FPT Education

FPT UNIVERSITY

SMALL OBJECT DETECTION IN UAV IMAGES

by

Lieu Bach Thanh

Nguyen Chi Khang

Nguyen Huynh Lam

DEPARTMENT OF ITS
THE FPT UNIVERSITY HO CHI MINH CITY

April 2024

SMALL OBJECT DETECTION IN UAV IMAGES

by

Lieu Bach Thanh

Nguyen Chi Khang

Nguyen Huynh Lam

Supervisor: Le Thanh Hai, Ph.D.

A final year capstone project submitted in partial fulfillment of the requirement for the Degree of Bachelor of Artificial Intelligence in Computer Science

ACKNOWLEDGMENTS

We extend our heartfelt gratitude to the individuals and organizations whose invaluable contributions and support were instrumental in the completion of this project:

- Our profound appreciation goes to all who generously aided us throughout this endeavor. A special acknowledgment is owed to our instructor, Mr. Le Thanh Hai, whose insightful suggestions and unwavering encouragement played a pivotal role in guiding our project to fruition, particularly in the formulation of this report.

- We wish to express our sincere thanks to Mr. Nguyen Quoc Trung and Mr. Le Phu Nguyen for their guidance and constructive feedback as reviewers. Their invaluable input during the project review significantly enhanced both our presentation skills and the quality of our report.

- We are indebted to Google Colab and the Kaggle platform for graciously providing us with access to free GPUs, which proved indispensable in conducting our experiments.

- Lastly, we extend our gratitude to our friends and family for their unwavering love and support throughout our academic journey.

We extend our thanks to all who contributed, large and small, without whom this project would not have been possible. Your support is deeply appreciated. Thank you.

AUTHOR CONTRIBUTIONS

Methodology, Bach Thanh, Chi Khang and Huynh Lam; software, Bach Thanh.; validation, Bach Thanh, Chi Khang, and Huynh Lam; formal analysis, Huynh Lam; investigation, Chi Khang.; resources, Bach Thanh, Chi Khang, and Huynh Lam; data curation, Chi Khang; writing—original draft preparation, Chi Khang, Huynh Lam.; writing—review and editing, Bach Thanh, Chi Khang and Huynh Lam; visualization, Huynh Lam.; supervision, Bach Thanh; project administration, Bach Thanh. All authors have read and agreed to the Final Capstone Project document.

ABSTRACT

Abstract

This paper proposes an improved YOLOv5 algorithm for small object detection in unmanned aerial vehicle (UAV) images. Several modifications are made to enhance the model's performance, including adding a prediction head to handle large-scale variance, integrating a Channel Feature Fusion with an Involution (CFFI) block to reduce information loss, applying a Convolutional Block Attention Module (CBAM) to focus on important spatial and channel features, and using a C3 structure with a Transformer block (C3TR) to capture contextual information. The proposed method also employs Soft Non-Maximum Suppression for improved bounding box scoring in dense scenes. Extensive experiments on the VisDrone2019 dataset demonstrate the effectiveness of these modifications, with the enhanced model outperforming other single-stage detectors and state-of-the-art single-stage detectors like YOLOv8s by a significant margin in terms of mean Average Precision (mAP). Achieving a notable mAP50 of 44.2% and mAP50:95 of 27.3% on the test set. The performance gains are attributed to the integration of attention mechanisms that help the model focus on crucial features for detecting small objects.

Keywords: small object detection, unmanned aerial vehicles, object detection, attention mechanisms, YOLOv5, VisDrone dataset.

Contents

List of Figures

List of Tables

1 INTRODUCTION

Drones, also known as Unmanned Aerial Vehicles (UAVs), are aircraft operated remotely or through pre-programmed controls without a human pilot, crew, or passengers on board. These flying machines are equipped with built-in sensors and cameras.

As human innovation and development, drone manufacturing and control technologies have become increasingly sophisticated. New generations of drones which are lightweight, compact, and affordable, are now widely utilized in various sectors, including agriculture, disaster relief, safety prevention, and express delivery, significantly impacting people's productivity and daily lives. With the continuous expansion of drone applications, object detection plays an increasingly important role as a key link in the missions carried out by UAVs and has very great research significance.

The advent of large-scale benchmark datasets, such as MS COCO and PASCAL VOC, UAVDT, and VISDRONE2019, ... has led to significant advancements in object detection tasks through the use of deep convolutional neural networks (CNNs). However, object detection in drone-captured images presents unique challenges due to the high altitude of the images and the small size and limited viewing angles of the objects:

- **Tiny Targets:** UAV images can contain objects of vastly different sizes, from buildings down to people and animals. Small objects take up a tiny fraction of the image, making them difficult to detect.

Figure 1: Tiny Object

- **Background Complexity:** UAV images often contain densely populated object areas with numerous identical entities, thereby increasing the likelihood of false positives. Additionally, substantial noise information within the UAV image background can attenuate or obscure the object, complicating continuous and comprehensive detection.

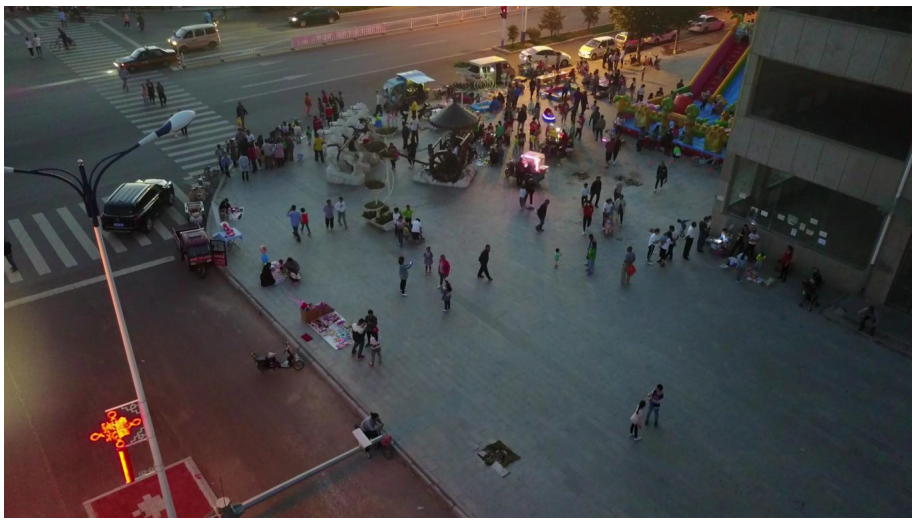


Figure 2: Background Complex

- **Category Imbalance:** Images captured by UAVs may exhibit category imbalance issues, such as a preponderance of objects within one category and a paucity within another, leading to a detector bias towards predicting categories with a larger number of instances.

Figure 3: Imbalanced Category

- **Problem with Object Rotation:** When using drones to capture images, objects can be positioned and oriented in any way. This rotation can induce morphological and visual alterations in the object's representation within the image, thereby undermining the efficacy of object detection algorithms that are predicated on shape and appearance.

Figure 4: Complexity of Rotated Object Detection

Therefore, the quest for a highly accurate object detection algorithm suitable for drone platforms has become a hot topic in the field of object detection. The current mainstream object detection algorithms mainly use deep learning methods, and we classify them into two categories: two-stage and one-stage. The R-CNN series represents the two-stage strategy, while YOLO exemplifies the one-stage approach, which is one of the most popular frameworks. You Only Look Once (YOLO), a one-stage object detection algorithm, is dominating UAV systems due to its low latency and high accuracy. It takes an image as input and outputs the information of the objects in one stage. The lightweight model can achieve real-time object detection in UAV systems.

