Mid Evaluation Report

AI/ML-012

- **Team Details**:

Team leader:

1.) Siddhi Borase
Roll no.: 240002072
GitHub: https://github.com/ssb120107
LinkedIn: https://www.linkedin.com/in/siddhi-borase-0b0353351/

Other team members:

2.) Aarushi Pandey
Roll no.: 240003002
GitHub: https://github.com/aarushipd12
LinkedIn: https://www.linkedin.com/in/aarushi-pandey-2a8678320

3.) Shriya Deo
Roll no.: 240041013
GitHub: https://github.com/Shriya-1807
LinkedIn: https://www.linkedin.com/in/shriya-deo-396581334

4.) Tanishka Sihara
Roll no.: 240003078
GitHub: https://github.com/tanishka-sihara
LinkedIn: http://www.linkedin.com/in/tanishka-sihara

- **Team**: AI/ML-012
- **PS Name**: Drone Footage Object Detection

- **Project Workflow**:

The project workflow that the team followed:

1. *Learning:*
   (a) Learning concepts and their applications, from various sources.
   (b) Usage of essential python libraries such as Pandas, Numpy, Matplotlib.
   (c) Exploring key steps and procedure to build the project.
   (d) Basics of PyTorch.

2. *Data Collection:*
   The online websites containing sources for the datasets were provided in the problem statement. After exploring and searching among the datasets, VisDrone2019 dataset was found to be the most suitable for object detection. It had pre-split folders as train, test, and validation data. Each folder contained annotations corresponding to each frame(in the case of videos dataset), or image(in the case of images dataset).

3. *Data Analysis:*
   To begin with coding, it was essential to understand the data storage hierarchy. As mentioned, a total of 6 folders were downloaded: train, val, and test folders for images and similarly for video footage.

   (a) Each of the folders in image format, consists of 2 sub folders-'annotations', 'images'. The 'annotations' sub folder consisted of annotation text files corresponding to each image stored in the 'images' sub folder.

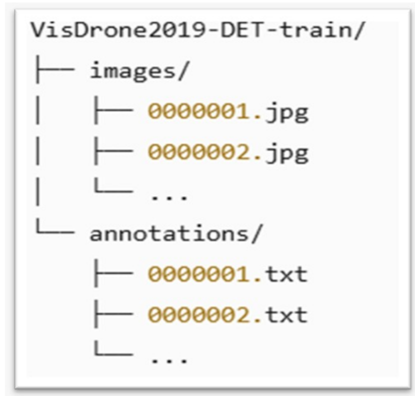

Figure 1: Dataset Hierarchy of image dataset folders

(b) Each of the folders in video format, consists of 2 sub folders- 'annotations', 'sequences'. The 'sequences' sub folder consists of multiple sub folders for multiple video footage, each of which contains the frames as (frame-name).jpg files. The 'annotations' sub folder consisted of annotation text files corresponding to each frame stored in each of the video footage sub folders.

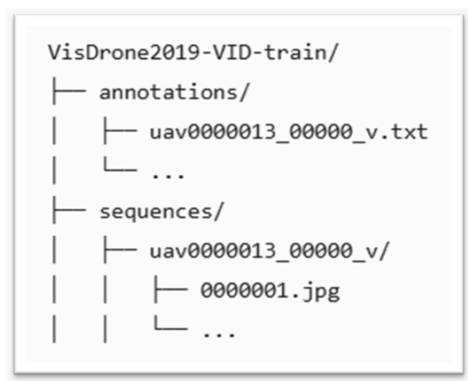

Figure 2: Dataset Hierarchy of video footage folders

4. *Data Pre-processing:*

(a) Pre-processing the data includes transforming raw, messy, and inconsistent data into a clean and structured format, removing noise, and making it suitable for the next step. YOLOv8n model was selected for training. This required the images/ frames to be resized to 928x928 pixels.

(b) And, this model also requires the annotations to be converted to a specific format suitable for yolo model implementation. These requirements were satisfied during data pre-processing.

