Deep Learning for Computer Vision

Autonomous, Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images

Project by: Snehil Nair

Roll No: 21f2000602

Email: 21f2000602@ds.study.iitm.ac.in

Introduction

In India, public littering and discarded trash are still a serious ecological, aesthetic, and social problem despite the effort put in by governments and various NGOs to make people aware of the problems. To tackle this issue, the project proposes to use a machine learning based solution that aims to use UAV's (Unmanned Ariel Vehicle's) to collect low altitude images and detect litter objects that can then be put on a global map. This project deals engineering a solution to do instance segmentation of trash and litter objects present in the images taken by the UAV. The data for this project contained images and mask information. A json file containing instance segmentation data was provided along with train and test csv files containing url's to images. The metric to be used for evaluation is IoU scores (Intersection over Union scores).

Approaches

After Preprocessing, the data contained 60 classes of distinct litter/trash objects. Due to disparities between url's provided in train csv and annotations provided, I generated test samples based on annotations json which led to a total dataset having *1438* examples where multiple instances of multiple objects present in many samples.

There are a few models to do instance segmentation and through some research, I decided to test the following models for this project as listed below:

1. Faster RCNN: Provides better performance metric but has slow inference speed

2. YOLO (You Only Look Once): Performance is compromised for faster inference speed. Most model applications boast real time detection capabilities.

3. SAM (Segment Anything Model): Decent performance and relatively decent inference speed

4. Panoptic Segmentation

Based on project requirements, I decided to test YOLO and SAM due to faster inference speeds. Although both these models have open pretrained implementation available, they both suffer from some restrictions. I used Ultralytics implementation of these models to start with a pretrained model and fine tune it according to the project as the dataset for training was quite limited. A few classes of each models are available, starting with nano, to small and larger models each corresponding to increasing number of parameters.

SAM model implementations (particularly as provided by Ultralytics) does not support training. The implementation that I did find for training used 3rd party api to do so, atleast for instance segmentation process.

YOLO implementation provides many models and allows training on custom dataset with good inference speeds. I was able to retrain a nano YOLO model for the custom task of Trash/litter object instance segmentation using google colab notebooks and Ultralytics package and pretrained model.

I was unable to find a satisfactory implementation or pretrained implementation of Panoptic Segmentation model. Training a model from scratch using 1400 samples for 60 different class seemed infeasible.

Limitations

I worked with YOLOv11 for this project, using the nano version of the model having nearly 2.6 million parameters and 56-60 ms of inference speed. During the process, there are some problems and limitations that stand out:

1. *Compute Requirements*: Training even a nano model on cpu is not feasible. During the project, I tried to train the model on both gpu and cpu, with the nano model taking nearly 30 minutes per epoch on cpu and 2 minutes per epoch on gpu. Since the training was done using T4 gpu's provided by google colab for 3-4 hours in a day, the training time was limited.
2. *Inference vs Performance tradeoff*: The model has good inference speed, providing a very good backbone for near real time applications, but there is a significant tradeoff of performance to achieve these speeds.

Training any model would require better compute, as models like SAM also start with 2 million parameters and work their way up to really large models that perform even better.

Quantitative Results

I used the YOLOv11 nano segmentation model with nearly 2.6 million parameters by Ultralytics with pretrained weights trained from an instance segmentation task. I then re-trained this model on 1436 samples and validated on 144 samples. The following are the qualitative results as measured on the validation set:

