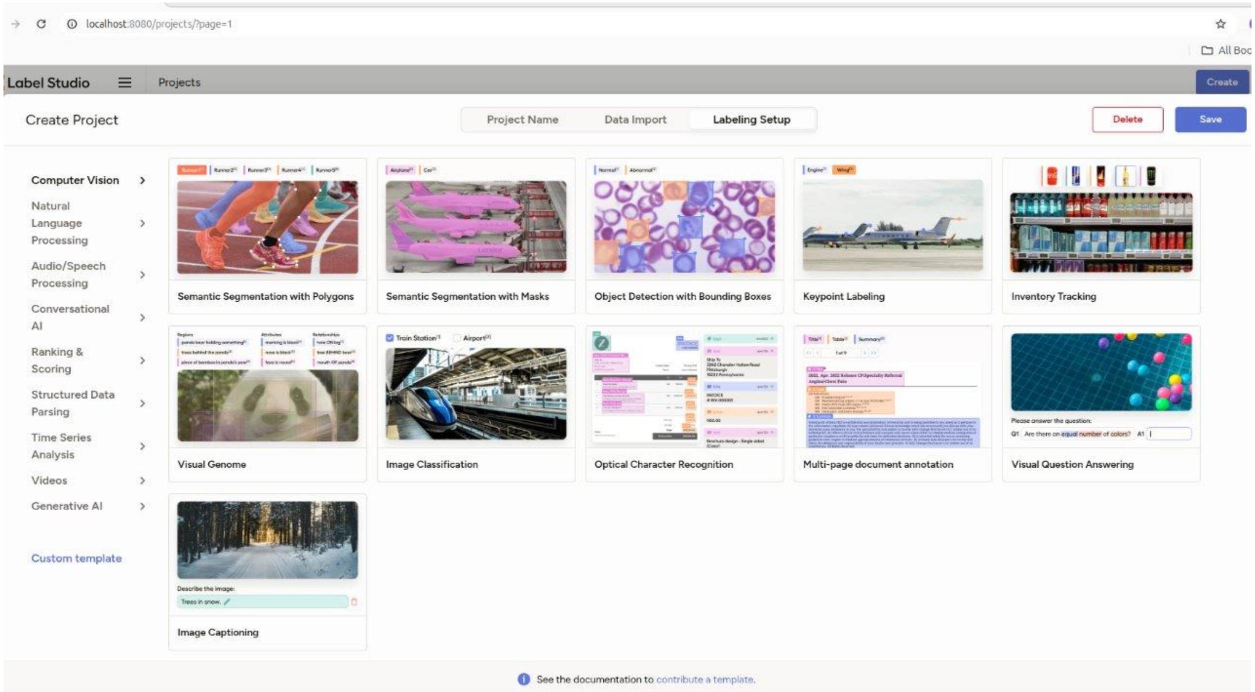
Then we moved on to the top right corner and clicked on the 'create' option. A new interface gets opened which is shown below and we fill the details in the first page of that interface (Project Name). For example I have filled my project name as 15-9-2024. Any additional information should be filled in the description section .



