ASSIGNMENT 3: Small Object Detection Report

Author: Shambhavi Danayak
Professor: Bo Shen
Student ID: 012654513
Submission date: 03/14/2025

NOTE: Here I am discussing my work on a high level, a more detailed documentation on code and work done is embedded in the jupyter notebook (A3.ipynb)

Dataset description and preprocessing steps

Dataset URL: Roboflow.com
The dataset is fully annotated and provides various downloading formats (YOLOv11, YOLOv9, YOLOv5, YOLO Darknet, PaliGemma etc.). There are a total of 2410 images divided into Train Set (1687 images), Valid Set (482 Images) and Train Set (241 Images). No annotation was required as already annotated.
To verify the images are annotated in YOLO format
<class_id> <x_center> <y_center> <width> <height> ,

Verifying is annotated to YOLO format <class_id> <x_center> <y_center> <width> <height>

[7]:
```
import os
dataset_path = "roboflow_data"
train_labels_path = os.path.join(dataset_path, "train", "labels")
label_files = os.listdir(train_labels_path)
print("Sample label files:", label_files[:5])
```

```
Sample label files: ['001027_jpg.rf.12581398841eaa4430db668bf5656def.txt', '000804_jpg.rf.24eea6470cedf8a08386e1b04eeafde7.txt', '001812_JPG.rf.87e9b64fc178950f95e82cc8039de549.txt', '001525_jpg.rf.0087d13d6e9347d73d232cd95c26790a.txt', '001145_jpg.rf.0997e940445a76588802d520a93fb670.txt']
```

[8]:
```
for label_file in label_files[:3]:
    label_path = os.path.join(train_labels_path, label_file)

    with open(label_path, "r") as f:
        lines = f.readlines()

    print(f"\nContents of {label_file}:")
    for line in lines:
        print(line.strip())
```

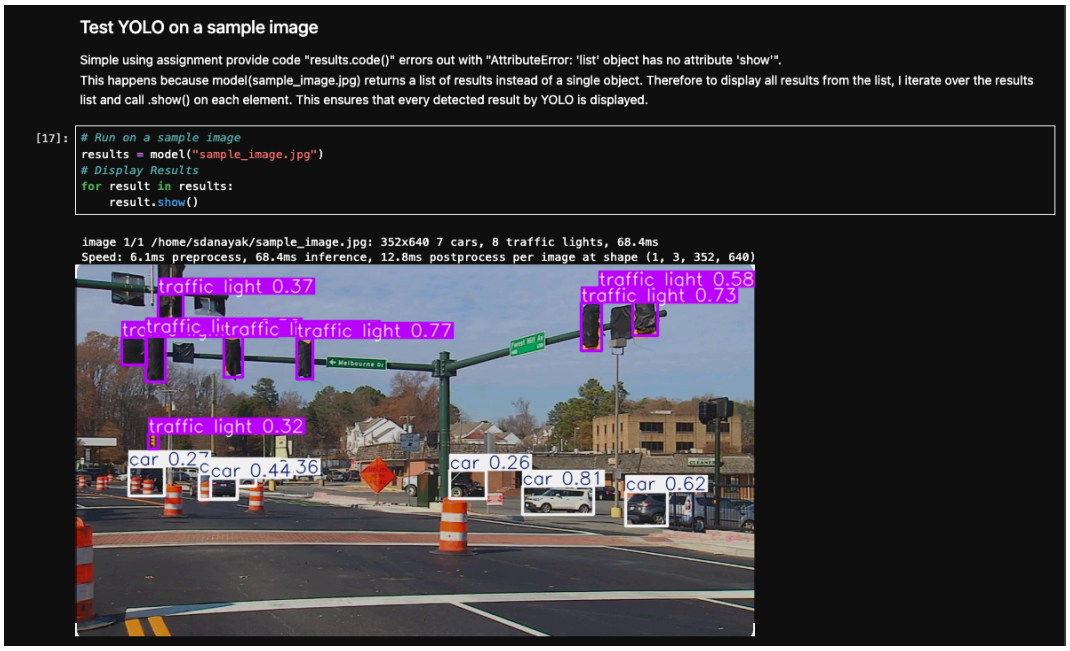
```
Contents of 001027_jpg.rf.12581398841eaa4430db668bf5656def.txt:
10 0.7415865384615384 0.46274038461538464 0.359375 0.4495192307692308

Contents of 000804_jpg.rf.24eea6470cedf8a08386e1b04eeafde7.txt:
13 0.5072115384615384 0.75 0.1814903846153846 0.23197115384615385

Contents of 001812_JPG.rf.87e9b64fc178950f95e82cc8039de549.txt:
7 0.46634615384615385 0.16947115384615385 0.0625 0.09254807692307693
```

jupyter 000017_jpg.rf.8bbe578f3950a0de00f0f4a5d01fd110.txt Last Checkpoint: 2 days ago

