VisDrone MOT Dataset - Object Detection and Tracking

Ilias Ouzounis

January 7, 2026

1 Problem Statement

Object detection and tracking from arial data is a challenging task, but one of high importance for various applications such as surveillance and traffic monitoring. It is usually performed in a two-step process, where objects are first detected in each frame and then associated across frames to form trajectories. Recent advances in deep learning have led to significant improvements in both object detection and tracking performance.

The VisDrone MOT dataset [1] is a large-scale benchmark for object detection and tracking in drone-captured videos. It contains over 10,000 video sequences with more than 2.5 million annotated bounding boxes for various object categories, including pedestrians, vehicles, and bicycles. The dataset presents several challenges found in real-world scenarios, such as occlusions, camera motion and varying object scales.

2 Object Detection

Object detection is a fundamental task in computer vision that involves identifying both the location and the classes of objects within an image. It is a much more complex task compared to image classification, which only assigns a single label to an entire image, as there can be multiple objects of different classes per image, sometimes even overlapping. Modern object detection algorithms typically employ deep learning techniques, particularly convolutional neural networks (CNNs) or transformers, to learn hierarchical features from the input images.

Solutions to the object detection problem often rely on foundational models like YOLO [2] that have learned hierarchical features from large-scale datasets such as COCO [3] or ImageNet [4]. These models can then be fine-tuned on datasets like the VisDrone dataset to adapt to the specific characteristics of drone-captured imagery and the classes of interest.

2.1 Evaluation Metrics

To assess the performances of object detection models a standard evaluation metric is the Mean Average Precision (mAP). This metric combines both the classification of the objects as well as the localization accuracy, providing a comprehensive measure of the model's effectiveness. The mAP checks the overlap between the predicted bounding boxes and the ground truth boxes using the Intersection over Union (IoU) metric. A prediction is considered correct if the IoU exceeds a certain threshold and the predicted class matches the ground truth class. To

cover various levels of localization precision, mAP is often reported at multiple IoU thresholds, such as 0.5 (mAP@50) and 0.75 (mAP@75), along with an average across all levels (0.50 to 0.95). Additionally, mAP can be calculated separately for different object sizes (small, medium, large) to provide insights into the model's performance across various scales.

2.2 State of the Art Methods

As of 2026, the state-of-the-art methods on the VisDrone dataset mainly utilize CNNs [2, 5] or specialized transformers [6]. Note that due to the challenging nature of the VidDrone dataset, the mAP scores are generally lower compared to other detection datasets like COCO. This primarily due to the high object density of VisDrone images and the small size of many objects. As a result, the leading models are specialized architectures designed to handle small and densely packed objects effectively.

Table 2.1: State-of-the-Art models on the VisDrone-DET dataset.

Models	**mAP** (%)	**mAP**$_{50}$ (%)
YOLOv11-M [7]	27.7	45.2
RT-DETR (R18) [8]	26.7	45.3
Deformable-DETR [9]	27.1	49.2
DIF-DETR [6]	30.5	49.7
FSTDNet [5]	31.9	53.6

2.3 YOLOv8 for VisDrone Detection

The first model we experimented with is YOLOv8n, a very lightweight model from the YOLO family, still able to deliver decent results. This model was selected for its simplicity, taking into consideration the limited available hardware. Initially, we evaluated the model out-of-the-box, only adjusting the class names to match those of the VisDrone dataset. The results were very underwhelming, with a mAP of only 6.7% on the test set. This poor performance can be largely attributed to the domain gap between the test set and the COCO dataset, on which YOLOv8 was originally trained. A lot of the classes in VisDrone, such as *Van* and *Tricyle* are not present in COCO, leading to misclassifications for every instance of these classes. In COCO there is no distinction between *Pedestrian* and *Person* so we decided to assign the COCO class to *Pedestrian* as it was more common. In Figure (2.1) we can see the mAP performance per class and in Figure (2.2) some examples of the predictions made by the model.

From these results, we can identify the limitations of the model. THe model is unable to accurately distinguish between classes that are not present in its training data and often mislabels them as the closest available class. The fine-tuning of the model seems essential to improve its performance on this specific dataset.

