Exploring Object Detection from UAV Aerial Perspectives with Deep Learning

Abstract:
Unmanned Aerial Vehicles (UAVs) have shown potential in tasks like traffic monitoring, surveillance, and disaster relief. However, object detection from aerial imagery poses unique challenges due to variations in object scale, cluttered environments, and altitude differences. This project investigates and compares YOLO-based deep learning models (particularly YOLOv11n and YOLOv11x) on the VisDrone dataset. Through systematic hyperparameter tuning, model comparisons, and integration with real-time detection and tracking capabilities, we present a pipeline suitable for UAV deployments. YOLOv11x achieves superior accuracy, while real-time enhancements, including object locking and directional guidance, demonstrate the system's potential for dynamic aerial applications.

---

1. Introduction

The increasing use of UAVs in various fields—such as traffic analysis, search-and-rescue operations, and infrastructure inspection—demands robust and efficient object detection frameworks. Unlike ground-based perspectives, aerial views introduce complexities: objects can be small, scenes dense, and backgrounds cluttered. Traditional detectors often struggle under these conditions.

Recent advancements in deep learning have produced object detection models like YOLO that combine accuracy with real-time capability. Among emerging variants, YOLOv11n and YOLOv11x differ significantly in scale and complexity. YOLOv11n, a lightweight model, may be suitable for resource-constrained environments, while YOLOv11x, featuring a deeper and richer architecture, could improve accuracy but at higher computational cost.

Objectives:

1. **Robust Aerial Object Detection:** Implement and evaluate YOLO models on the VisDrone dataset to identify stable, accurate configurations.
2. **Real-Time Adaptation:** Develop a real-time detection pipeline, integrating tracking, trajectory visualization, and object locking for interactive UAV operation.

The code, dataset processing scripts, additional plots, and more detailed documentation are available in my GitHub repository:
https://github.com/xuy50/ecgr8119-final_project.

---

2. Methods and Materials

2.1 Dataset: VisDrone

The VisDrone dataset, developed for UAV vision tasks, includes over 10,000 static images and multiple video sequences. Its annotations cover diverse object classes (e.g., pedestrians, cars, vans, trucks) encountered in real UAV footage. One key challenge was converting VisDrone's raw annotations into YOLO-compatible format: from pixel-based bounding boxes to normalized coordinates (`class_id center_x center_y width height`).

2.2 Model Selection

We focused on two YOLOv11 variants:

- **YOLOv11n:** Lightweight, fewer parameters, faster training, but prone to overfitting on complex aerial scenes.
- **YOLOv11x:** Larger, deeper architecture, increased capacity for nuanced feature extraction, potentially more stable and accurate.

2.3 Training and Hyperparameters

We trained both models using the Ultralytics YOLO framework. Key parameters:

- **Epochs:** Tested 50 to 100 epochs; 100 epochs yielded stable performance before overfitting.
- **Batch Size:** Ranged from 8 to 84; larger batches (e.g., 84) improved stability without exhausting GPU memory.
- **Image Resolution:** Compared 640x640 and 720x720; 720x720 slightly improved accuracy but increased computation.
- **Optimizers:** SGD, AdamW, and Adam were tested. SGD provided the most stable and generalizable results, reducing overfitting compared to Adam.

2.4 Real-Time System Integration

A webcam-based pipeline was implemented to simulate UAV camera feeds:

- **Basic Detection:** Frames are passed through YOLO for bounding box predictions.
- **Tracking & Trajectory Visualization:** YOLO's built-in tracking mode assigns persistent IDs. Historical positions of each detected object are stored to visualize trajectories, useful for monitoring movement patterns.
- **Object Locking & Directional Guidance:** By clicking on a displayed object, the user "locks" it. The system then calculates offsets relative to the frame's center, providing directions (e.g., "Move Left") to help center the object in view.

2.5 Implementation Details and Code Modifications

Early attempts with YOLO's tracking code caused frame drops and ID inconsistencies. Modifications were made to improve ID persistence and reduce computational overhead. After refining trajectory buffering and ID assignment logic, tracking stabilized, enabling smooth real-time visualization.

All code, including data preprocessing scripts, model configuration files, and real-time detection enhancements, is available in my GitHub repository.

3. Results

3.1 Quantitative Performance

YOLOv11n initially converged quickly but suffered from higher classification and box regression losses, indicating limited capacity to handle complex aerial features. By contrast, YOLOv11x converged more steadily, achieving lower losses and improved mean Average Precision (mAP) scores, reflecting better recognition of varied object sizes and cluttered scenes.

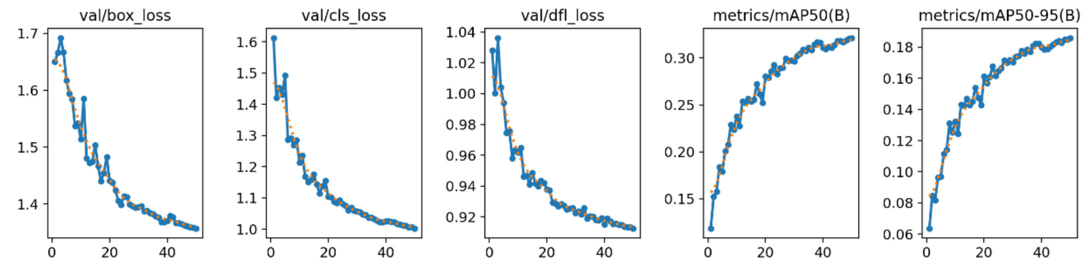


Figure 1: Training loss trends for YOLOv11n with SGD, illustrating higher final losses.

