YOLOv8 Bottle Detection Report

Project Overview

This project involved training a YOLOv8 object detection model to identify **bottles** in images using two Kaggle datasets. The main objective of this project was to merge two annotated datasets, train a YOLOv8 model, evaluate its performance, and visualize predictions.

Colab Link:
https://colab.research.google.com/drive/1fr23T-wOAYvr7qQtuAAV9BSxflNpmYJf?usp=sharing

Video link:
https://drive.google.com/file/d/1zmGwY8O77bTkXzN5Z_q3SdDtejvFGosR/view?usp=sharing

Dataset Summary

I used two Kaggle datasets:

1. **Plastic Bottles Image Dataset** by *Siddharth Kumar Sah*
   Folder & Contents: It had three folders: train, test and valid. Each of the folders included images and labels according to those images.
   Summary: 3999 images, 3999 labels, 1 yaml file and 1 cache file.
   Link: https://www.kaggle.com/datasets/siddharthkumarsah/plastic-bottles-image-dataset

2. **Bottle Dataset** by *Samuel Ayman*
   Folder & Contents: It had a single folder: 'bottle' and inside it there were another folder named 'labels' and rest of the files were images
   Summary: 1000 images(.jpg) and 1000 labels(.txt)
   Link: https://www.kaggle.com/datasets/samuelayman/bottle?select=bottle

Both datasets contained YOLO-formatted annotations.

Sorting:

At first, I downloaded two dataset from kaggle and stored it using two folders 'plastic' and 'bottle' inside the 'data' folder. From there, I copied only the images that had corresponding label files into a new folder called 'final_dataset', renaming them to avoid duplication. But I didn't take any images from the 'test' folder of 'plastic'. All images and labels were placed in organized 'train' folders. Then, we split the data into training and validation sets (80/20), resulting in a clean, merged dataset ready for training the YOLOv8 model.

Directory Structure

After organizing and merging datasets, the directory structure became:

```
final_dataset/
├── images/
│   ├── train/      # 3480 bottle images
│   └── val/        # 871 bottle images
└── labels/
    ├── train/      # YOLO .txt label files
    └── val/        # YOLO .txt label files
```

Code & Logic

1. Dependencies & Kaggle Setup

Set up Kaggle credentials, downloaded the datasets, and unzipped into data/.

```
!pip install ultralytics
from ultralytics import YOLO
import os
import shutil

from sklearn.model_selection import train_test_split
from google.colab import files
files.upload()
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

!kaggle datasets download -d siddharthkumarsah/plastic-bottles-image-dataset
!kaggle datasets download -d samuelayman/bottle
```

2. Merged the datasets Images and labels from each dataset were renamed and merged into final_dataset/images/train and labels/train.

```
os.makedirs('final_dataset/images/train', exist_ok=True)
os.makedirs('final_dataset/labels/train', exist_ok=True)


# Source paths
img_src = 'data/bottle/bottle'
lbl_src = os.path.join(img_src, 'labels')
# Move all image-label pairs to final_dataset
for fname in os.listdir(img_src):
    if fname.endswith(('.jpg', '.png', '.jpeg')):
        base = os.path.splitext(fname)[0]
        label_path = os.path.join(lbl_src, base + '.txt')
        # Check if label exists
        if os.path.exists(label_path):
            new_img_name = f'bottle_{fname}'
            new_lbl_name = f'bottle_{base}.txt'
            shutil.copy(os.path.join(img_src, fname), os.path.join('final_dataset/images/train', new_img_name))
            shutil.copy(label_path, os.path.join('final_dataset/labels/train', new_lbl_name))
plastic_base_dir = 'data/plastic/Plastic Bottle Image Dataset'
subfolders = ['train', 'valid']  # skip 'test'
for sub in subfolders:
    plastic_img_dir = os.path.join(plastic_base_dir, sub, 'images')
    plastic_lbl_dir = os.path.join(plastic_base_dir, sub, 'labels')
    if not os.path.exists(plastic_img_dir) or not os.path.exists(plastic_lbl_dir):
        print(f"Skipping missing folder: {sub}")
        continue
    for fname in os.listdir(plastic_img_dir):
        if fname.endswith(('.jpg', '.jpeg', '.png')):
            base = os.path.splitext(fname)[0]
            label_path = os.path.join(plastic_lbl_dir, base + '.txt')
            if os.path.exists(label_path):
                new_img_name = f'plastic_{sub}_{fname}'
                new_lbl_name = f'plastic_{sub}_{base}.txt'
                shutil.copy(os.path.join(plastic_img_dir, fname), f'final_dataset/images/train/{new_img_name}')
        shutil.copy(label_path, f'final_dataset/labels/train/{new_lbl_name}')
```