In this paper, we propose an improved YOLOv5 algorithm, we have made several modifications to improve the performance of the network. Firstly, we observed that YOLOv5 is not adept at detecting extremely small objects so we added an additional prediction head to detect small objects from feature maps at a higher

resolution. Secondly, a Channel Feature Fusion with Involution (CFFI)[1] has been added between the Backbone and the Neek to reduce the loss of semantic information, Involution blocks can selectively focus on crucial regions of the image, potentially leading to better performance in tasks like object detection, especially for small objects. Then we apply a Convolutional Block Attention Module (CBAM)[2] after C3the block in the Neck, by applying an attention mechanism that focuses on the most important features in an image, particularly for small objects (Convolutional Block Attention Module)CBAM improves the total Performance while lower computation cost. Finally, the C3 structure with a Transformer Block (C3TR) at the end backbone makes more effective use of context information, also for accurate object location in high-density scenes. The experiment result shows that our YOLO Model has improved the performance of YOLOv5 on the VisDroneDET-19 dataset which 27.3% (mAP@50:95) and 37.43%(mAP50) on the test set.

2 RELATED WORK

There are several techniques to capture small objects in UAV images, two effective ways are **anchor-based** and **anchor-free**. Anchor-based approaches can be subdivided into (1) two-stage detector: R-CNN family [3, 4, 5] adding Region Proposal Network (RPN) to accurately specify potential object locations but slows down the detection time and (2) one-stage detectors: YOLO, SSD, RentinaNet [6, 7, 8] significantly improve inference time by skip the region proposal and directly detect over a dense sampling of locations.

2.1 Data augmentation

Object detection in UAV images is more challenging. There exists a lot of tiny objects in the images shot by UAV, such as the size of the object less than 32 pixels VisDrone2019-DET [9] is a seriously unbalanced dataset, leading to the long-tail distribution problem (figure 5). A few classes like car, pedestrian, and person, account for more than 70% of the examples, while others, like tricycles, covered tricycles, and buses, are much rarer. [10] MixUp, CutMix enhanced data augment by mix samples. [11] applied mosaic augmentation for data enhancement. Combining those can improve the detection model on an unbalanced dataset.

2.2 Small object detection model

2.2.1 Ensembled multi-model for small object detection

For small object detection, one of the effective ways is to aggregate multiple models together to improve the detection accuracy of small objects. DPNet-ensemble model [12] won the best performance in the VisDrone-DET2019 object detection challenge with an Average Precision (AP) score of 29.62%. It builds upon the Feature Pyramid Network (FPN) architecture and further enhances the Cascade R-CNN [13] object detector. RetinaNet architecture [8] also were chosen to derive in this challenge and got 26.46% AP.

2.2.2 Improved YOLOv5 model

You Only Look Once (YOLO) [6] was first produced by Joseph Redmon et al. in 2016. Since then, the YOLO model family has been developed and widely used for

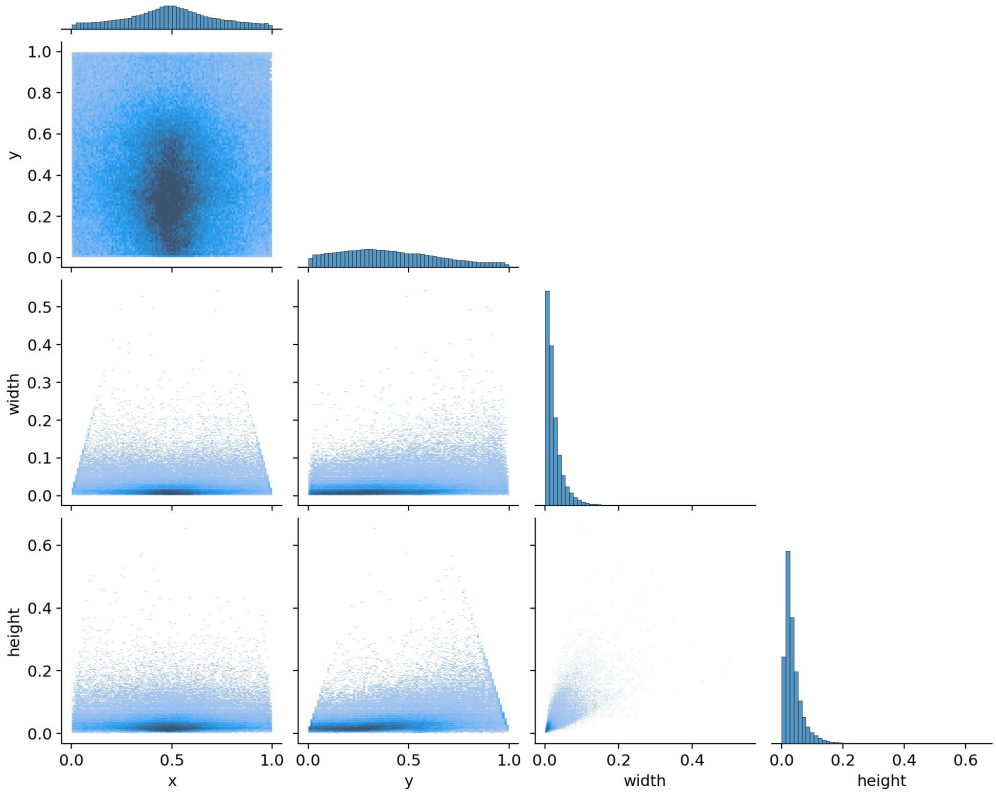


Figure 5: Labels correlogram in VisDrone-DET2019

computer vision problems, especially object detection because it ensures both accuracy and speed. Besides, YOLO is open-source, allowing for easier access, modification, and customization for specific needs. YOLOv5 is one of the typical representatives for the small object detection problem that has been researched and improved by many authors and has produced quite positive results. For those reasons, we decided to choose the YOLOv5 architecture as the baseline model to improve and evaluate our model on the VisDrone2019-DET benchmark. In this part, we review previous methods that improved YOLOv5 architecture in small object detection tasks. **Backbone**: The default backbone used in YOLOv5 is CSPDarknet53 that already applied in YOLOv4 [14]. C3 modules are used for further feature extraction and combine convolutional layers with route connections. SPPF module improves the backbone's ability to capture features at different scales. **Neck**: The role of the

