Lightweight YOLO for Real-Time UAV Detection in Urban Places

Justin-Marian Popescu
justin.popescu1605@stud.acs.upb.ro

Alfred-Andrei Pietraru
alfred.pietraru@stud.acs.upb.ro

University Politehnica of Bucharest,
Faculty of Automatic Control and Computer Science

Abstract

Unmanned Aerial Vehicles (UAVs) operating in urban environments must accurately detect surrounding objects and individuals to comply with safety regulations. A lightweight YOLO-based detection model is proposed, optimized for *Open*-category UAVs constrained by weight, power, and altitude. The architecture enables real-time inference on edge devices such as the Google Coral USB (Edge TPU), ensuring high frame-rate performance by integrating multi-scale feature learning, small-object augmentation, and adaptive loss weighting, the model improves accuracy for small and medium targets. Hardware-aware techniques, including precision quantization, pruning, and graph-level optimization, further minimize latency and energy consumption. The proposed system delivers efficient, reliable, and scalable aerial perception for applications such as traffic monitoring, threat assessment, and emergency response. All implementation resources, including code and pretrained weights, are available at github.com/AJ-AI-mas/urban-uav.

1 Introduction

One of the main rules when it comes to flying a drone, is the fact that the UAV object should never be directly above a person or a vehicle. For a human operator, this elementary rule is already well known and depending of the environment really simple to implement. One issue of the autonomous drones is that most of them do not a system responsible of enforcing this rules, making them dangerous for nearby people and due to a legislation barrier, hard to integrate into our society.

The European drone legislation categorizes flights based on operational risks, with altitude being one of the determining factors. There are three main categories: *Open*, which are low-risk operations, *Specific* and *Certified*. The latter two involve higher-risk operations and are restricted from flying over densely populated areas due to their greater weight, speed, and required altitude.

We are going to focus on drones that are part of the *Open* category, which are lightweight, energy-efficient, and they are required to fly at a maximum altitude of ~120 meters above ground level. Such drones are suitable for a wide range of applications where compliance and safety require accurate detection of people and vehicles to actively avoid overflight.

This paper aims to develop a lightweight YOLO architecture specifically optimized for real-time aerial detection tasks running on an Edge TPU. Compared to other detection frameworks such as SSD, EfficientDet, or Faster R-CNN, YOLO offers a superior trade-off between speed, accuracy, and architectural adaptability, making it the best case for constrained edge environments. Our approach focuses on maximizing detection accuracy for small and medium-sized objects under resource limitations, including restricted memory, power, and latency budgets. By combining efficient network design, hardware-aware optimization, and targeted training strategies, we enable robust object detection in complex urban scenes with high throughput and minimal energy consumption.

2 Problem Statement

The autonomous drones used in industry today rarely include intelligent systems designed to address this issue are computationally expensive, making them unsuitable for complex navigation tasks. Developing faster methods for detecting small objects, particularly people or vehicles viewed from a drone, would encourage the broader adoption of UAVs across multiple domains and scientific research.

We have to pursue two main technical objectives. The first is to evaluate multiple **YOLO versions (v5–v11)**, covering both anchor-based and anchor-free paradigms, to identify those architectures that can be efficiently compiled and executed on the embedded device. The second is to select the most suitable anchor-free and design an Edge-TPU-optimized variant that delivers higher inference speed, energy efficiency, and detection accuracy compared to the official implementations.

- Most real-time systems require continuous detection at high frame rates and image resolutions, which significantly increases computational demand.
- The Google Coral USB Accelerator has a limited number of supported operations, and the quantized version of the architecture must not exceed **8 MB** due to strict on-chip memory constraints, making efficient compression, pruning, and calibration essential.
- Real-world images often contain occlusions, motion blur, varying illumination, and complex object interactions for example, multiple people in a crowd leading to unstable detections and reduced accuracy.
- Small targets are challenging to detect due to their **minimal pixel footprint** (often below 1% of the image), with key visual cues lost during downsampling and quantization, particularly at high altitudes or long ranges.
- Lack of standardized preprocessing and annotations across UAV datasets causes inconsistencies that can lead to inconsistencies that affect quantization stability, model convergence, and cross-dataset generalization.

Overall, the challenge lies in balancing detection accuracy, inference speed, and hardware efficiency within strict computational limits. The ultimate goal is to design a lightweight, hardware-aware CNN detector that can perform efficiently real-time perception on embedded UAV systems while maintaining high accuracy and generalization across diverse aerial scenarios.

3 Objectives

The primary objective of this work is to design, train, and optimize a lightweight object detection model tailored for UAV-based perception in complex urban environments. The research aims to achieve a balanced trade-off between accuracy, computational efficiency, and real-time inference capability under the resource constraints typical of aerial platforms.