3. Split train/val sets Used train_test_split (80/20 split) to create a validation set.

```
image_files = [f for f in os.listdir(img_dir) if f.endswith(('.jpg', '.jpeg', '.png'))]

train_files, val_files = train_test_split(image_files, test_size=0.2, random_state=42)

for f in val_files:

    base = os.path.splitext(f)[0]

    label = f"{base}.txt"

    shutil.move(os.path.join(img_dir, f), os.path.join(val_img_dir, f))

    shutil.move(os.path.join(lbl_dir, label), os.path.join(val_lbl_dir, label))

model = YOLO('yolov8n.pt')  # or try 'yolov8s.pt' for better accuracy
```

4. Configured YOLOv8

```
bottle_data.yaml:
train: final_dataset/images/train
val: final_dataset/images/val
names:
  0: bottle
```

5. Training Command

```
model = YOLO('yolov8n.pt')

model.train(

    data='bottle_data.yaml',     # Path to your dataset config

    epochs=10,                   # Number of training epochs

    name='bottle_detector',      # Experiment name

)
```

6. Evaluation & Prediction

```
results = model.val()

model = YOLO('runs/detect/bottle_detector/weights/best.pt')  # adjust path as needed
```

```
# Get full paths of the first 500 images in val folder

val_dir = 'final_dataset/images/val'

val_images = sorted([

    os.path.join(val_dir, f)

    for f in os.listdir(val_dir)

    if f.endswith(('.jpg', '.jpeg', '.png'))

])[:500]

# Run prediction

results = model.predict(source=val_images, save=True, imgsz=640)
```

Model Performance Summary

Training stats (**10 epochs**):

- Training images: 3480
- Validation images: 871 (took 500 from them because the runtime my session was always getting terminated when I am trying to validate)

Validation Output:

Precision:	0.691
Recall:	0.545
mAP@0.5:	0.506
mAP@0.5:0.95:	0.348

Interpretation:

- **Precision (69.1%)**: good: most predictions are correct
- **Recall (54.5%)**: moderate: some bottles were missed
- **mAP@0.5 (50.6%)**: main object detection accuracy metric

Metrics Visualization:

The training graphs show steady improvement across all metrics over 10 epochs:

- Box Loss, Class Loss, and DFL Loss (for both training and validation) consistently decreased, indicating the model learned effectively without overfitting.

- Precision improved to ~70%, meaning most predicted boxes were correct.

- Recall increased to ~55%, showing the model is catching more real bottles.

- mAP@0.5 rose above 50%, and mAP@0.5:0.95 reached ~35%, showing strong detection performance, even under stricter conditions

Sample Predictions

- Predictions were mostly correct but sometimes missed certain bottles or produced low-confidence boxes.
- Some examples showed predictions with confidence scores around 0.58.

Tools Used

Python (Google Colab)	3.11.13
Ultralytics YOLOv8 8.0.178	8.3.163
Kaggle API (via kaggle package)	
NumPy	2.0.2
Matplotlib	3.10.0
OpenCV	4.12.0
scikit-learn 1.2.2	1.6.1
OS Platform	Linux 6.1.123+ (posix)
Torch (PyTorch)	2.6.0+cu124