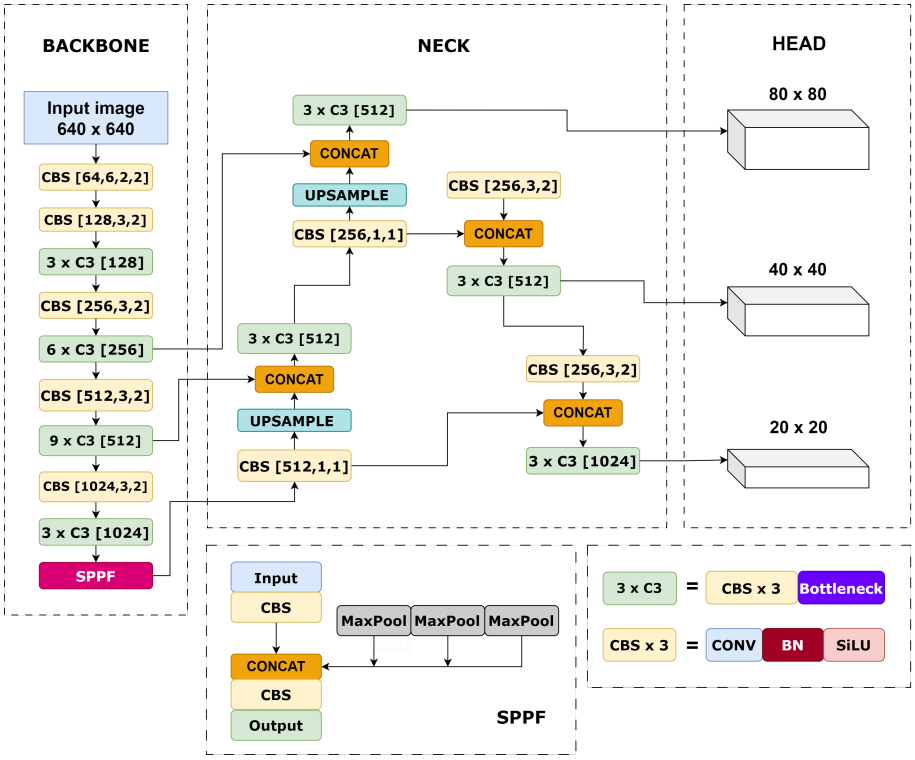


Figure 6: YOLOv5 architecture

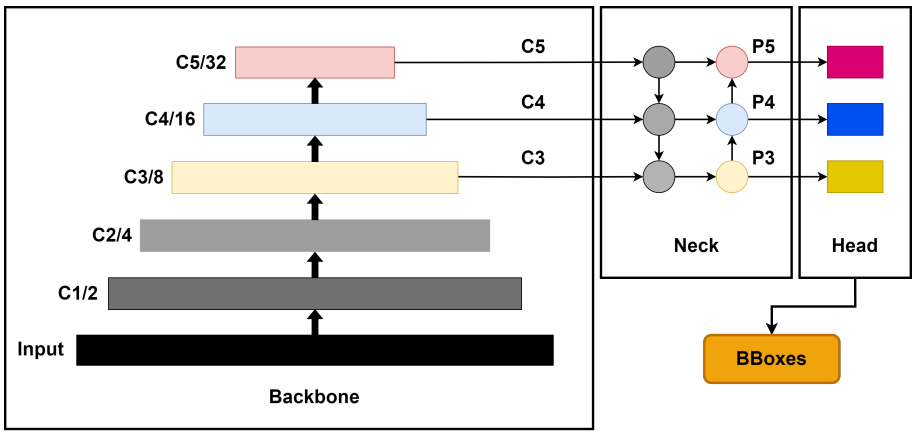


Figure 7: FPN and PANet structure in YOLOv5

neck in YOLO architecture is to further refine the features extracted by the backbone and prepare them for object detection. This often involves additional layers or modules that help enhance spatial information, capture contextual relationships, and improve the representation of objects within the feature maps. Feature Pyramid Networks (FPN) [15] a top-down pathway and lateral connections to the backbone feature maps to create a pyramid of features at different scales. This helps in detecting objects of various sizes. Aggregation Network (PANet) [16] builds upon FPN by

introducing a more advanced fusion mechanism to aggregate multi-scale features, enhancing object detection performance further. To mitigate information loss in the early stages of Feature Pyramid Networks (FPNs), the HIC-YOLOv5 [17] model, introduced by Duo Li et al., incorporates Involution Blocks [1] between the backbone and neck. It also facilitates improved feature representation and information sharing within the network. TPH-YOLOv5 [18] integrated the Convolutional Block Attention Module (CBAM) [2] into architecture to leverage both spatial and channel-wise attention mechanisms to refine spatial information within feature maps. This refinement aims to improve the model's ability to accurately localize objects in the image. **Head**: HIC-YOLOv5, TPH-YOLOv5, DSD-YOLOv5 [17, 18, 19] and many other YOLO improved architectures all add more 1 prediction head with size 160 x 160, compared to the original YOLOv5 which ensembled three prediction heads with sizes (80×80, 40×40, 20×20) of large, medium and small objects respectively, increasing the ability to correctly predict tiny objects because high-resolution feature map is more sensitive to tiny objects.

3 PROJECT MANAGEMENT

This project aims to develop an artificial intelligence (AI) model capable of accurately identifying small objects captured by unmanned aerial vehicles (UAVs). The primary objective is to automate the process of analyzing UAV imagery for the presence and location of these objects, improving accuracy compared to traditional object recognition models. Target Metrics:

- Quality: The model's success will be measured by its mean Average Precision (mAP) score. mAP is a widely used metric in object detection tasks that considers both precision and recall across different object categories and various confidence thresholds. We will target an mAP score of above 34%.
- Time: Project completion is targeted within 14 weeks. This timeline includes data collection, research, model development, training, and testing phases.
- Cost: To minimize project expenses, we will leverage free cloud resources offered by AI training platforms during the research, development, and testing phases. This approach will significantly reduce the budget allocated to computational resources.

The supporting information can be downloaded by the table 1:

Items	Link
Visdrone2019-DET Dataset	https://github.com/VisDrone/VisDrone-Dataset
Source code	https://github.com/lieubachthanh/AIP490-Small-Object-Detection-SP24AI05

Table 1: Source code and data

Task name	Priority	Owner	Owner	End date	Status	Issues
Review and analyze public dataset	High	Thanh, Lam, Khang	02/01/2024	07/01/2024	Completed	unbalanced data
Find documents and Review papers	High	Thanh, Lam, Khang	08/01/2024	14/01/2024	Completed	The problem has high difficulty, the accuracy for the problem is still low
Experiment previous research	Medium	Thanh, Lam, Khang	15/01/2024	21/01/2024	Completed	Did not reproduce the published results of previous studies
model development	High	Thanh, Lam, Khang	22/01/2024	25/02/2024	Completed	...
Experiment and error correction	Medium	Lam, Khang	26/2/2024	17/03/2024	Completed	Limited training resources
Compare results	High	Thanh, Lam, Khang	18/03/2024	24/03/2024	Completed	...
Develop demo prototypes	Medium	Thanh	25/03/2024	31/03/2024	Completed	...
Writing appendix	High	Thanh, Lam, Khang	25/03/2024	14/04/2024	Completed	...
Future work	low	Thanh, Lam, Khang	14/04/2024	...	Pending	...

Table 2: Project plan

4 MATERIALS AND METHODS

4.1 Materials

4.1.1 Dataset

The VisDrone2019-DET dataset, developed by Tianjin University, comprises aerial imagery acquired via unmanned aerial vehicles (UAVs). It encompasses a total of 10,209 images, divided into 6471 for training, 548 for validation, and 3190 for testing purposes. This dataset includes annotations for ten object categories: pedestrians, people, buses, cars, vans, trucks, bicycles, tricycles with sunshades, motorcycles, and tricycles. Analysis of category distribution, spatial distribution, and object sizes within the dataset indicates a prevalence of small objects and dense object arrangements. A representative selection of images from the dataset is presented in Figure 9.

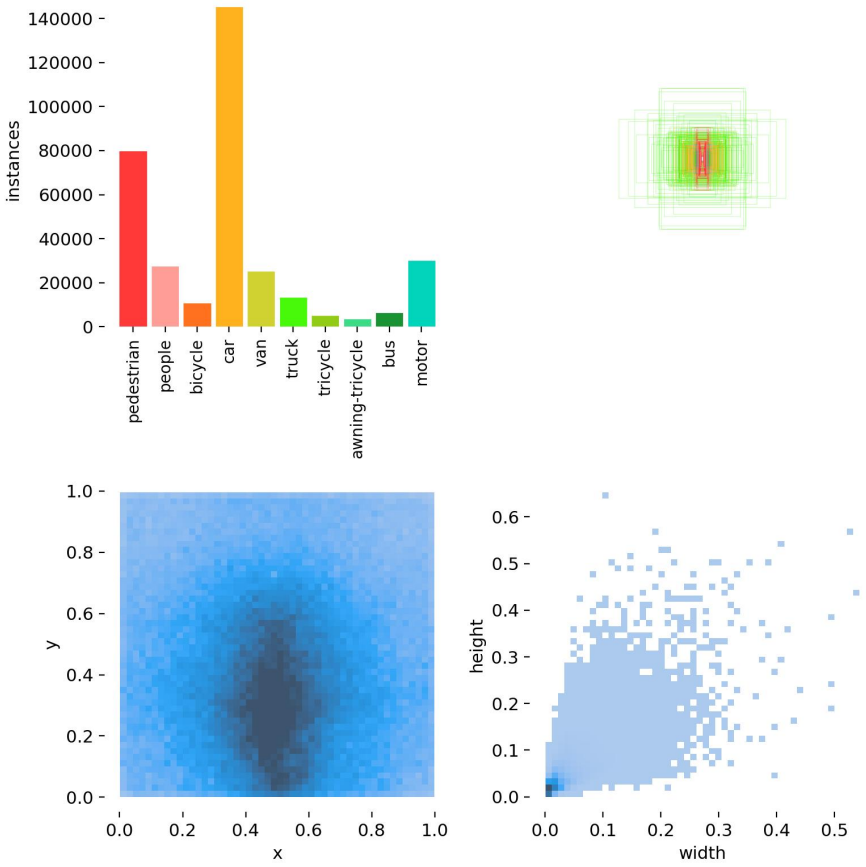


Figure 8: VisDrone-DET2019 dataset label information

Figure 9: Partial dataset showcase

4.1.2 Framework and libraries

PyTorch - Initially developed by Meta AI and now a component of the Linux Foundation, stands as a pivotal machine learning framework rooted in the Torch library. It finds extensive utility across domains such as computer vision and natural language processing. Distributed under a modified BSD license, PyTorch is an open-source platform accessible to all. Although featuring a C++ interface, its Python counterpart remains the focal point of development, boasting refinement and versatility. Leveraging PyTorch, developers can effortlessly construct intricate neural networks, courtesy of its primary data structure, Tensor, akin to Numpy arrays. Owing to its adaptability, rapidity, and user-friendly interface, PyTorch is witnessing a surge in popularity, both within the commercial sphere and among academic circles. Recognized as one of the premier tools for deep learning, PyTorch's ascent as a leading machine learning framework can be attributed to its widespread availability, expedited model development, swift training iterations, support for high-performance GPU training, and a robust ecosystem.