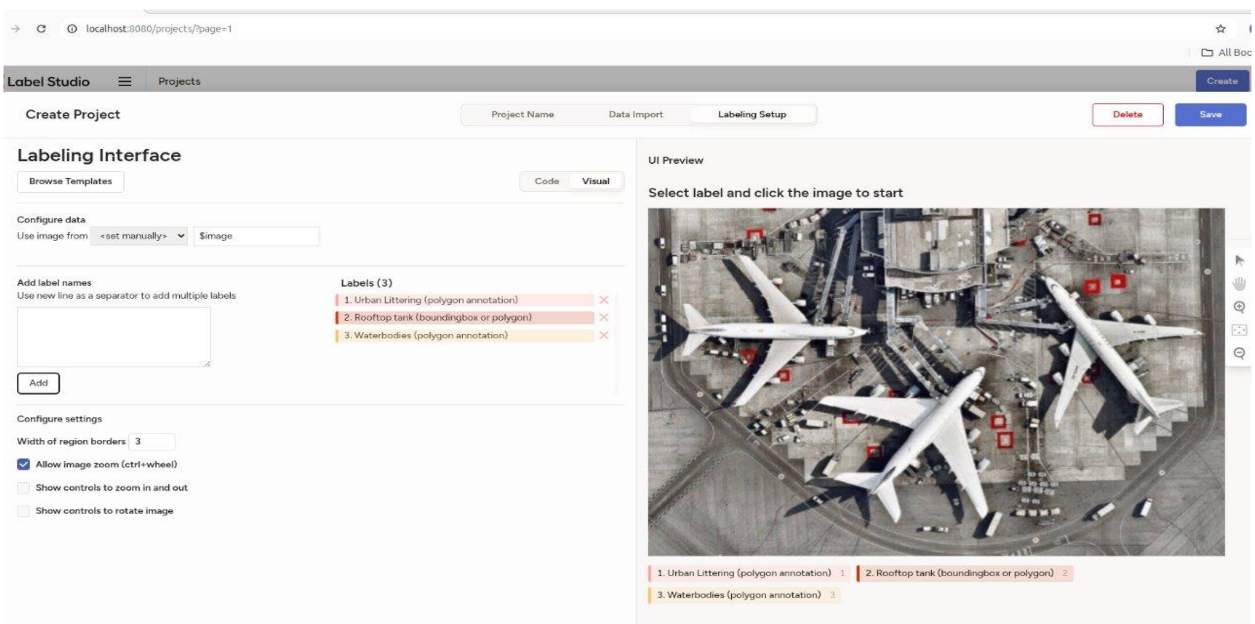
We then moved in the next interface by just clicking on the Data Import, and on clicking the add URL or Import Files we bring the files or JPGs in which annotation works need to be done.



Then we move on to the next interface by clicking Labelling Setup and in the opening page itself we get different options to choose in which domain we are working and what kind of annotation we want to perform . For example, I choose Computer Vision and under it Semantic segmentation with polygon.

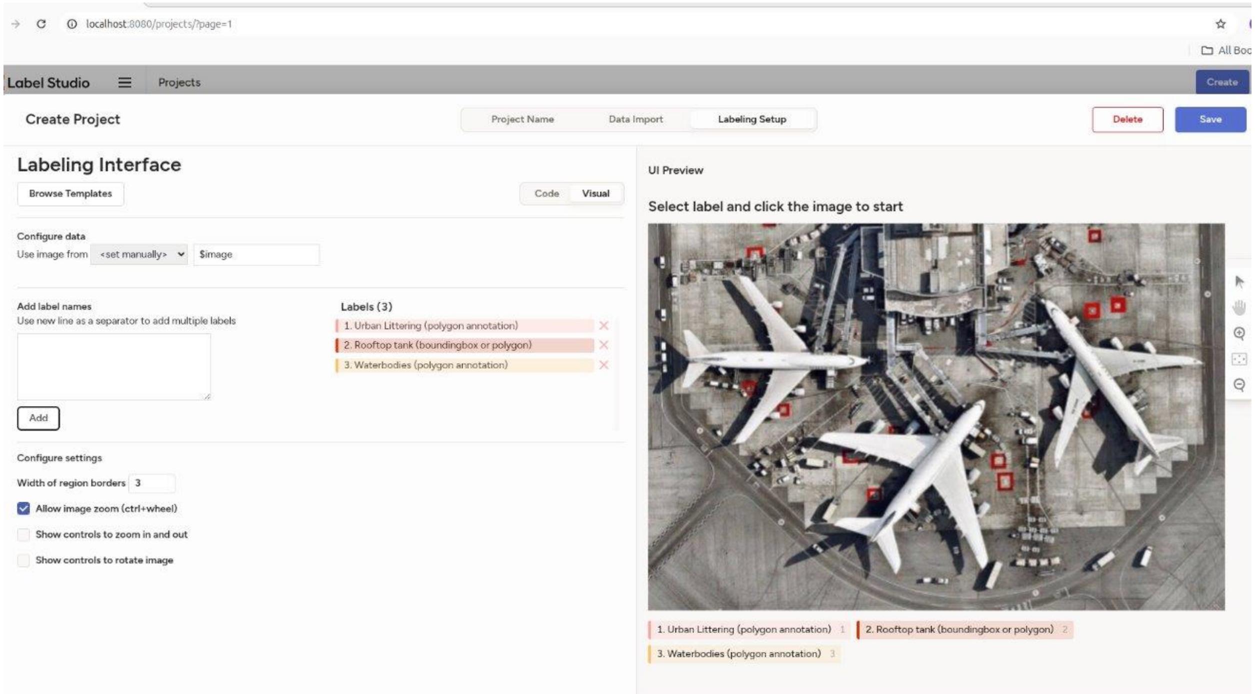


And when I clicked on what kind of work I want to do then it opens a interface as shown below.