In comparison, YOLOv11x's curves showed smoother and faster convergence with lower final losses.

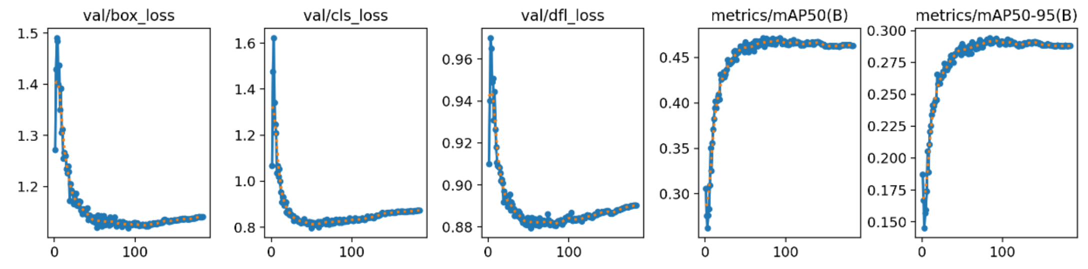
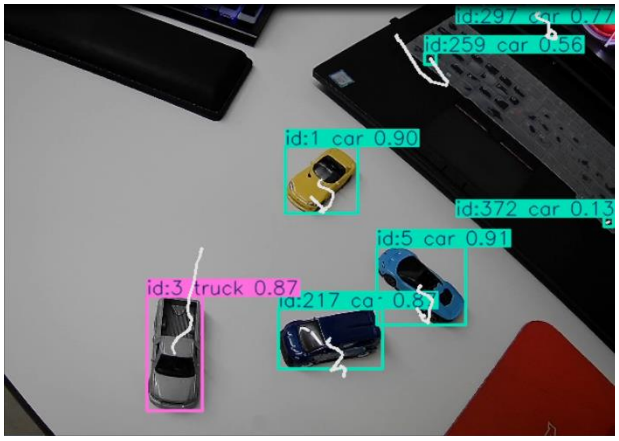


Figure 2: YOLOv11x training performance using SGD and larger batch size, highlighting more stable and lower loss convergence.

3.2 Real-Time Detection and Tracking

The real-time pipeline successfully processed frames at interactive rates. Objects such as toy cars (emulating vehicles in UAV footage) were consistently detected. Tracking ensured object IDs remained stable across frames, enabling trajectory visualization. By locking on selected objects, the system provided directional cues to maintain focus on targets.



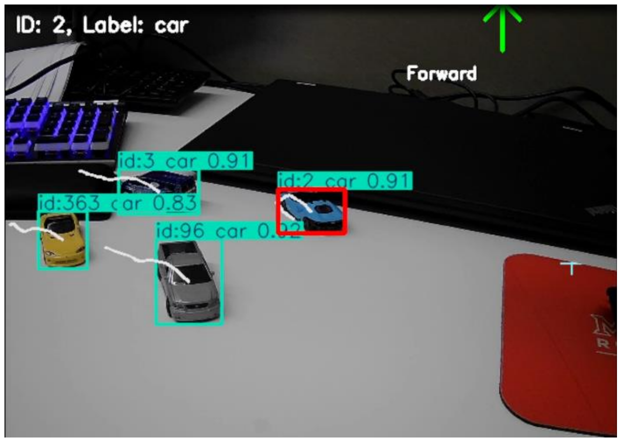


Figure 3, 4: Real-time detection demonstration with YOLOv11x. Bounding boxes and trajectories are overlaid, enabling interactive object tracking and guidance.

---

4. Discussion

This study demonstrates the importance of model choice, hyperparameter tuning, and pipeline optimization in UAV-based object detection. YOLOv11x outperformed its lightweight counterpart, YOLOv11n, due to its deeper architecture and greater representational capacity. Although YOLOv11x is heavier computationally, careful optimization and code improvements maintained feasible performance for real-time inference.

Integrating tracking and trajectory visualization added contextual understanding of object movements, crucial for UAV applications such as traffic flow analysis. Object locking and directional guidance enhanced user interaction, potentially aiding UAV pilots in following targets of interest.

Remaining challenges include occasional misclassifications of less frequent classes and the need for more sophisticated tracking algorithms (e.g., DeepSORT) to handle complex occlusions and maintain stable IDs. Future work will also consider hardware integration on actual UAV platforms and further improvements to handle domain shifts or more varied scene conditions.

---

5. Individual Contributions

This is an individual course final project undertaken solely by myself. I performed all tasks including dataset preprocessing, YOLO model training and evaluation, code modifications for tracking stability, implementation of the real-time detection pipeline, and integration of trajectory visualization and object locking features. All project details, code, and documentation are available in my GitHub repository.

References (not counted in page limit)

- VisDrone Dataset: https://github.com/VisDrone/VisDrone-Dataset
- Ultralytics YOLO: https://docs.ultralytics.com/
- Project Repository (Code, Details, and Additional Plots): https://github.com/xuy50/ecgr8119-final_project