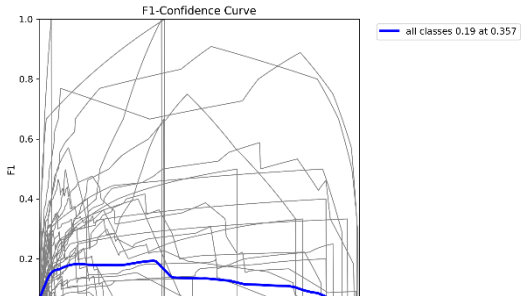


Figure 1 F1 Score vs Confidence

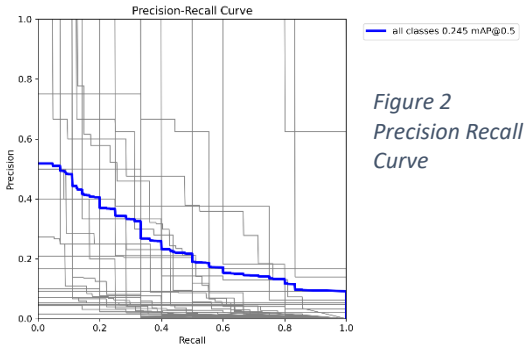


Figure 2 Precision Recall Curve

1. *Inference Speed*: The models completes inference on validation set in under a minute. It takes the model a total of nearly 356 ms to complete inference on a single image including pre and post processing.
2. *IoU Scores*: The intersection over union score for bounding box is of the order of 0.1 since model was not provided with bounding box information during training.
3. *Mean Average Precision*: The model has a high variance of mean average precision scores across classes. The mAP scores at 50% threshold for IoU gives a precision of as high as 0.92 for classes with more than one instance afor both boxes and masks and as low as 0.
4. *F1 score*: The model struggles with a relatively low F1 score of with the best being 0.19 at a confidence level of 0.357.

Table 1 Average Precision, Recall, mAP50 and mAP50-95 values

Box Precision	Box Recall	Box mAP50	Box mAP50-95	Mask Precision	Mask Recall	Mask mAP50	Mask mAP50-95
0.62	0.18	0.259	0.233	0.62	0.18	0.245	0.194

Qualitative Results

The model seems to struggle most with occluded objects. Glass based litter seems to be the least detected type of objects and are often misclassified. As evident from the below sample of images, the model also seems to perform badly on classification of detected objects.

