Deggendorf Waste Sorting Assistant

Computer Vision Project

The ***Problem***

International students in Deggendorf struggle with the German waste sorting system. Different colored bins have specific rules that are often explained *only in German*, making it difficult to know what goes where. This leads to confusion, improper sorting, and potential fines when bins are contaminated with the wrong materials.

Our Solution

Smart Bin Recognition

Project Goals

Develop an MVP image classification model *capable of identifying waste bins* in Deggendorf. That provide users with clear guidance on proper waste disposal based on bin classification.

It took us *3 major* steps

Step 1

The Dataset

Step 2

The Labeling

Step 3

The Codebase

How we made

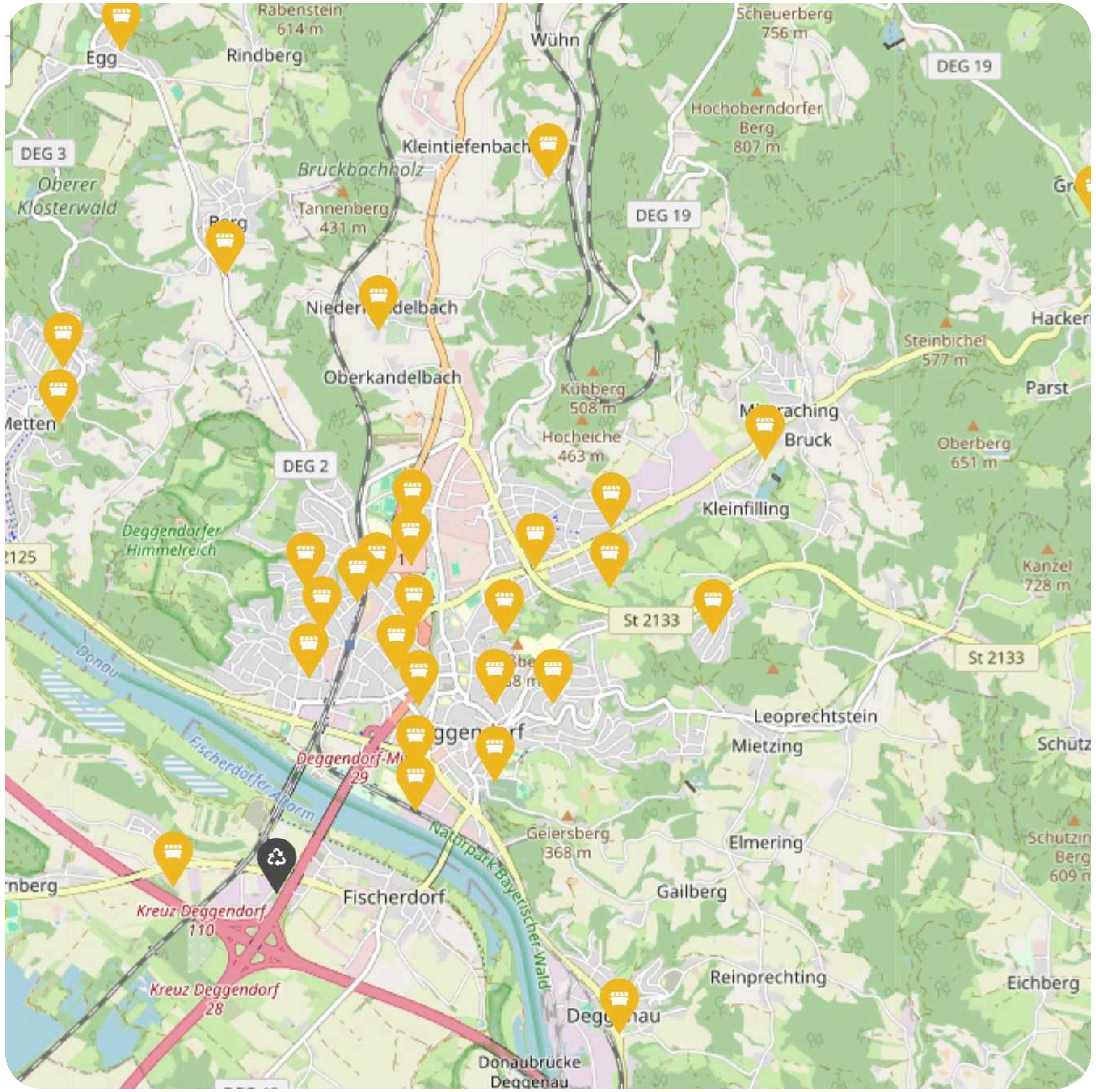
The Dataset

Building Our Own Dataset

We captured *466 real-life photos* of waste bins in Deggendorf to ensure high relevance and local accuracy.