4.1.3 Hardware

Cloud Service - Free GPU-enabled cloud platforms like Google Colab and Kaggle serve as invaluable resources for researchers and developers, albeit with certain lim-

itations that impact workflow efficiency. Google Colab, in its regular iteration, imposes restrictions on GPU utilization, limiting the training period for certain modules like the Retriever to 4-5 hours. The Colab Pro version, on the other hand, provides access to either T4 or P100 GPUs, albeit randomly allocated, significantly extending the compute capabilities. Tasks such as pretraining masked language models can span nearly 24 hours of continuous training. However, an inherent drawback of Google Colab is the session interruption after 24 hours, potentially leading to data loss if not backed up timely. Kaggle notebooks, while also supporting GPU usage akin to Colab, present challenges in file interaction for code editing. Moreover, Kaggle imposes periodic checks for user activity every 3 to 4 hours, necessitating constant interaction to avoid interruptions, unlike the uninterrupted Colab Pro version. Furthermore, in the event of disconnection, Google Colab retains data upon reconnection, whereas Kaggle does not. Additionally, Kaggle imposes a GPU usage limit of 30 hours per week, offering GPU options like P100 or T4, facilitating multi-GPU techniques such as distributed data, data parallel, or model parallel training.

Google Colaborator - commonly known as Google Colab, represents a free cloud-based service provided by Google, offering users the capability to execute Jupyter notebooks on virtual machines (VMs) equipped with GPUs, TPUs, and other hardware accelerators. Primarily crafted to democratize machine learning and data science, it targets students and researchers lacking access to high-end hardware or costly software licenses. A key advantage of Google Colab lies in its provision of potent hardware resources at no cost, facilitating the training of machine learning models and execution of data analysis tasks. Furthermore, it streamlines collaboration and sharing of Jupyter notebooks among users. Nevertheless, there exist certain constraints associated with Google Colab that necessitate user awareness: Limited Runtime: The free version grants users 12 hours of continuous usage per session, automatically disconnecting after 90 minutes of inactivity. This poses a risk of losing progress, especially if sessions are terminated, resulting in the loss of checkpoints and necessitating reinitialization. Limited GPU and TPU Availability: Despite access to GPUs and TPUs, their availability is finite, occasionally requiring users to wait in queues for resource allocation. Limited Storage: Colab allots a restricted storage space, approximately 68 GB, shared across a user's notebooks, prompting the need for data and model file transfers between local machines and Colab. No Guaranteed Uptime: Given its status as a free service, Google Colab lacks uptime guarantees, mandating frequent saving and contingency plans in case

of service interruptions. Security Concerns: As a shared environment, users should remain vigilant regarding potential accessibility of their code and data by other users sharing the same VM. Notably, the free version provides access to 1x T4 GPU, while the pro version offers the same with 24-hour runtime, and the pro plus version introduces 1x GPU A100, albeit with limited usage before reverting to NVIDIA T4. Despite these constraints, Google Colab stands as a formidable and accessible tool for data scientists and machine learning practitioners, facilitating hardware resource access and collaborative Jupyter notebook usage without imposing financial burdens associated with high-end hardware or software licenses.

Kaggle - Kaggle stands as an online hub fostering a vibrant community of data science professionals, offering avenues for collaboration, idea exchange, and participation in machine learning contests. Its array of features encompasses datasets, kernels, discussions, and competitions, empowering its users. Among these features, Kaggle Notebooks emerge as a standout, facilitating direct execution of Jupyter notebooks within the Kaggle ecosystem. This feature boasts several advantages: Easy Collaboration: Users seamlessly share notebooks, fostering collaborative knowledge dissemination within the Kaggle community. Free Cloud-Based Computing: Kaggle furnishes users with complimentary cloud-based resources, including GPUs and TPUs, facilitating the training of intricate machine learning models. Pre-Installed Libraries: Kaggle Notebooks come equipped with essential libraries like Pandas, Numpy, and Scikit-learn, obviating the need for manual installation and expediting project initiation. Version Control: Built-in version control empowers users to monitor and revert code modifications effortlessly, ensuring project integrity. Reproducibility: Kaggle Notebooks streamline result reproducibility by enabling easy sharing of code and data. Nonetheless, Kaggle Notebooks present certain limitations: Limited Resources: Despite the provision of free cloud resources, competition for access may arise during peak usage periods due to resource constraints. No Persistent Storage: The absence of persistent storage necessitates data and model transfers between local machines and Kaggle, adding overhead. Security Concerns: As a shared environment, users must exercise caution to safeguard their data and code. In essence, Kaggle Notebooks emerge as a potent tool for data science practitioners seeking collaborative avenues, featuring complimentary cloud resources, pre-installed libraries, and version control. Nonetheless, users must navigate the platform's limitations, including resource constraints and the need for diligent data and code management.

4.1.4 Project Management tool

Notion - developed by Notion Labs Inc., stands as a freemium productivity and note-taking solution, offering an array of administrative features like bookmarking, task management, and project monitoring. Available across desktop and mobile platforms including Windows, macOS, Android, and iOS, it extends offline functionality for enhanced usability. Users can create custom templates, embed multimedia content, and collaborate in real time. Notion integrates Kanban boards, task lists, wikis, and databases within its collaborative framework, supporting modified Markdown for flexible content creation. Serving as a comprehensive workspace for note-taking, information management, and project coordination, it enables users to engage in discussions, receive feedback, and manage ongoing projects seamlessly. Accessible across platforms and web browsers, Notion incorporates a content "clipping" tool for efficient organization and task scheduling. With LaTeX support, users can effortlessly write equations in block or inline formats. Notion's user-friendly interface requires no specialized training, while its AI functionality empowers users to generate and update content, summarize notes, conduct daily standups, and perform language translation and tone checks. The platform offers a library of free and premium templates and boasts robust security features including single sign-on via Security Assertion Markup Language and private team areas for Business and Enterprise tiers. Additionally, Notion seamlessly integrates with a range of SaaS tools including GitHub, GitLab, Zoom, Lucid Software, Cisco Webex, and Typeform, further enhancing its versatility and usability.