In the add labels section we added different categories of annotations that you want to do and click on add.

For example I added three categories of annotations I want to do on any image.



THE FOLLOWING ARE THE EXAMPLES OF THE DIFFERENT TYPES OF ANNOTATIONS DONE

Waterbody Annotation

Urban Littering Annotation

Rooftop Tank Annotation

5. 

Data Analysis and Results

I have annotated the images based on

1. Urban Littering – Polygon Annotation

Annotation Type: Polygon Annotation

Purpose: Polygon annotations are ideal for capturing the irregular shapes and sizes of litter items such as plastic bottles, wrappers, and cans scattered in urban environments.

Label Studio Implementation:

- **Tool Used:** PolygonLabels tag Label Studio
- **Process:** Annotators click sequentially around the perimeter of each litter item to create a closed polygon that precisely outlines its shape.
- **Advantages:**
- Provides high-precision annotations for objects with complex boundaries.
- Enables detailed semantic segmentation, which is crucial for training models to detect and classify various types of litter.

2. Rooftop Tank – Bounding Box or Polygon Annotation

Annotation Types: Bounding Box and Polygon Annotation

Purpose: Rooftop tanks can vary in shape and orientation. Depending on the specific requirements of your project, you might choose:

- **Bounding Box Annotation:** Suitable for quickly annotating tanks that are roughly rectangular or when high precision is not critical.
- **Polygon Annotation:** Preferred when tanks have irregular shapes or when precise boundaries are necessary.

3. Waterbodies – Polygon Annotation

Annotation Type: Polygon Annotation

Purpose: Waterbodies such as lakes, rivers, and ponds often have complex, non-uniform shapes that are best captured using polygons.

Label Studio Implementation:

- **Tool Used:** PolygonLabels tag
- **Process:** Annotators click along the edges of the waterbody to create a polygon that accurately traces its boundaries.

Advantages:

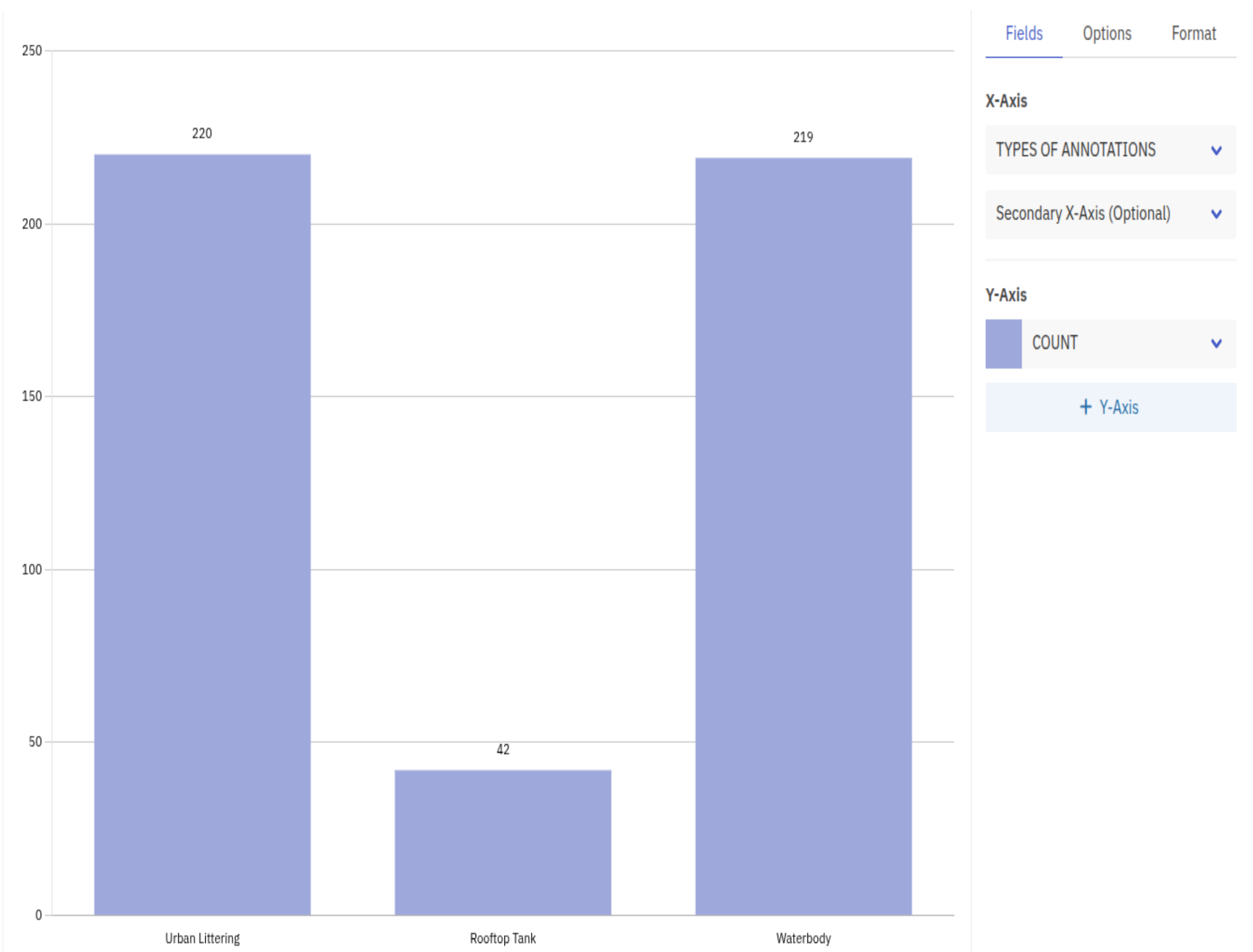
- Allows for precise delineation of waterbody boundaries, which is essential for tasks like land cover classification and environmental monitoring.

Summary Table:

Image Subtype	Annotation Type(s)	Label Studio Tool(s)	Use Case Highlights
Urban Littering	Polygon	PolygonLabels	Precise detection of irregularly shaped litter items
Rooftop Tank	Bounding Box, Polygon	RectangleLabels, PolygonLabels	Flexible annotation based on tank shape and precision needs
Waterbodies	Polygon	PolygonLabels	Accurate mapping of complex waterbody boundaries

Total number of images	244
Total number of images containing only urban littering	25
Total number of images containing only rooftop tanks	0
Total number of images containing only waterbody	8
Total number of images containing all (urban littering, rooftop tanks , waterbody)	170
Total number of images containing (urban littering , waterbody)	41
Total number of annotations (urban littering)	220

Total number of annotations (rooftop tank)	42
Total number of annotations (waterbody)	219



CODE TO COUNT TOTAL NUMBER OF ANNONATIONS IN THE COMPRESSED FILE .

```
import zipfile
import json
from collections import Counter

def count_annotations_in_zip(zip_path):
# Define the annotation categories of interest
target_categories = {'urban littering', 'waterbody', 'rooftop'}
annotation_counts = Counter()

# Open the ZIP file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
# Iterate through each file in the ZIP archive
for file_name in zip_ref.namelist():
# Process only JSON files
if file_name.endswith('.json'):
with zip_ref.open(file_name) as file:
try:
# Load the JSON content
data = json.load(file)
# Extract annotations; adjust the key as per your JSON structure
annotations = data.get('annotations', [])
for annotation in annotations:
# Extract the category label; adjust the key as per your JSON structure
category = annotation.get('label', '').lower()
if category in target_categories:
annotation_counts[category] += 1
except json.JSONDecodeError:
```

```python
print(f"Warning: Failed to parse {file_name}")

# Display the counts for each category
for category in target_categories:
print(f"{category.capitalize()} annotations: {annotation_counts.get(category, 0)}")

# Example usage
zip_file_path = 'path_to_your_annotations.zip' # Replace with your ZIP file path
count_annotations_in_zip(zip_file_path)
```

```
Urban Littering (polygon): 2
Rooftop Tank (polygon or rectangle): 2
Waterbodies (polygon): 1
```

SAMPLE OUTPUT OF FEW FILES OF ANNOTATION

I have followed this process of compressing and running in all folders of json and founded individually the results and then I summed it up , to compile my final results.