A central goal is to ensure **real-time inference** on resource-constrained UAV hardware, sustaining high frame rates with minimal inference latency on embedded accelerators. The model training and deployment pipeline is optimized for **end-to-end efficiency**, enabling continuous video-stream inference through frame-level decoding, adaptive resizing, and batched execution. During training, both **trainable and non-trainable parameters** are carefully balanced to minimize memory footprint while preserving representational capacity.

The optimization process incorporates latency-aware objectives, mixed-precision computation, and gradient-efficient updates to enhance throughput without sacrificing accuracy. These design choices collectively ensure stable, low-latency operation and robust detection performance under dynamic urban flight conditions.

Another key objective is to maximize **detection accuracy for small and medium-scale objects**, which are predominant in aerial imagery yet remain difficult to detect due to limited pixel resolution, partial occlusions, and cluttered urban backgrounds. To address these challenges, the model employs multi-scale feature learning, small-object augmentation, and adaptive loss weighting to strengthen sensitivity and localization precision.

For deployment, the system is optimized for **hardware-efficient execution** through structured pruning, precision-aware quantization, and compiler-level graph acceleration **TensorRT**. The quantization strategy is tailored to each platform **PyTorch FP16 mixed-precision** is employed on GPU systems and TensorRT platforms to maximize throughput and stability, whereas **full TFLite INT8 quantization** is used for **Edge TPU** deployment, ensuring end-to-end integer inference with no CPU fallback. Quantization-aware training (QAT) and post-training calibration (PTQ) help preserve accuracy, particularly for small-object detection. These optimizations yield a compact, low-latency, and energy-efficient detection framework capable of real-time performance across diverse UAV deployment scenarios.

4 Datasets

To train, validate, and evaluate the proposed UAV detection model, four publicly available aerial object detection datasets are employed. For consistency across experiments, all datasets are normalized to follow the VisDrone annotation format. Object class labels from the remaining datasets are renamed to match this unified schema, ensuring compatibility and balanced label representation during training and evaluation.

Objects without corresponding ground-truth mappings are excluded from the loss computation to prevent bias. In the first phase, the model is trained on the VisDrone dataset, and in the subsequent phase, only the trainable parameters are fine-tuned on the remaining three datasets to refine performance and evaluate cross-domain generalization under varying flight and environmental conditions.

- **VisDrone**[1]: Serves as a key benchmark for evaluating small-object detection and dense urban scene performance, comprising over **10,209** images and approximately **540,000** annotated object instances collected across 14 Chinese cities under diverse weather and illumination conditions. It defines **10 object categories** including (*pedestrian, person, car, van, bus, truck, motor, bicycle, awning tricycle, and tricycle*), with detailed annotations supporting detection, tracking, and crowd-density analysis tasks.
- **Stanford Drone Dataset (SDD)** [2]: Collected from a bird's-eye view drone over the Stanford University campus, this dataset includes over **60 videos** and **19,000** annotated image frames with approximately **11,000 trajectories** of six different types of classes (*pedestrian, bicyclist, skateboarder, car, bus, and cart*). It offers high spatial and temporal resolution across **eight unique scenes**, supporting research in object detection, multi-agent tracking, and trajectory prediction in socially interactive environments.
- **AU-AIR** [3]: A multimodal UAV dataset designed for real-world outdoor environments, integrating synchronized RGB imagery with on-board sensor data. It contains **32,823** annotated video frames with **132,034** object instances across eight traffic-related categories (*person, car, van, truck, motorbike, bike, bus, trailer*). Data was captured at varying altitudes ($\sim 5-30m$) and camera angles ($45°-90°$), providing diverse visual conditions for evaluating real-time aerial object detection.
- **UAVDT** [4]: Comprises **77,819** images and **835,000** annotated objects across four classes (*car, truck, bus, other vehicle*). The dataset covers multiple operational conditions altitude levels (low, medium, high), occlusion levels (none, small, medium), and three weather settings (daylight, night, fog) making it one of the most comprehensive benchmarks for evaluating UAV perception robustness in traffic-dense urban environments. Each frame includes bounding box annotations with target IDs and long-term tracking multi-object detection and temporal analysis.

Apply augmentations like the HSV adjustments, geometric transformations (rotation, translation, scaling, shear), flipping up and down the images, and stochastic composition method such as mosaic. This unified normalization and augmentation pipeline ensures balanced exposure to underrepresented classes and improves generalization across datasets, providing a good benchmark ground for lightweight UAV perception systems.

5 Evaluation Metrics

To rigorously evaluate the proposed UAV detection model, we consider both **accuracy** and **computational efficiency** are assessed through a comprehensive set of metrics. These indicators measure not only the model's detection capability but also its suitability for real-time deployment on embedded computer vision devices.