4.2 Methods

4.2.1 Ours YOLO model

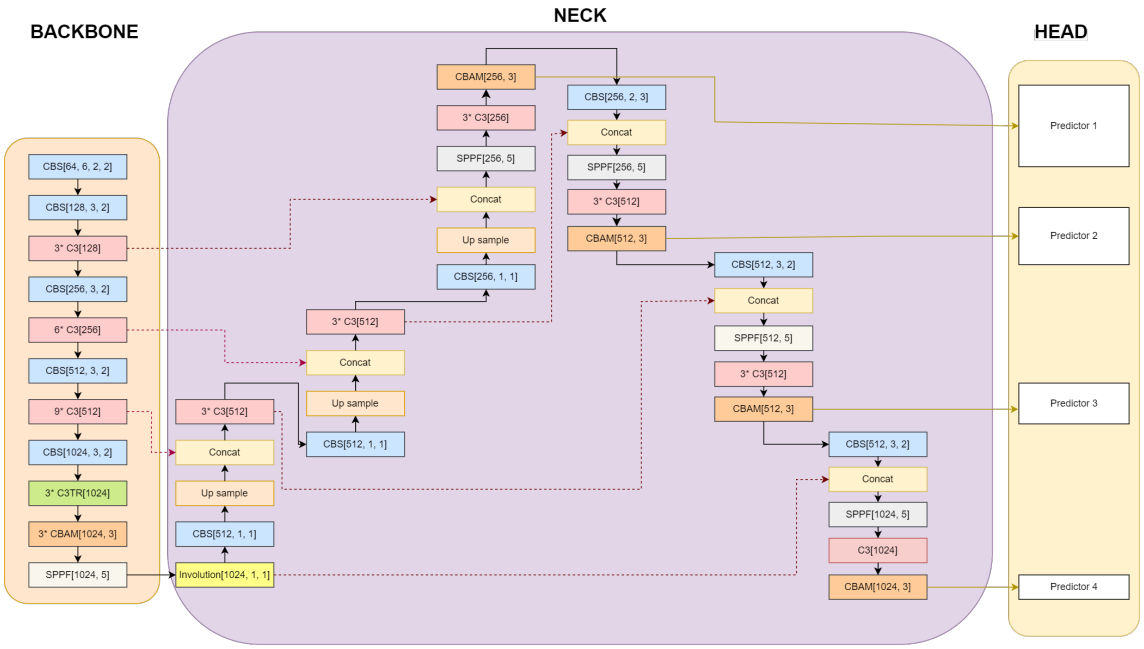


Figure 10: Architecture of ours YOLO model

- We add another prediction head to deal with small sized objects with adjustment to the number of channels less than other prediction heads.
- We integrate the C3TR to help the model pay attention to small objects that might be overlooked by traditional convolution operations, which is beneficial for detecting objects of various scales and complexities.
- The Involution block is placed between the backbone and neck, aiming to amplify the channel information within the feature map.
- The CBAM[2] is applied that recalibrates feature maps using both channel and spatial attention. The channel attention focuses on "what" features are important, while the spatial attention focuses on "where" they are important.

4.2.2 Channel Feature Fusion with Involution (CFFI)

In object detection networks, the utilization of the **Path Aggregation Network (PANet)** [16] has emerged as a prominent technique for effectively pooling features across different pyramid levels and integrating features of various scales. Notably,

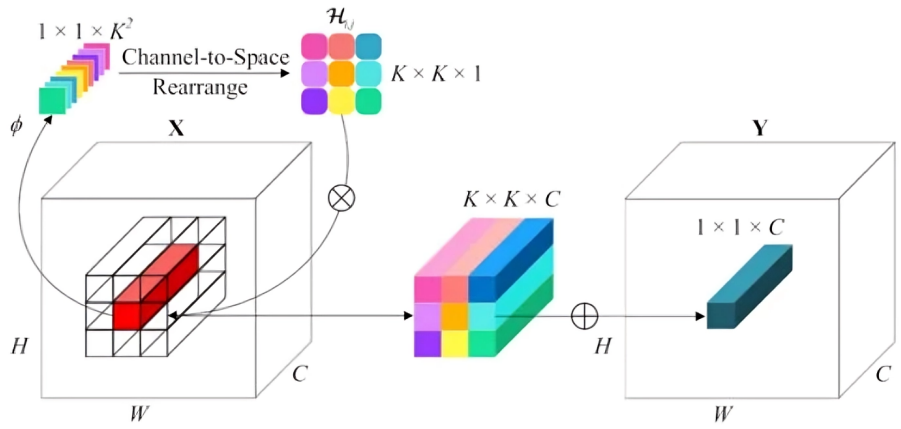


Figure 11: Schematic illustration of involution[21]

in architectures like YOLOv5, PANet plays a crucial role in enhancing the overall performance of object detection systems. Within the PANet framework, the integration of high-level feature maps from the backbone, which contain abundant semantic information, is imperative. To facilitate seamless integration with the backbone's feature mapping, researchers have employed 1 × 1 convolution layers. These layers serve to mitigate the computational burden by reducing the channel number of the features. However, it is acknowledged that this reduction in channel number may inadvertently lead to a loss of significant information, as highlighted in recent research [20]. To address the challenge of information loss stemming from the reduction in channel numbers, an Involution block has been strategically integrated between the backbone and the neck. This addition aims to address the challenge of information loss during the initial stages of FPN by enhancing and sharing channel information more effectively. As a consequence, this refinement significantly mitigates the loss of crucial details, particularly beneficial for detecting objects of smaller sizes. Moreover, the incorporation of the Involution block underscores its superior adaptability to a diverse array of visual patterns across different spatial positions, further augmenting the network's robustness and efficacy in object detection tasks [21].

The structure of Involution is illustrated in Figure 11. The involution kernel formula is $\mathcal{H}_{i,j} \in \mathbb{R}^{K \times K \times 1}$, is generated based on the function Φ at a pixel (i, j), with K representing the kernel size of involution.. In involution, the size of the input feature map inherently dictates the size of the involution kernel, thereby automatically aligning the kernel size with the spatial dimensions of the input feature map. This intrinsic alignment offers a distinct advantage: the involution kernel

can dynamically distribute weights across various positions, adapting to the spatial characteristics of the input feature map.

the CFFI functioned as a bridge between the backbone and neck, which enhanced the representation ability of the feature pyramid. The information encompassed within the channel dimension of an individual pixel is implicitly propagated throughout its spatial neighborhood. This dissemination of information proves valuable in acquiring enriched receptive field information across the vicinity.

4.2.3 Convolution Block Attention Module (CBAM)

CBAM module takes an intermediate feature map generated by a convolutional layer. This feature map contains information about the image across different channels. CBAM has two sequential sub-modules: Channel Attention Module and Spatial Attention Module (figure 12). The intermediate feature map adaptively refined through our module (CBAM) at every convolutional block of deep network. CBAM module were sequentially applied channel and spatial attention modules, so that each of the branches can learn 'what' and 'where' to attend in the channel and spatial axes respectively.

$$\begin{aligned} \mathbf{F}' &= \mathbf{M_c}(\mathbf{F}) \otimes \mathbf{F}, \\ \mathbf{F}'' &= \mathbf{M_s}(\mathbf{F}') \otimes \mathbf{F}', \end{aligned} \tag{1}$$

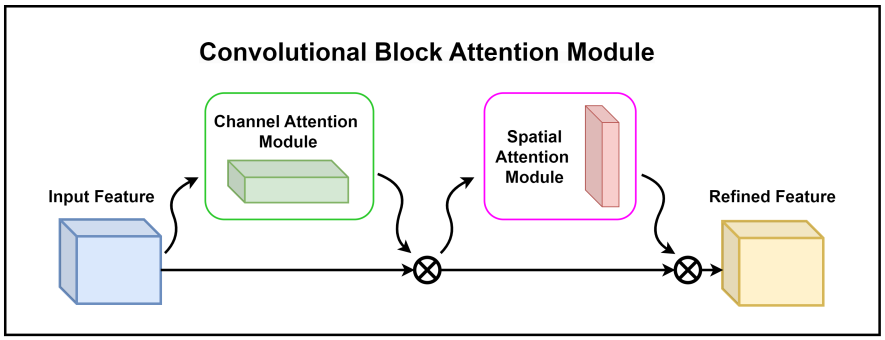


Figure 12: Convolutional Block Attention Module

Channel attention module 13: This module focuses on **"what"** information is important. It uses two techniques average pooling and max pooling. These methods compress the feature map along the spatial dimensions (height and width), retaining only the overall channel-wise information. The average pooling captures the general presence of a feature across the image, while max pooling highlights the most prominent activation in each channel. These compressed features are then

passed through a small convolutional layer to generate a channel attention map. This attention map assigns weights to each channel, indicating how important it is relative to others.