(c) Converting RGB images to grayscale and canny edges is also a crucial step of pre-processing, but yolo models are designed to learn from the full color information present in RGB images, hence this was not a required step.

(d) The annotations were grouped using the target ids. This would be beneficial in future for object tracking as well. This was done by grouping annotations according to a 2 dimensional tuple, grouping object category with target ids.

5. *Train-test splitting:*

   (a) According to a general flow of any ML project, the next step is to split the data into training and testing data.

   (b) But the VisDrone2019 dataset already had separate folders containing distinct training, testing, and validation data. So this step was not required.

6. *Model Training:* Model training for image dataset was done on google colaboratory as it offered a better GPU for running 100 epochs. Due to issues faced in uploading folders on drive, training for videos dataset was done on VScode using local GPU.

   (a) YAML file was created, which serves as a configuration file that defines and manages key parameters, settings, and structures needed for the training process. The simple, indentation-based syntax of YAML makes it easy for users to read, write, and understand configuration files, even for complex projects.

   (b) The YOLOv8n model was trained for image dataset for 100 epochs on Google Colaboratory, and for video dataset for —epochs on VS-code.

7. *Evaluation*: The model was evaluated, with unsatisfactory results. Thus, it requires changes as it has huge scope for improvement.

- **Progress and Current Status**:

1. YOLOv8n model has been trained for images dataset, and run on 100 epochs. The model has been evaluated, but mAP, precision, F1 score, and other metric values are not satisfactory. The immediate next step will, definitely, be to improve on evaluation metric values.

2. The model is under training for videos dataset, and is not complete due to some errors, and efforts towards debugging are in progress.

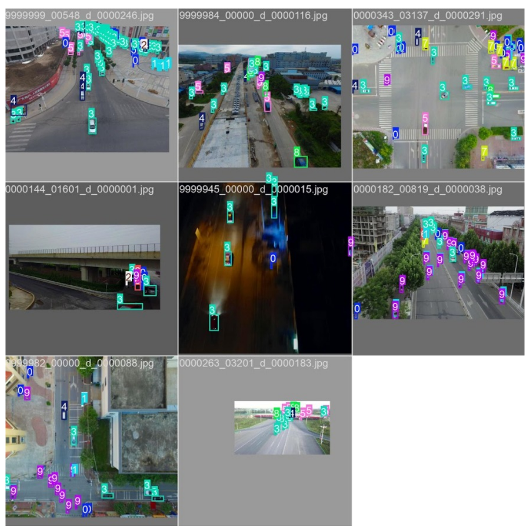

Figure 3: Images after Training

This was done for 100 epochs, which was successfully completed on Google Colaboratory, after facing multiple issues and actively resolving all of them. The following image shows the output.

```
                 all        548      38759      0.496      0.376      0.389      0.232

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     98/100      8.11G       1.27     0.8784     0.9276        265        640: 100%|██████████| 405/405 [01:57<00:00,  3.44it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 18/18 [00:07<00:00,  2.49it/s]
                   all        548      38759      0.489      0.379      0.388      0.232

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
     99/100      8.13G      1.268     0.8746     0.9276        529        640: 100%|██████████| 405/405 [01:56<00:00,  3.47it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 18/18 [00:07<00:00,  2.32it/s]
                   all        548      38759      0.488      0.378      0.388      0.231

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    100/100      6.52G      1.264     0.8699     0.9251        321        640: 100%|██████████| 405/405 [01:57<00:00,  3.45it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 18/18 [00:08<00:00,  2.25it/s]
                   all        548      38759      0.485       0.38      0.389      0.232

45 epochs completed in 1.656 hours.
Optimizer stripped from /content/drive/MyDrive/visdrone_train/yolov8/weights/last.pt, 6.3MB
Optimizer stripped from /content/drive/MyDrive/visdrone_train/yolov8/weights/best.pt, 6.3MB

Validating /content/drive/MyDrive/visdrone_train/yolov8/weights/best.pt...
Ultralytics 8.3.160 🚀 Python-3.11.13 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,007,598 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 18/18 [00:13<00:00,  1.33it/s]
                   all        548      38759      0.499      0.377      0.389      0.232
            pedestrian        520       8844      0.508      0.382      0.407       0.18
                people        482       5125      0.564      0.286      0.344      0.133
               bicycle        364       1287      0.314      0.131      0.128     0.0533
                   car        515      14064      0.734      0.784      0.816      0.579
                   van        421       1975      0.546      0.443       0.46      0.325
                 truck        266        750      0.457      0.368      0.344       0.23
              tricycle        337       1045      0.435      0.291      0.271      0.146
       awning-tricycle        220        532      0.267       0.19      0.147     0.0919
                   bus        131        251       0.63      0.474      0.539      0.392
                 motor        485       4886      0.534      0.419      0.437      0.192
Speed: 0.3ms preprocess, 2.4ms inference, 0.0ms loss, 5.0ms postprocess per image
Results saved to /content/drive/MyDrive/visdrone_train/yolov8
ultralytics.utils.metrics.DetMetrics object with attributes:
```