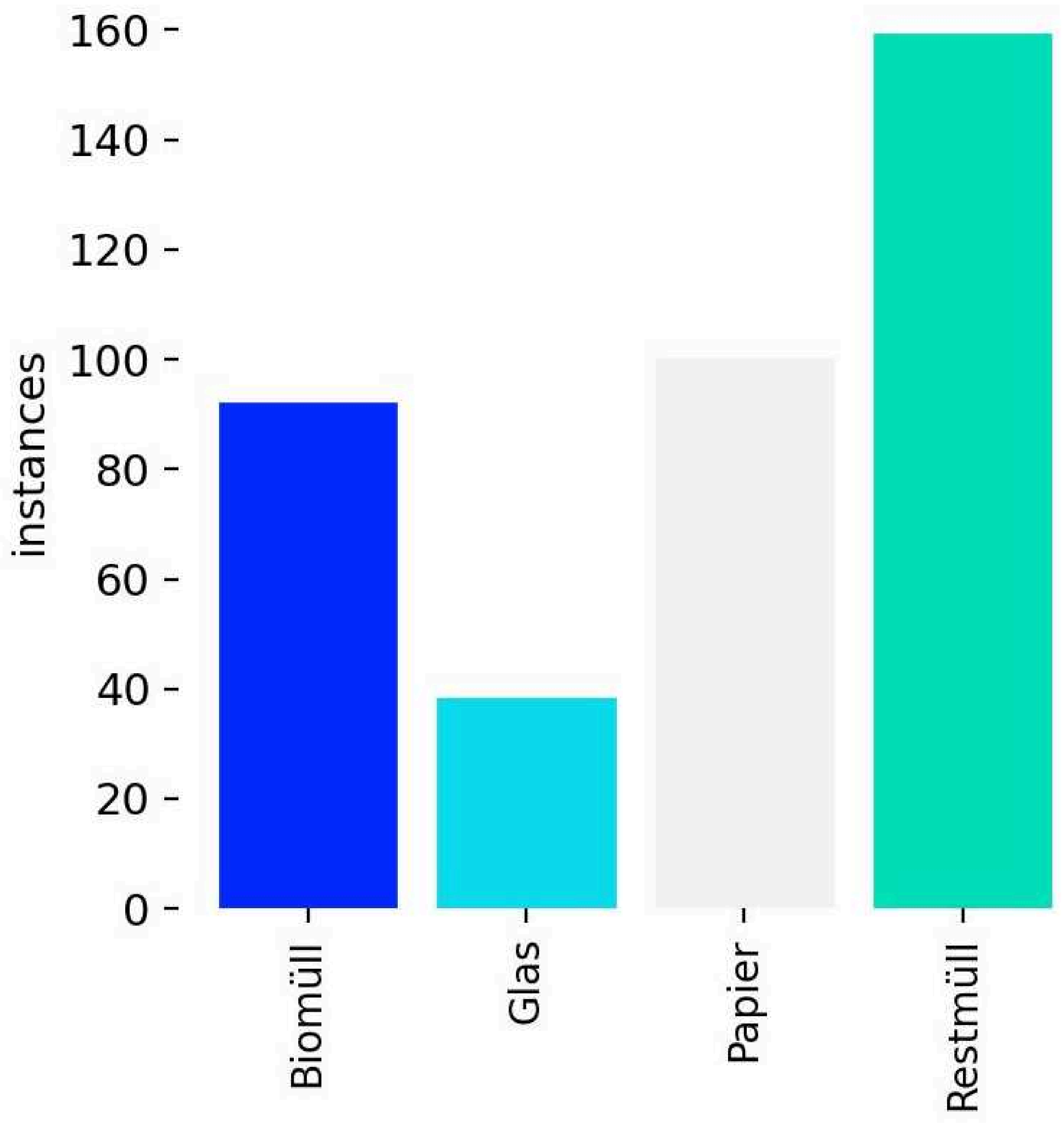
Storage Solution

We used Google Drive as a *collaborative storage solution* for our dataset, with direct integration and sync with the Jupyter Notebook.



The Dataset

The Categories



We made *4 categories* for the bins found in Deggendorf: Biomüll, Glas, Papier and Restmüll. The biggest problem was the Glas bins.