- **Mean Average Precision (mAP@0.5, mAP@0.5:0.95)** The main metric for object detection, representing the mean of class-wise Average Precision (AP) computed over IoU thresholds. mAP@0.5 evaluates detections at an IoU threshold $\tau = 0.5$, while mAP@0.5:0.95 averages results across $\tau \in [0.5, 0.95]$, offering a balanced measure of detection accuracy and localization quality.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i, \quad AP_i = \int_0^1 P_i(R)\, dR, \quad IoU = \frac{|B_p \cap B_{gt}|}{|B_p \cup B_{gt}|} \tag{1}$$

- **Precision (P)** and **Recall (R)** Precision measures the proportion of correct detections among all predicted boxes, while Recall quantifies the proportion of correctly identified targets among all ground-truth objects, both computed with respect to an IoU threshold.

$$P = \frac{TP(IoU \geq \tau)}{TP(IoU \geq \tau) + FP}, \qquad R = \frac{TP(IoU \geq \tau)}{TP(IoU \geq \tau) + FN} \tag{2}$$

- **F1-Score** Defined as the harmonic mean of Precision and Recall, the F_1 provides a balanced measure of detection performance. In the context of UAV-based detection, it reflects how effectively the model maintains an optimal trade-off between false positives and false negatives when predicting bounding boxes.

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} = \frac{2TP(IoU \geq \tau)}{2TP(IoU \geq \tau) + FP + FN} \tag{3}$$

- **Inference Latency (ms/frame)** and **Frames Per Second (FPS)** Inference latency represents the end-to-end time to process a single frame, including image preprocessing (t_{pre}), network forward pass (t_{net}), and post-processing (t_{post}) such as bounding-box decoding and Non-Maximum Suppression (NMS). The overall throughput is expressed as:

$$t_{\text{inf}} = t_{\text{pre}} + t_{\text{net}} + t_{\text{post}}, \qquad FPS = \frac{1000}{t_{\text{inf}}} \tag{4}$$

- **GFLOPs and Model Capacity** These metrics describe the network's computational footprint and efficiency. Smaller model size and lower parameter count reduce memory usage and improve throughput, making the model more suitable for real-time UAV deployment.

Collectively, these metrics provide a comprehensive assessment of the CNN architecture detector's **accuracy, reduce compute time, and real-time performance**. The goal is to maximize mAP and F_1-Score while minimizing inference latency, computational cost (FLOPs), and parameter count, ensuring fast, energy-efficient operation on resource-constrained UAV hardware.

6 Experimental Setup

The experiments will be conducted using the Ultralytics YOLO repository for model training, validation, and exporting which is written in Pytorch. Visualization of predictions and datasets will be performed using Voxel51. Tensorboard will be used for monitoring the evaluation metrics across epochs. The model architecture will be defined in a YAML configuration file, which will then be used to generate the corresponding model for training and deployment.

The model export pipeline involves multiple steps: first, the trained PyTorch model is converted to ONNX, then to TensorFlow, followed by conversion to TensorFlow Lite, and finally adapted to be compatible with the Edge TPU board. Quantization is applied to produce an optimized INT8 TensorFlow Lite model, which is then compiled using the Edge TPU Compiler before deployment and inference on the target hardware.

From a hardware perspective, training and validation will be performed on a GeForce RTX 5090 GPU with 32GB of memory. Testing will be done using the Google Coral USB Accelerator after the export step.

The model was trained using an initial learning rate of 0.01 with the stochastic gradient descent (SGD) optimizer, a momentum of 0.9, and a weight decay of 5e-4. Due to the small size of the target objects, input images were resized to a resolution of 640 × 640 pixels. Training was performed for 200 epochs with a batch size of 32. Non-Maximum Suppression (NMS) was applied independently for each class to remove duplicate detections, with a confidence threshold of 0.25, an Intersection over Union (IoU) threshold of 0.45, and a maximum of 300 retained detections per image.

References

[1] Dawei Du, Pengfei Zhu, Longyin Wen, et al. Visdrone-det2019: The vision meets drone object detection in image challenge results. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 213–226, 2019. doi: 10.1109/ICCVW.2019.00030. URL https://arxiv.org/abs/1804.07437.

[2] Joshua Andle, Nicholas Soucy, Simon Socolow, and Salimeh Yasaei Sekeh. The stanford drone dataset is more complex than we think: An analysis of key characteristics, 2022. URL https://arxiv.org/abs/2203.11743.

[3] Ilker Bozcan and Erdal Kayacan. Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance, 2020. URL https://arxiv.org/abs/2001.11737.

[4] Dataset Ninja. Visualization tools for uavdt dataset. https://datasetninja.com/uavdt, oct 2025. URL https://datasetninja.com/uavdt. visited on 2025-10-18.