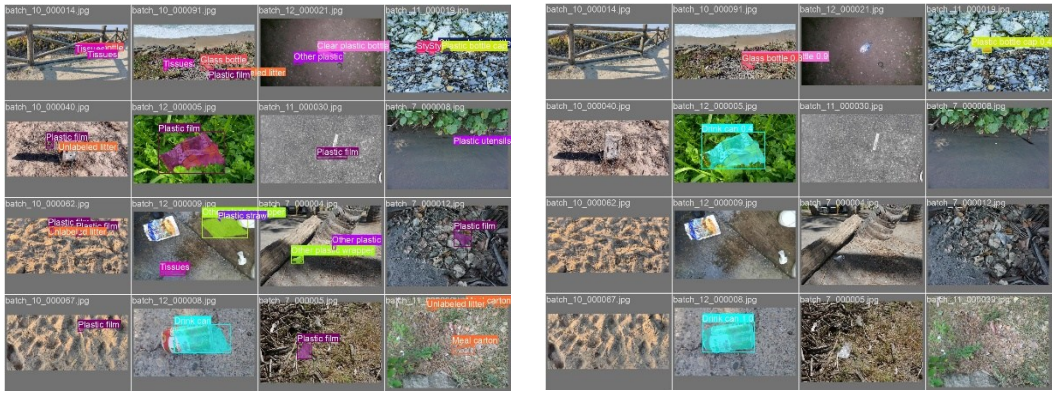


Figure 2 Validation Batch 1 a) (Left) Ground Truth vs b) (Right) Predictions

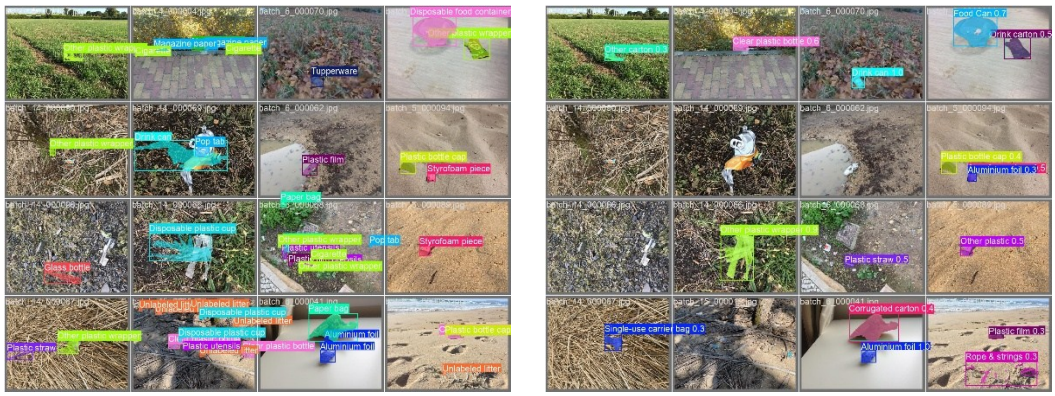
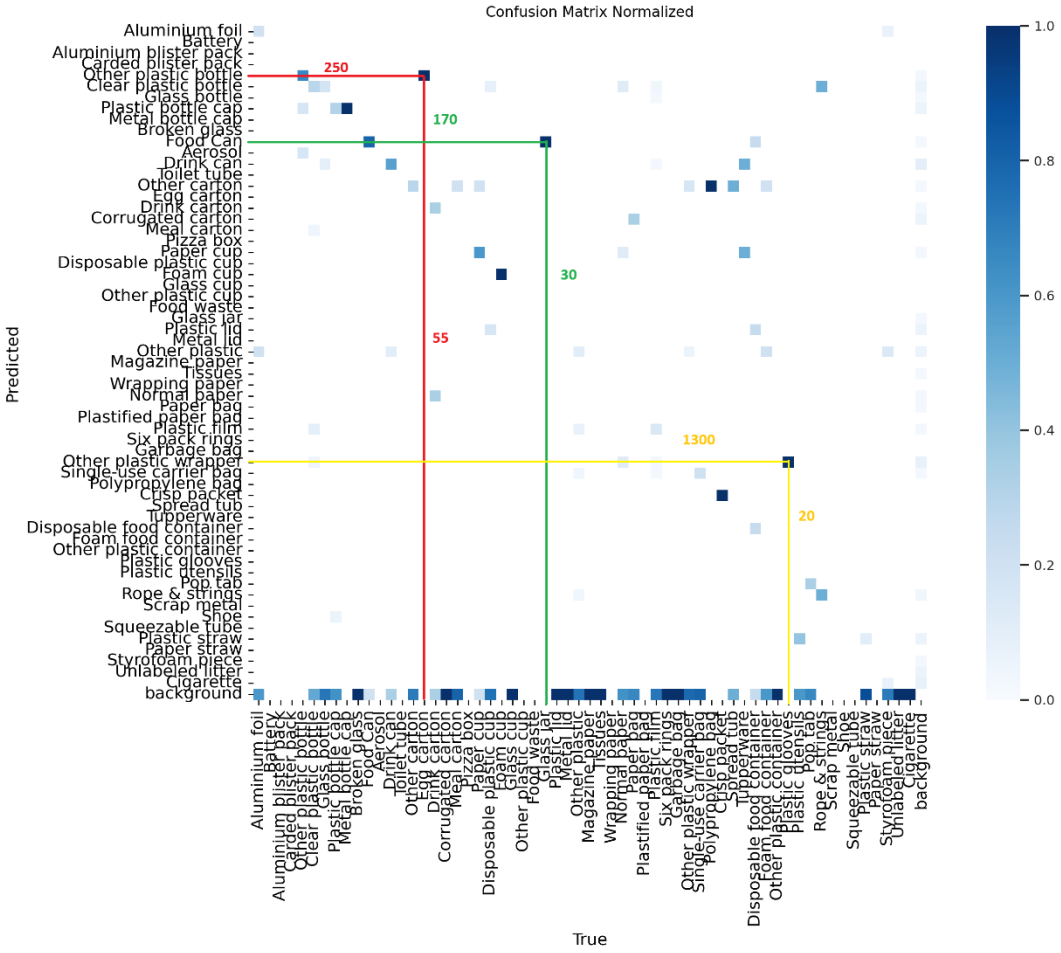


Figure 3 Validation Batch 2 a) (Left) Ground Truth b) (Right) Predictions

Observations

The Nano model of YOLO does not perform well on the given dataset. The model fails to detect a lot of objects in the scene and misclassifies objects when detected. These might be, however, issues arising due to training data. The dataset itself is skewed and imbalanced, making training a model with better metrics difficult. The classes with high amount of instances in the given dataset, for example, are `Cigarettes` and `Unlabelled Litter` with more than 3000 and 2000 samples each, which do not have high confusion rates.



The Confusion could be due to the disparity between instances of classes in the dataset as can be seen in the annotated confusion matrix above.

The bounding box and mask error could be addressed by using a larger dataset and a larger model. Ultralytics does catalogue a variety of bigger models that perform better on instance segmentation with larger model size of upto 62 million parameters.

The IoU score for the model is low as expected since during training, the model was not provided original bounding boxes and hence does not take bounding box losses into account. This was due to how the model was implemented in the backend. IoU score, hence, may not be a valid or accurate measure of model performance here.