How we proceeded with

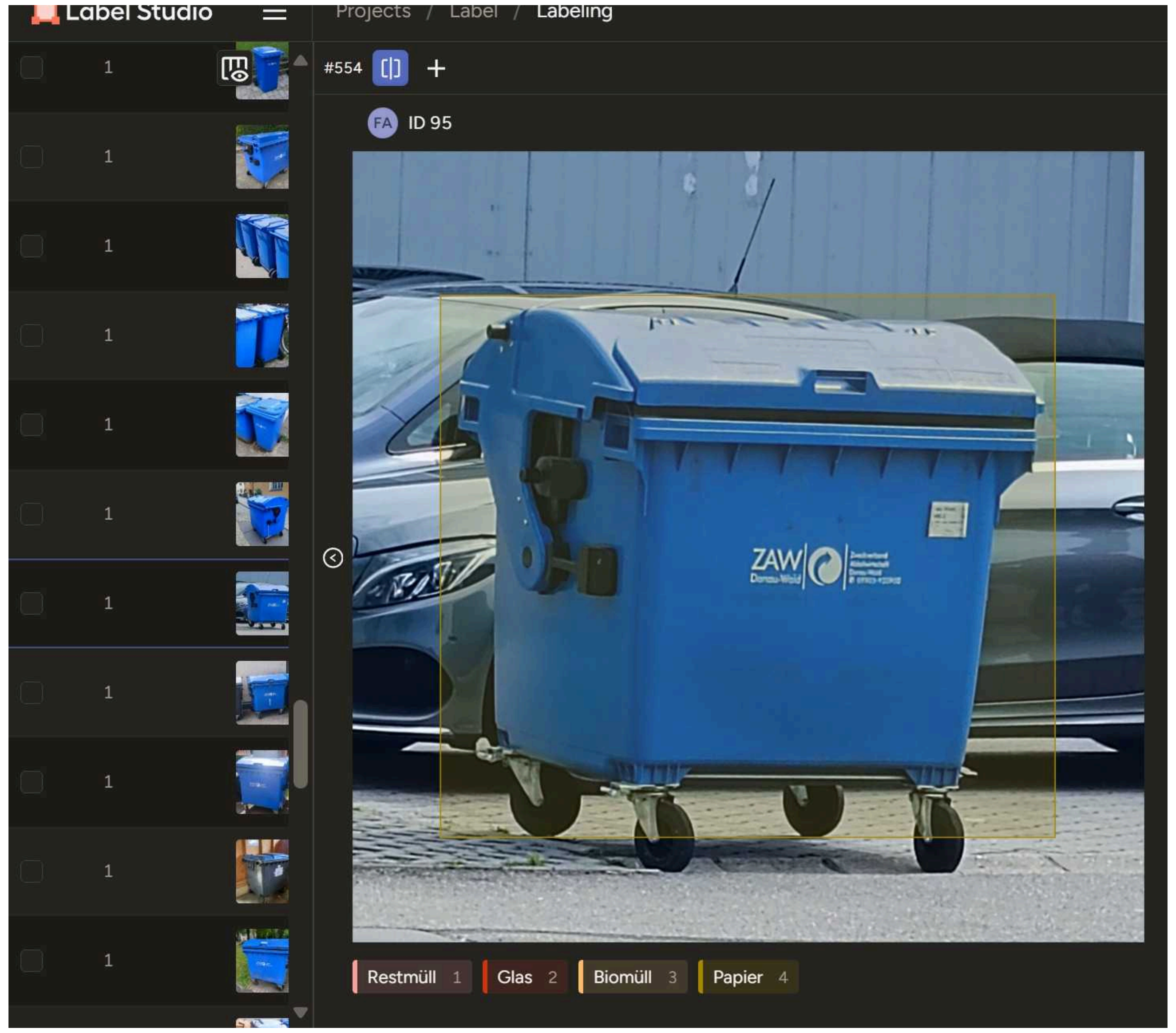
The Labeling

Custom Labeling in Google Colab

We initially created our own labeling GUI Widgets inside of Jupyter Notebook, but it was lacking in speed and efficiency

Manual Labeling with Label Studio

Each bin was manually labeled using bounding boxes. Label Studio helped us export labels directly in YOLO format.