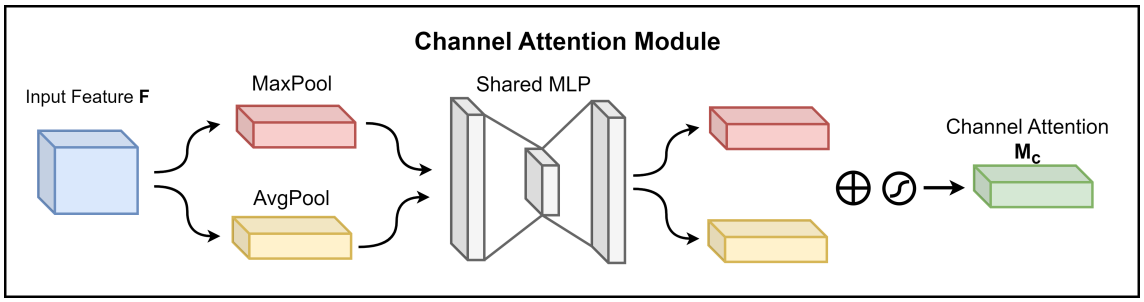


Figure 13: Channel Attention Module

$$\begin{aligned} \mathbf{M_c}(\mathbf{F}) &= \sigma(\text{MLP}(\text{AvgPool}(\mathbf{F})) + \text{MLP}(\text{MaxPool}(\mathbf{F}))) \\ &= \sigma\left(\mathbf{W_1}\left(\mathbf{W_0}\left(\mathbf{F^c_{avg}}\right)\right) + \mathbf{W_1}\left(\mathbf{W_0}\left(\mathbf{F^c_{\max}}\right)\right)\right), \end{aligned} \tag{2}$$

where σ denotes the sigmoid function, $\mathbf{W_0}$ and $\mathbf{W_1}$ are the MLP weights and are shared for inputs and ReLU activation.

Spatial attention module 14: This module focuses on **"where"** the important information is located in the image. It operates on the original feature map. Similar to CAM, it uses average pooling to capture the overall activation across channels at each spatial location. This information is then passed through a small convolutional layer to generate a spatial attention map. This attention map assigns weights to each spatial location in the feature map, highlighting areas with more relevant features.

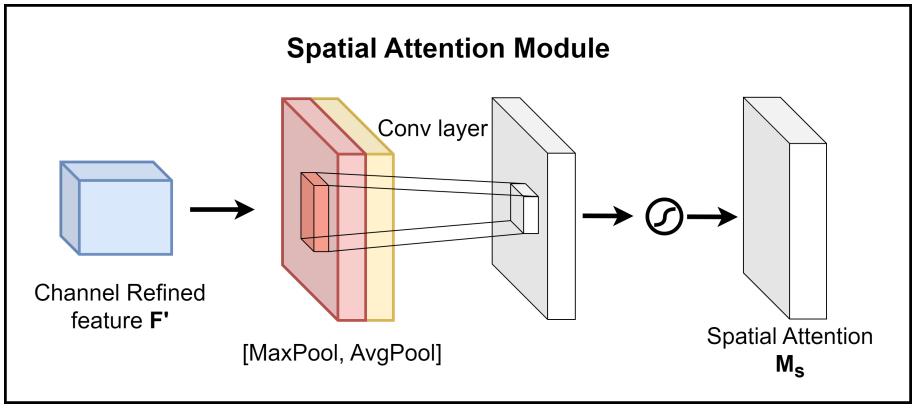


Figure 14: Spatial Attention Module

$$\begin{aligned}\mathbf{M_s}(\mathbf{F}) &= \sigma\left(f^{7\times7}([\text{AvgPool}(\mathbf{F}); \text{MaxPool}(\mathbf{F})])\right) \\ &= \sigma\left(f^{7\times7}\left(\left[\mathbf{F}^{\mathbf{s}}_{\text{avg}}; \mathbf{F}^{\mathbf{s}}_{\text{max}}\right]\right)\right),\end{aligned} \tag{3}$$

where σ denotes the sigmoid function, $f^{7\times7}$ represents for convolution operation with kernel size 7×7.

CBAM in small object detection: In UAV images, several pixel objects with confusing backgrounds seem to have difficulty catching spatial information. We take advantage of the ability to extract spatial information thanks to CBAM attention mechanism to significantly improve the ability to identify and locate objects. We have tested and evaluated the positive results in table 9.

4.2.4 C3 structure with a Transformer Block (C3TR)

The C3 structure primarily focuses on increasing the depth of the network and enhancing feature extraction capabilities. This report proposes a novel structure called C3Tr based on the C3 structure.

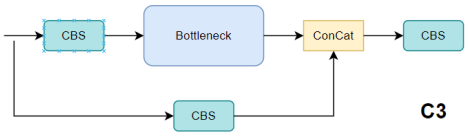


Figure 15: C3 structure in the original model's backbone network

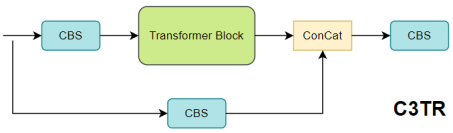


Figure 16: C3TR Structure

The C3TR block is a component used in computer vision algorithms, particularly in object detection models. It is designed to enhance the feature extraction and fusion capabilities of the model.

The C3TR block is obtained by replacing the original Bottleneck with the Transformer Block module[22] based on the C3 structure1. This modification allows the model to segment the feature map, expand the perceptual field of the feature map, and solve the problem of backpropagation gradient disappearance and gradient explosion.

In essence, the C3TR block helps to improve the model's ability to detect and recognize objects in images, particularly when the objects are of varying sizes and orientations. It is especially useful in scenarios where the objects in the images are not always aligned horizontally, such as in drone-captured images.

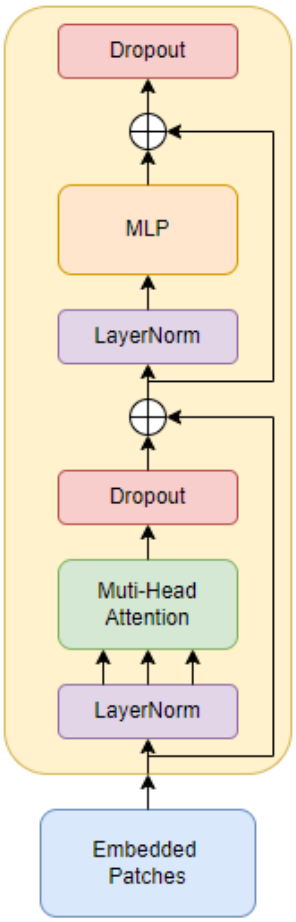


Figure 17: Transformer Encoder

C3TR module is the C3 module by replacing the bottleneck module in the C3 module to the transformer encoder block[22]. This block includes layers like Flatten, Multi-head attention, and feedforward neural network (FFN). By applying C3TR, our model has the ability to capture complex relationships within the image and multi-scale information across different locations in an image.

Flatten Operation Converts 2D feature vectors into 1D vectors while preserving positional information.

Multi-head Attention Applies different linear mappings to feature maps, allowing simultaneous attention to feature information at multiple scales. This component allows the model to focus on different parts of the input for each attention head. It helps the model to capture various aspects of the input.

Feed-Forward Neural Network (FFN) Consists of two linear transformations with a ReLU activation function in between. It is applied independently to each position in the sequence. and Dropout function between them.

Add & Norm Each sub-layer (Multi-Head Attention and FFN) in the TransformerBlock is followed by a residual connection and a layer normalization to help stabilize the learning process.

4.2.5 Soft non-maximum suppression (Soft NMS)

This report introduces Soft-NMS [23] as an alternative to the conventional NMS algorithm in the context of YOLOv5 for prediction framework screening. Unlike NMS, which selects prediction frames based on the highest confidence and conducts IoU operations sequentially, Soft-NMS introduces a penalty function to reduce the scores of overlapping prediction frames, thereby avoiding direct destruction. The application of Soft-NMS aims to address the challenges encountered when vehicle density is high and closely aligned in images captured by UAVs. In such situations, NMS tends to suppress multiple anchor frames initially associated with distinct targets, leading to false detection of occluded vehicles.