Figure 4: Output after running for 100 epochs

- **Conclusion**:

The trained model, with images resized to 640x640, was evaluated on evaluation metrics. The results were as follows:

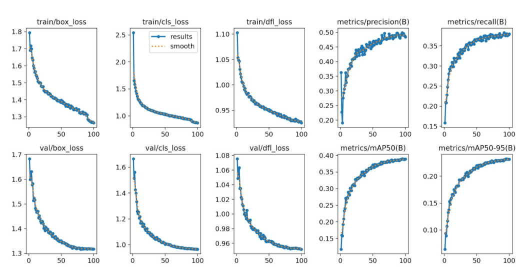

Figure 5: Results after training images resized to 640x640

Low accuracy score might have occured due to class imbalance (more ’cars’ objects, and comparatively very few ’tricycles’ objects).

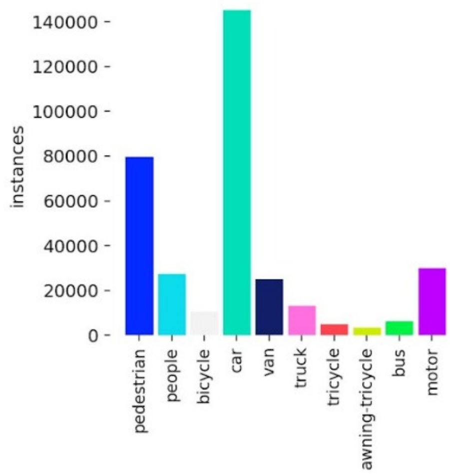

Figure 6: Class Imbalance

Other reasons could be that the images being resized to 640x640, while larger dimensions are actually required; (and other such issues). This was one-step rectified by changing image resize to 928x928. Following were the results after rectification (results were improved but still not staisfactory).

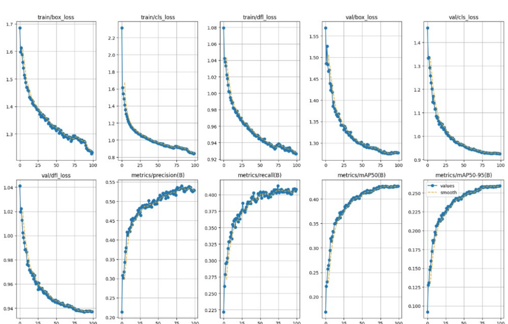

Figure 7: Results after training images resized to 928x928

The immediate next focus of the team will be to further improve on the evaluation metrics.

References

- Machine Learning Google Crash Course: https://developers.google.com/machine-learning/crash-course
- Machine Learning playlist: https://youtube.com/playlist?list=PLfFghEzKVmjuYSU8vohCLxkKuod8ulUF1
- OpenCV from freeCodeCamp.org: https://youtu.be/oXlwWbU8l2o?si=ocHU8b7W6694Zhv6