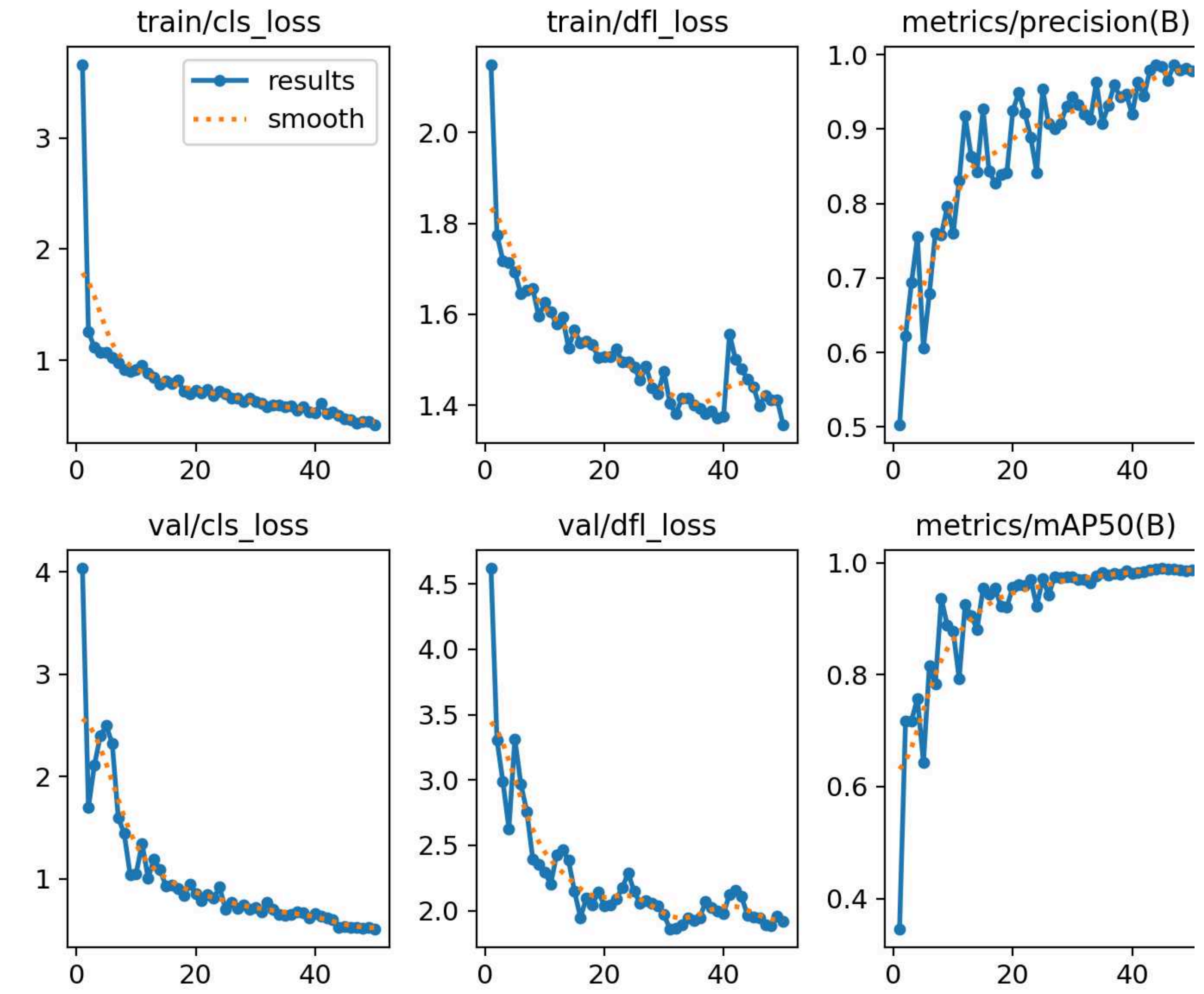


Fixing Filename Encoding Issues

Umlaut characters in image names caused problems — we solved this by renaming files for smooth YOLO integration.

How we developed the

The Codebase



Fine-Tuning YOLOv8

We fine-tuned a pretrained YOLOv8 model on our labeled dataset to teach it how to detect different bin types.

Real-Time Detection with OpenCV