The NMS algorithm, mathematically represented by Equation (4), operates by sequentially selecting predictor boxes based on their confidence scores and conducting IoU operations with other predictor boxes. Here, b_i and s_i denote the i^{th} predictor box and its score, respectively, while N_t represents the set threshold. The candidate box with the highest score is denoted as M, and when its IoU with b_i exceeds the threshold, the score s_i of b_i is set to 0, potentially leading to erroneous removal of prediction boxes containing vehicles.

$$s_i = \begin{cases} s_i, & \text{IoU}(M, b_i) < N_t \\ 0, & \text{IoU}(M, b_i) \geq N_t \end{cases} \tag{4}$$

Conversely, Soft-NMS selects M as the benchmark box and computes IoU with neighboring predictor boxes. When the IoU is below the threshold, the adjacent prediction frame is retained; otherwise, the penalty function attenuates the scores rather than setting them to 0. Equation (5) presents the Soft-NMS algorithm, where σ represents the hyperparameter of the penalty function. This approach penalizes

prediction frames with higher IoU values, preserving those with larger overlap areas during suppression iterations and preventing the removal of targets within highly overlapping prediction frames.

$$s_i = \begin{cases} s_i, & \mathrm{IoU}\left(M, b_i\right) < N_{\mathrm{t}} \\ s_i \mathrm{e}^{-\frac{\mathrm{IoU}\left(M, b_i\right)^2}{\sigma}} & \mathrm{IoU}\left(M, b_i\right) \geq N_{\mathrm{t}} \end{cases} \tag{5}$$

Through the integration of Soft-NMS, the model demonstrates improved robustness in detecting obscured vehicles, thus contributing to the overall effectiveness of object detection in challenging environments.

4.2.6 Training Process

Loss function of our YOLO:

$$\text{Loss } = L_{cls} + L_{obj} + L_{box} \tag{6}$$

Where L_{cls}, L_{obj} and L_{box} are bounding box regression loss function, classification loss function, and confidence loss function, respectively.

Box Regression Loss: Which evaluates how accurately predicted bounding boxes match the ground truth bounding boxes. This is where CIoU [24] (The Complete Intersection over Union) comes into play. CIoU loss is a more advanced version of the IoU loss that considers the aspect ratio and the distance between the center points of the bounding boxes. The formula for CIoU loss is given by:

$$L_{box} = 1 - IoU + \frac{d^2}{c^2} + \alpha v \tag{7}$$

where

- IoU is the Intersection over Union, which measures the overlap between the predicted bounding box and the ground truth bounding box.
- d is the distance between the center points of the predicted bounding box and the ground truth bounding box.
- c is the diagonal length of the smallest enclosing box covering the two boxes.
- v is the aspect ratio consistency term that penalizes aspect ratio mismatches between the predicted bounding box and the ground truth bounding box.

- α is a trade-off parameter that balances the aspect ratio consistency term.

Object Confidence Loss: This component uses Binary Cross-Entropy (BCE) to measure the error in predicting the presence of an object within a bounding box. It helps the model learn to accurately predict the confidence score of having an object within the predicted bounding box.

$$L_{obj} = -\frac{1}{M}\sum_{j=1}^{M}[t_j \cdot \log(p_j) + (1 - t_j) \cdot \log(1 - p_j)] \tag{8}$$

where

- (M) is the number of grid cells.
- (t_j) is ground truth for the (j)-th grid cell, usually the IoU between the predicted bounding box and the actual bounding box.
- (p_j) is the predicted probability of the existence of an object in the (j)-th grid cell.

Classification Loss: Binary Cross-Entropy(BCE) also use to assess the object detector's classification performance. This metric quantifies the error between the predicted class labels and the ground truth labels associated with the proposed bounding boxes.

$$L_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}[y_{i,c} \cdot \log(p_{i,c}) + (1 - y_{i,c}) \cdot \log(1 - p_{i,c})] \tag{9}$$

where

- (N) is the number of samples.
- (C) is the number of classes.
- ($y_{i,c}$) is the actual label for class (c) of sample (i), with a value of 0 or 1.
- ($p_{i,c}$) is the predicted probability for class (c) of sample (i).

The combination of these loss components allows YOLOv5 to effectively learn to detect and classify objects in images. CIoU specifically helps in refining the accuracy of the bounding box predictions, making it a crucial part of the overall loss function.

4.3 Evaluation Metrics

After training the model, we adopt some of the main metrics used for object detection algorithms, which include:

Recall The proportion of true positive detections (correctly identified objects) out of all actual objects in the image.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision The proportion of correctly identified objects among all detections the model made (avoiding false positives).

$$\text{Precision} = \frac{TP}{TP + FP}$$

False Positives (FP) Objects that the model incorrectly identifies as being present in the image.

False Negatives (FN) Objects that are actually present in the image but the model fails to detect.

True Positives (TP) Objects that the model correctly identifies as being present in the image.

Average Precision (AP) Measures the model's ability to both correctly classify objects and localize them accurately. It takes into account both precision (avoiding false positives) and recall (detecting all true positives).
AP is calculated by measuring the area under the Precision-Recall Curve (PRC). The PRC plots the trade-off between precision (correctly identified objects) and recall (detecting all true positives) as the classification threshold varies.

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} AP_r = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1.0\}} P_{interp}(r)$$

Where

$$P_{interp}(r) = \max_{\tilde{r}:\tilde{r}>r} p(\tilde{r})$$

Mean Average Precision (mAP) The average AP across all object categories in the dataset. Provides a more comprehensive overview of the model's performance.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

5 RESULTS

5.1 Experiment

Data	**Samples**
Training	6471
Validation	548
Test	1580

Table 3: Dataset division

Processor	Intel(R)Xeon(R)CPU@2.20GHz
Operating system	Linux
Ram	29GB
Graphics card	P100 GPU
Programming language	Python3.10.13
Deep learning libraries	PyTorch2.1.2
Deep learning toolkit	CUDA12.1, cuDDN8.9.0

Table 4: Kaggle Experimental Environment

Parameter	Value	Parameter	Value
Learning Rate	0.01	Weight Decay	0.0005
Batch Size	32	Momentum	0.937
Image Size	640 × 640	Epoch	300
Dataloader	4	Optimizer	SGD

Table 5: Training Parameters

Parameter	**Value**
HSV Hue	0.015
HSV Saturation	0.7
HSV Value	0.4
Translate	0.1
Image scale	0.5
Image flip left-right	0.5
Mosaic	1.0
Mixup	0.1

Table 6: Augmentation hyperparameters

We trained our model on the same resource conditions (Kaggle environment) until the model showed signs of overfitting on the valid set. After that, we then fine-tuned the model by testing on many sets of hyper-parameters (we already showed in table 5) to reach the model's improvement limit. The total training process is 300 epochs about 35 hours of training times with a batch size of 8, the training process was interrupted and had to take place 3 times for 100 epochs each time because of the Kaggle training time limit (12 hours/session). We trained from scratch for the first time and took advantage of the transfer learning mechanism [25] for the next training times. The results of training and validation are shown in the figures below.