File Edit View Settings Help

```
1 13 0.28365384615384615 0.20552884615384615 0.2536057692307692 0.17307692307692307
```

Finally a sample image was used to test YOLOv8 pre-trained model,



Challenges faced in detecting small objects

There were some challenges faced while training the YOLO Model for small object Detection,

1. running on a personal machine was taking more than 5 hours to run 20-30 epochs and the notebook kept on crashing. To avoid this issue university A100 server was used then port forwarded to localhost in order to run on the browser. Running on A100 significantly lowered the training time.
2. **Low Resolution issues:** small objects sometimes appear blurry or pixelated in the image, making it hard for the model to extract meaningful features.
3. **Anchor Box Tuning Issues:** Since we used YOLOv8 it does not support manual tuning of anchor sizes, I used other augmentation features to cover up for that, ‘translate’, ‘mosaic’ and ‘perspective’.
4. **Memory & performance issues:** Using Large video files led to memory overload and the jupyter notebook crashed. Using stream=True and processing smaller video helped but required careful resource management as even the server had limitation on disk quota.
5. **Issue with blurring:** YOLOv8 does not support argument “blur_ratio” so instead i used arguments ‘hsv_h’ (introduces color variability), ‘hsv_s’ (alters image saturation) and ‘hsv_v’ (modifies brightness).

Experimentation results with different settings (e.g., resolution, NMS, confidence threshold)

NOTE: The experiments results with image resolution, data augmentation and anchor box tuning were saved in different folders for easier comparison. (runs_high_resolution, runs_Data_aug_scale_withAnchorBoxTuning and runs_Data_aug_scale)

1. **Higher Image resolution (imgsz=1280 vs. 640):** Image resolution 640 (mAP at IoU=0.5 is 0.882) and finished in lower computation time (0.259hrs = ~15 mins). Increasing the image resolution to 1280 did not improve the detection of small objects (mAP at IoU=0.5 is 0.867) and increased computation time (0.761 hrs = ~45 mins). Further hyperparameter tuning could help with better results.
2. **Data Augmentation (scaling, tilting etc):** Data augmentation did not improve the model performance rather there is a drop in mAP value (0.845 at IoU=0.5). Data augmentation techniques may have introduced some noise. For example, scaling and degree may have altered object shapes, making bounding box placement less accurate.
3. **Anchor Box Tuning:** YOLOv8 does not support manual tuning of the anchor box so I used some data augmentation arguments, 'translate', 'mosaic' and 'perspective.' This slightly improved the mAP value (0.860 at IoU=0.5) indicating better detection.
4. **Inference on sample image and video by tuning Non-Maximum Suppression(NMS) and confidence threshold:** Increasing NMS threshold determines how much overlap is allowed between bounding boxes before one is suppressed. Confidence Score is the minimum level of certainty a model needs to have in its prediction. Higher confidence score (0.5) helped filter out low-confidence detections, thereby improving accuracy. There needs to be a balance between confidence scores as a very high confidence score could reduce false positives but will miss on smaller objects whereas lower confidence score could lead to inaccuracy and higher computation time. Overall we can say fine-tuning NMS and confidence threshold is crucial for achieving an optimal balance between precision and Recall.

Suggestions for improving small object detection

Next steps that can be taken to improve the small object detection,

1. **Improve image quality in the dataset:** provides model with high image resolution (1280x1280) as it provides more pixel details, making small objects easier to detect.
2. **Improving Training Data:** including more images with small objects from different perspectives and environments. This will make the dataset more diverse thereby helping the model to generalize better.
3. **Better NMS confidence threshold tuning:** lower NMS threshold ensures small objects aren't suppressed too early due to overlap with large objects. A balanced confidence score to ensure small objects are not missed.
4. **Analyze False Positives & Negatives, confusion matrices, mAP scores etc.:** Identifying misclassified objects and adjusting model parameters accordingly could also further better the model.

THANK YOU!