OpenCV allowed us to build a live interface where bin types are detected instantly using the trained model.

Testing & Adjustments

We evaluated the model’s predictions and iterated as needed to improve its real-time accuracy and reliability.

The Codebase

The Training

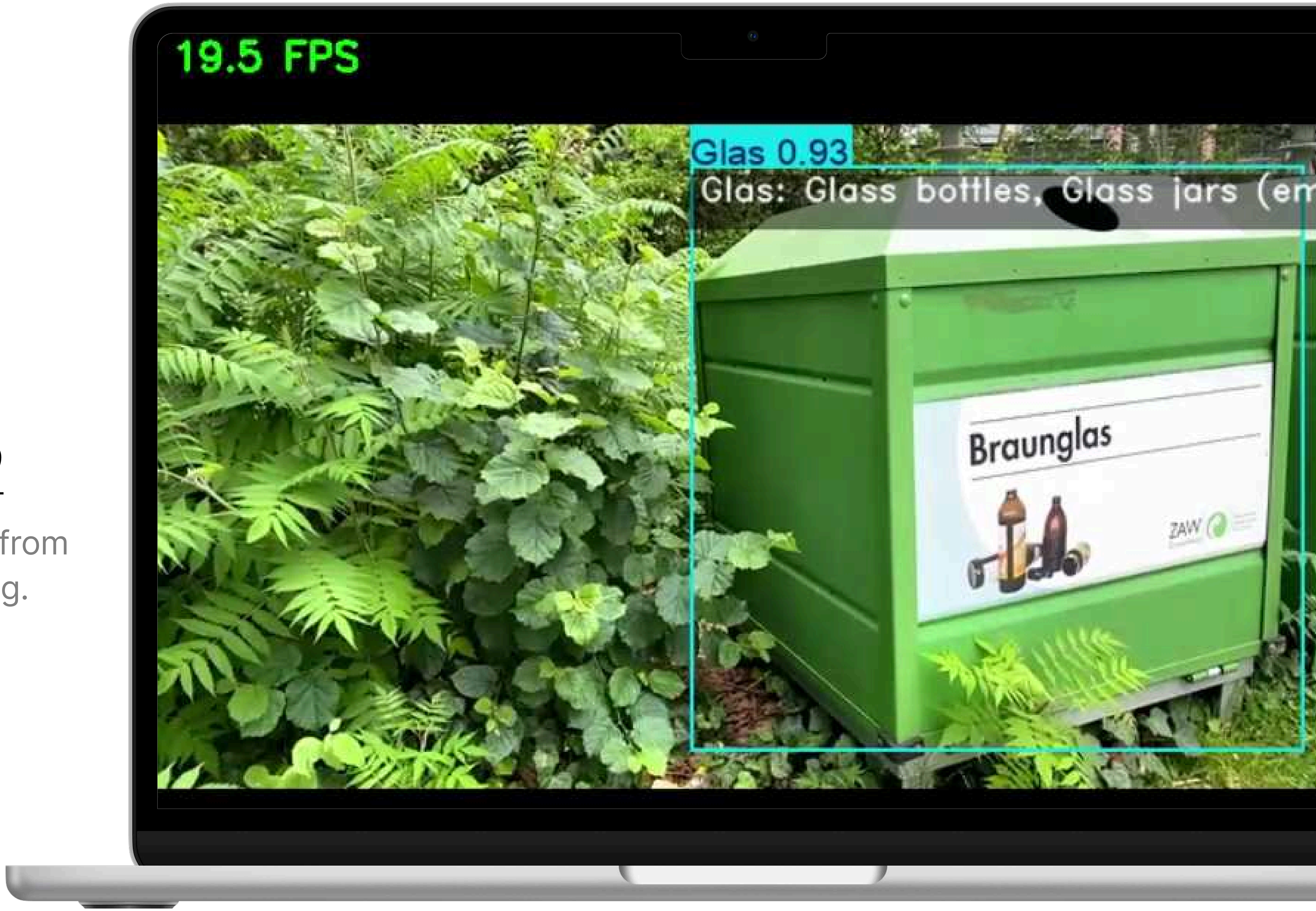
- Trained on: GPU
- Batch size: 4
- Epochs: 50
- Model: **yolov8s.pt**