CODE TO OVERLAY ANNOTATIONS was also included(Refer Appendix)

OUTPUT AFTER OVERLAYING THE ANNONATION USING JPG AND JSON (WATERBODY ANNOTATION)

OUTPUT AFTER OVERLAYING THE ANNONATION USING JPG AND JSON (WATERBODY AND URBAN LITTERING ANNOTATION)

OUTPUT AFTER OVERLAYING THE ANNONATION USING JPG AND JSON (ROOFTOP TANK ANNOTATION)

CONCLUSION AND FUTURE SCOPE OF WORK

In this project, we utilized aerial imagery and computer vision techniques to annotate key urban features—specifically litter accumulation zones, rooftop water tanks, and waterbodies—using Label Studio. Polygon annotations were employed to map areas with high waste accumulation, aiding in the development of efficient waste disposal strategies. Bounding boxes and polygons were used to identify rooftop water tanks, providing insights into urban water storage infrastructure. Additionally, natural and man-made waterbodies were delineated to monitor their size and seasonal changes, contributing to effective water resource management and flood risk assessment.

Despite these accomplishments, certain limitations were encountered. The manual annotation process proved to be time-consuming and labor-intensive, highlighting the need for automation. Furthermore, the variability in image quality and urban layouts posed challenges in maintaining annotation consistency. The current approach also lacks the capability to perform real-time detection, which is essential for dynamic urban planning scenarios. The process of manually annotating complex urban features demands significant human resources and is prone to inconsistencies, especially when dealing with large volumes of data. This underscores the need for developing automated or semi-automated annotation tools that can assist in efficiently processing and accurately labeling aerial imagery .

Looking ahead, integrating deep learning models trained on the annotated datasets can significantly enhance the efficiency and accuracy of urban feature detection. Implementing semantic segmentation and object detection algorithms will facilitate real-time analysis, enabling proactive urban planning and smarter city services. In addition to the previously mentioned limitations, several other challenges were encountered during the annotation of aerial imagery for urban feature detection. One significant issue is the complexity of accurately delineating small objects, such as rooftop water tanks or scattered litter, which often occupy minimal pixel areas in high-resolution images. This small object detection problem is exacerbated by the limitations of current object detection algorithms, which may struggle to identify and accurately annotate these features due to their size and the potential for them to be obscured by shadows or other urban elements .

Addressing these challenges is crucial for improving the accuracy and efficiency of urban feature detection in aerial imagery.

APPENDIX -

CODE TO OVERLAY THE ANNOTATION

```
import json
from google.colab.patches import cv2_imshow
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen     Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving DJI_0288.JPG to DJI_0288 (1).JPG
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen     Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving DJI_0288.json to DJI_0288 (1).json
```

```
import cv2
import numpy as np
with open('DJI_0288.json') as f:
data =json.load(f)
image_name = data[0]['data']['image'] # Assuming 'image' is nested under a 'data' key within the dictionary.
annotations = data[0]['annotations']
img = cv2.imread('/content/DJI_0288 (1).JPG')
for ann in annotations:
for res in ann['result']:
points = res['value']['points']
# Convert relative points (0-100) to absolute pixel values
pts = np.array([
[int(p[0] * img.shape[1] / 100), int(p[1] * img.shape[0] / 100)]
for p in points
], np.int32)
pts = pts.reshape((-1, 1, 2))
```

```
cv2.polylines(img, [pts], isClosed=True, color=(0, 255, 0), thickness=2)
from google.colab.patches import cv2_imshow
cv2_imshow(img)
```

My github link - https://github.com/vaibhavm291/IDEAS-ISI-KOL-INTERNSHIP-PROJECT