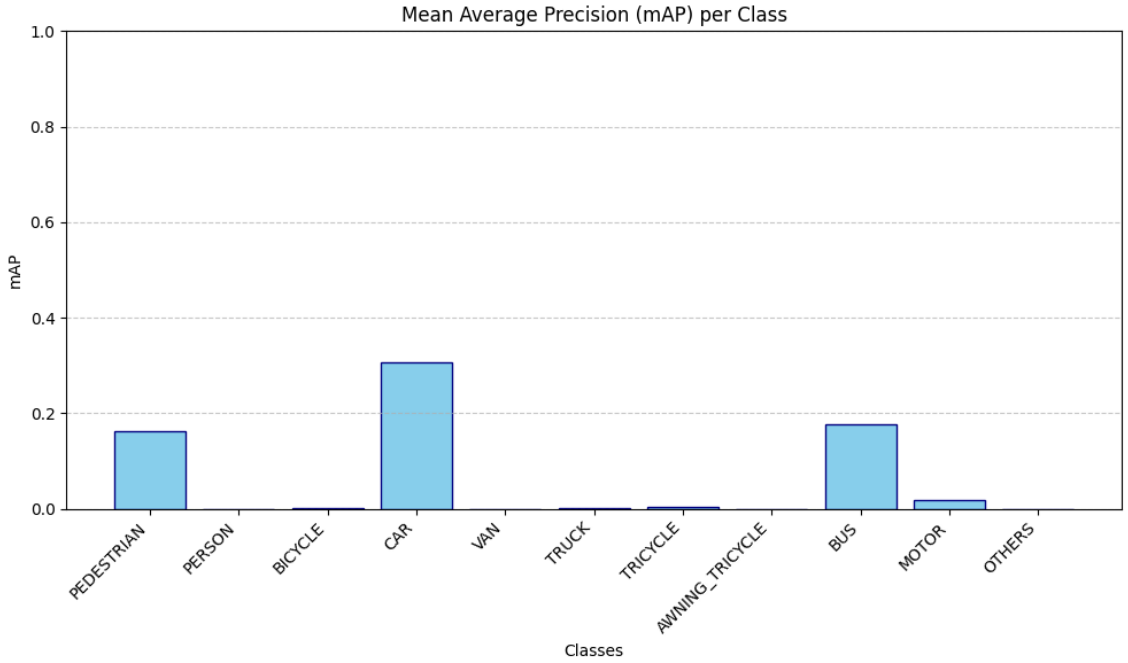

Figure 2.1: mAP per class for YOLOv8n on VisDrone-MOT test set. Classes not present in COCO have 0 mAP.

Figure 2.2: Examples of YOLOv8n detections (left) vs ground truth (right) on VisDrone images.

2.4 Fine Tuning YOLOv8 for VisDrone

To address the limitations observed with the out-of-the-box YOLOv8n model, we proceeded to fine-tune the model specifically for the VisDrone dataset. Training times for state-of-the-art models border on 40 to 50 hours on high-end GPUs, which was not feasible to match and train our own model. Instead, we borrow a pre-trained YOLOv8s model that has already been fine-tuned on the VisDrone dataset and implement it in our evaluation pipeline. The results improve significantly with a respectable mAP of 18.73% on the test set and a mAP@50 of 37.65%. The

per-class performance is shown in Figure (2.3) and some qualitative results in Figure (2.4).

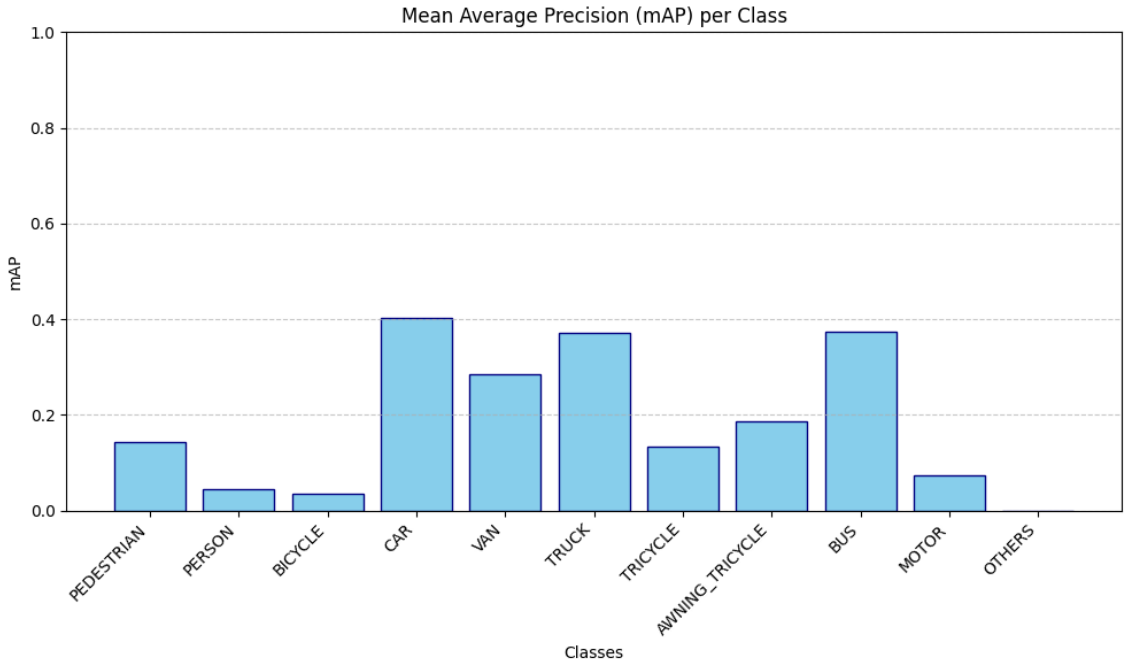

Figure 2.3: mAP per class for fine-tuned YOLOv8s on VisDrone-MOT test set. All classes show improved performance.