- Val: **20%** *(94 images)*
- Train: **80%** *(372 images)*

```
Ultralytics 8.3.160 🚀 Python-3.11.13 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=randaugment, batch=4, b
Overriding model.yaml nc=80 with nc=4


 22        [15, 18, 21]  1   2117596  ultralytics.nn.modules.head.Detect           [4, [128, 256,
Model summary: 129 layers, 11,137,148 parameters, 11,137,132 gradients, 28.7 GFLOPs

Transferred 349/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
train: Fast image access ✅ (ping: 0.3±0.1 ms, read: 430.2±232.8 MB/s, size: 2169.0 KB)
train: Scanning /content/drive/MyDrive/cv_garbage/YOLO_Dataset/labels/train.cache... 372 images,

val: Fast image access ✅ (ping: 0.6±0.3 ms, read: 171.0±76.1 MB/s, size: 2609.7 KB)
val: Scanning /content/drive/MyDrive/cv_garbage/YOLO_Dataset/labels/val.cache... 94 images, 0 bac
Plotting labels to /content/drive/MyDrive/cv_garbage/models/waste_detector_20250628_0743162/label
optimizer: AdamW(lr=0.001, momentum=0.937) with parameter groups 57 weight(decay=0.0), 64 weight(
Image sizes 960 train, 960 val
Using 2 dataloader workers
Logging results to /content/drive/MyDrive/cv_garbage/models/waste_detector_20250628_0743162
Starting training for 50 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/50      2.28G      1.358      2.166      2.047         12        960: 100%|██████████| 9
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|█


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/50      2.73G      1.241      1.398      1.916         13        960: 100%|██████████| 9
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|█


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/50      2.76G      1.126      1.326      1.815         13        960: 100%|██████████| 9
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|█


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/50      2.76G      1.145      1.249      1.816         14        960: 100%|██████████| 9
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|█


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/50       2.8G      1.046      1.115      1.723         10        960: 100%|██████████| 9
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|█
```

Here is a *Demo Video*

This is a screen recording of a *live session* from a phone connected to a laptop by streaming.



Lessons learned

Fine-tuning pretrained models saves time

Using YOLOv8 gave us a solid head start and allowed for efficient training.

User context matters

Creating a locally relevant dataset ensures better adoption and usefulness.

Manual labeling is time-consuming but essential

Bounding box labeling helped the model focus and perform more accurately.

Pretrained models aren't perfect

Even with YOLOv8, custom tuning was necessary to adapt to our specific bin types and conditions.

Next Steps

Multi-language Support

to clear up sorting for every resident.

Mobile or Web Integration

so anyone can use it without the laptop workaround.

Partner with Deggendorf city / THD

to integrate services like pickup schedules.

Thank you for your attention

Any questions?