Figure 2.4: Examples of the fine-tuned YOLOv8s detections (left) vs ground truth (right) on VisDrone images.

From the results in Figure (2.4) we get some valuable insights on the strengths and weaknesses of the fine-tuned model. Our first observation is that the model now tries to identify all subjects in the image, even the very small ones, which is a significant improvement over the previous model. Because for the Ground Truth annotations it is not clear which objects are annotated and which are not, the model has learned to be more inclusive in its detections.

However, this leads to an increase of false positives, as seen in the second examples where cars in the distance are not annotated but detected by the model. Another observation is that the model still struggles to distinguish between certain classes, such as *Tricyle* and *Awing Tricyle* or *Van* and *Car*. Classes that are visually similar, especially when viewed from a drone's perspective and at a distance remain a challenge for even the state-of-the-art models. Finally, if we look at the mAP for different object sizes, we can clearly see where performance dips. Large objects, larger than 96x96 pixels, achieve a mAP of 30.39%, much bigger than the total mAP of 18.63%. On the other hand, small objects, smaller than 32x32 pixels, only achieve a mAP of 6.65% and medium objects (32x32 to 96x96 pixels) a mAP of 23.35%. These results further highlight the challenges associated with detecting small objects in drone imagery and the need for models that can work better at these scales.

All of this suggests that while fine-tuning has improved overall performance, there is still room for improvement in class differentiation, possibly through further training or architectural adjustments. Common techniques to improve performance, especially for small objects, include splitting the images into smaller patches or using specialized small object detection heads in addition to the main detection architecture.

3 Object Tracking

Object tracking is a task that pairs and complements object detection nicely. In many applications, it is not only important to know where an object is, but also track how it moves over time. This is particularly relevant in video analysis, surveillance, and autonomous systems, where understanding the trajectory and behavior of objects is crucial.

Object tracking algorithms typically build upon object detection pipelines that first identify objects in individual frames. The key idea is to associate each detected object with a unique identifier and then try to match these identifiers in subsequent frames based on various criteria such as spatial proximity, appearance features, and motion patterns. The most obvious and straightforward approach is to use the Intersection over Union (IoU) metric to match bounding boxes between frames. The reasoning is that an object in motion will likely not change its position drastically between consecutive frames, so the bounding boxes with the highest IoU are likely to correspond to the same object. Assuming that the bounding box predictions are accurate, this simple method can yield decent tracking results across frames.

Some challenges with this approach include objects overlapping each other which can lead to identity switches and model inaccuracies that can lead to a miss in detection in some frames, breaking the tracking continuity. To improve on this basic approach, a more sophisticated method utilizing Kalaman Filters can be implemented to better predict the future position of objects based on their past motion. This alleviates some of the issues with occlusions and missed detections, as the filter can provide a prediction even when the object is not detected in a frame.

Still, the method using Kalaman Filters relies heavily on smooth and in-frame movement of objects. In scenarios where objects move erratically or leave and re-enter the frame, more advanced techniques may be required. More complex methods involve using a Neural Network to learn appearance features of objects and use these extracted features to match objects across frames. This can be particularly useful in crowded scenes where multiple objects of the same class are present, although it comes at the cost of increased computational complexity.

4 Reproducibility of Results

The source code for all experiments is available on GitHub at https://github.com/HliasOuzounis/VisDrone-detection. The repository contains scripts to download the VisDrone dataset, set up the environment, and run the detection and tracking experiments as described in this report.

Detailed instructions are provided in the README file to guide users through the process of reproducing the results.

References

[1] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[2] Ultralytics. Yolov11: Real-time object detection for aerial platforms. *arXiv preprint arXiv:2400.00000*, 2024.

[3] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. *European Conference on Computer Vision*, pages 740–755, 2014.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[5] Hong Chen, Linqiang Liu, et al. An effective and lightweight full-scale target detection network for uav images based on deformable convolutions and multi-scale contextual feature optimization. *Remote Sensing*, 16(16):2944, 2024.

[6] Jing Wang, Hejiang Li, and Caihong Huangfu. Dif-detr: Dynamic interactive fusion transformer with adaptive feature enhancement for efficient aerial small object detection. *Journal of Computer Science and Artificial Intelligence*, 5(3), 2025.

[7] Ultralytics. Ultralytics yolov11, 2024.

[8] Wenyu Lv, Shangliang Zhao, Qinyao Xu, Jinman Wei, Guanzhong Cheng, Cheng Dang, Yi Liu, Jie Chen, Haifeng Qin, and Qiwen Lai. Detrs beat yolo on real-time object detection. *CVPR*, 2024.

[9] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.