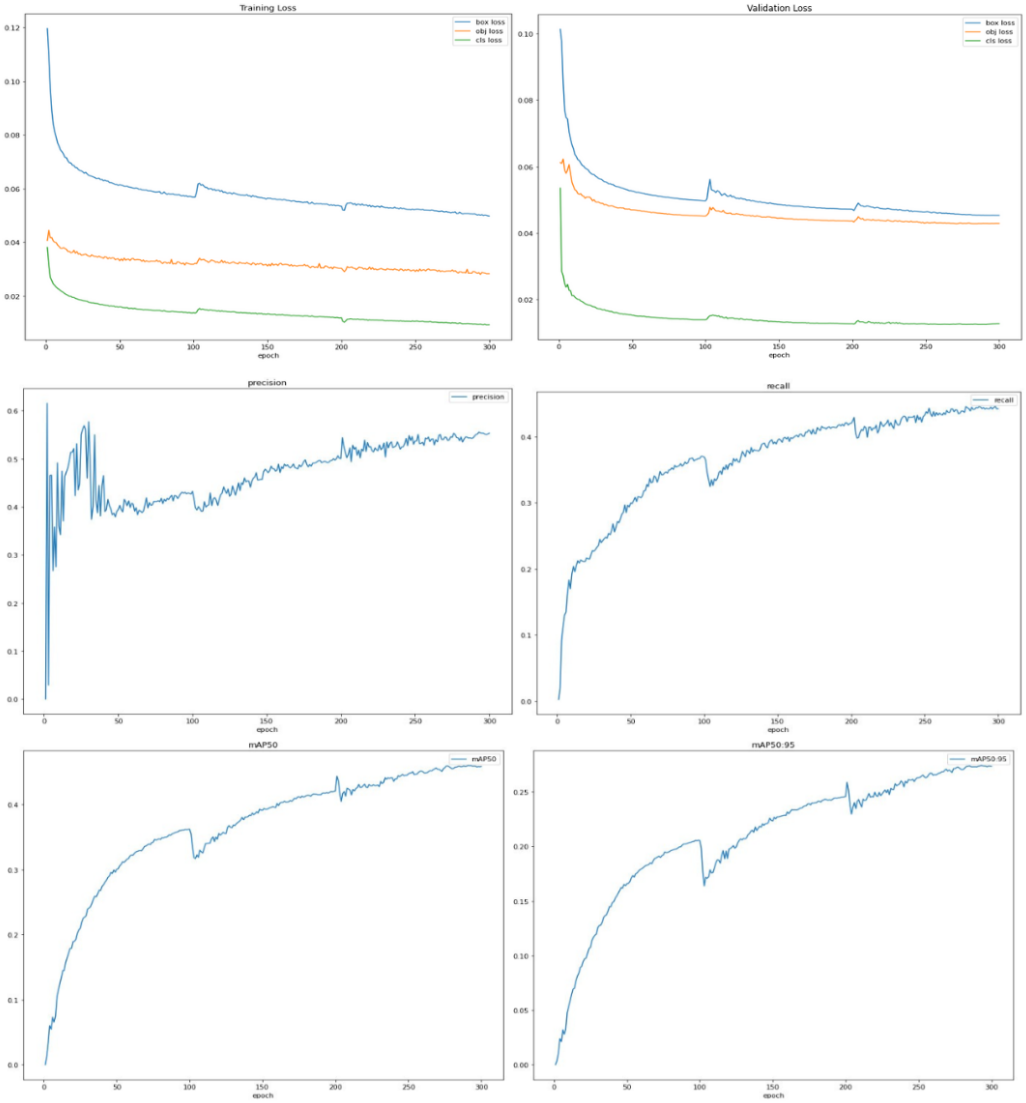


Figure 18: Training process visualization

5.2 Ablation Study

Method	Param	Precision	Recall	mAP50	mAP50:95
Baseline	7M	38.2	31.2	27.8	15.0
YOLOv5_4Head	8.3M	40.5	35.5	32.2	17.4
YOLO + CBAM	12.9M	46.0	36.4	34.5	18.9
YOLO + CBAM + Involution + C3TR	13.2M	48.0	38.9	37.4	21.2
YOLO + CBAM + Involution + C3TR + Soft-NMS	**13.2M**	50.8	37.3	**44.2**	**27.3**

Table 7: Ablation Study Table (VisDrone2019-test-dev)

We carried out a series of tests to examine the impact of four alterations: the inclusion of an extra prediction head, the implementation of an involution block, the application of CBAM, and the integration of the C3TR module.

YOLO+4P: The addition of the 4th prediction head to the YOLO model has resulted in a slight decrease in precision, but an improvement in mAP50 and mAP50:95 scores. This suggests that while the model may be making more false positive predictions, it is also able to detect more objects across different scales. Adding one more prediction head could be beneficial in scenarios where detecting as many objects as possible YOLO + CBAM: The Convolutional Block Attention Module (CBAM) focuses on enhancing the representational power of the feature maps. This allows the model to pay more attention to important features and less to irrelevant ones, improving the precision of the model. The significant improvement in all metrics compared to the YOLO+4P model suggests that the CBAM module is effective in improving the model's ability to detect objects. YOLO + CBAM + Involution: The involution block serves as a self-attention mechanism[22], allowing the model to focus on the most important features. This can be particularly beneficial in complex scenes where the objects of interest are surrounded by many distracting elements. The slight improvement in mAP50 and mAP50:95 scores suggests that the involution block enhances the model's overall object detection performance. YOLO + CBAM + Involution + C3Tr: The C3Tr module is designed to capture contextual information, which is crucial for object detection. It helps the model to understand the relationship between different objects in the scene, which can be particularly beneficial for detecting small objects that are often surrounded by larger ones. The significant improvement in the mAP50:95 score suggests that the C3Tr module effectively enhances the model's ability to detect objects across different scales. In

summary, each module brings unique benefits to the model and contributes to improving its performance in different ways. However, it's important to consider the trade-off between precision and recall when evaluating the effectiveness of these modifications.

5.3 Comparison of Different Detectors

To assess the efficacy of our enhanced algorithm, we conducted a comparative analysis with several popular single-stage detection algorithms. Each algorithm underwent training for 200 epochs, and the VisDrone2019 dataset was utilized for testing and comparison, employing mAP50 and mAP50:95 as evaluation metrics. Table 8 presents the comprehensive results of these comparisons. Our findings demonstrate that our proposed YOLO algorithm attains superior performance in aerial image detection tasks. Notably, when compared with the latest YOLOv8s detection model, our algorithm outperforms it by 12.6% in mAP50. Furthermore, compared to the original model, the enhanced version exhibits a substantial 16.8% increase in mAP50.

methods	Parameter	Precision	Recall	mAP50	mAP50:95
YOLOv5s	7M	38.2	31.2	27.8	15.0
YOLOv8s	11.2M	45.4	33.1	31.6	18.1
ourYOLO	**13.2M**	50.8	37.3	**44.2**	**27.3**

Table 8: Comparison of results on the VisDrone2019 dataset with different algorithms on VisDrone2019-DET-test-dev

While none of the methods achieved exceptionally high performance, several approaches demonstrated promising results considering their relatively low parameter counts. Notably, the our YOLO method, with 13.2M parameters, achieved an mAP50 of 50.9% and an mAP50:95 of 33.0%, which is respectable given its compact model size. compared with the DSD-YOLOv5 method, which used only 26.1M parameters, obtained mAP50 of 50.9% and mAP50:95 of 32.4%, showing the effectiveness of our YOLO method in using parameters. These findings suggest that while further improvements are needed, several algorithms can provide good trade-offs between accuracy and model complexity for small-object recognition tasks in UAV imagery.

Methods	Parameter	Precision	Recall	mAP50	mAP50:95
SSD[7]	...	...	...	10.6	5.0
RetinaNet[8]	...	...	...	21.1	12.8
Faster R-CNN[3]	...	...	...	35.6	19.6
YOLOv5s	7M	48.6	34.3	34.6	19.6
YOLOv8s	11.2M	52.1	38.9	40.1	23.9
HIC-YOLOv5[17]	9.3M	...	...	44.31	25.95
DSD-YOLOv5[19]	26.1M	...	...	50.9	32.4
Our YOLO	**13.2M**	55.0	44.6	**50.9**	**33.0**

Table 9: Comparison of results on the VisDrone2019 dataset with different algorithms on VisDrone2019-DET-val

Class	Images	Instances	Precision	Recall	mAP50	mAP50:95
All	1610	75102	50.8	37.3	44.2	27.3
Pedestrian	1610	21006	53.4	33.4	46.2	21.3
People	1610	6376	48.7	20.3	34.0	**14.0**
Bicycle	1610	1302	37.5	**12.4**	**24.1**	11.9
Car	1610	28074	70.5	**77.1**	**79.3**	**53.3**
Van	1610	5771	47.8	41.3	43.8	31.7
Truck	1610	2659	48.8	46.7	48.8	33.7
Tricycle	1610	530	**34.6**	29.2	28.2	17.3
Awning-Tricycle	1610	599	44.8	22.9	30.9	21.0
Bus	1610	2940	**72.3**	53.3	66.2	49.6
Motor	1610	5845	49.8	36.1	40.5	19.0

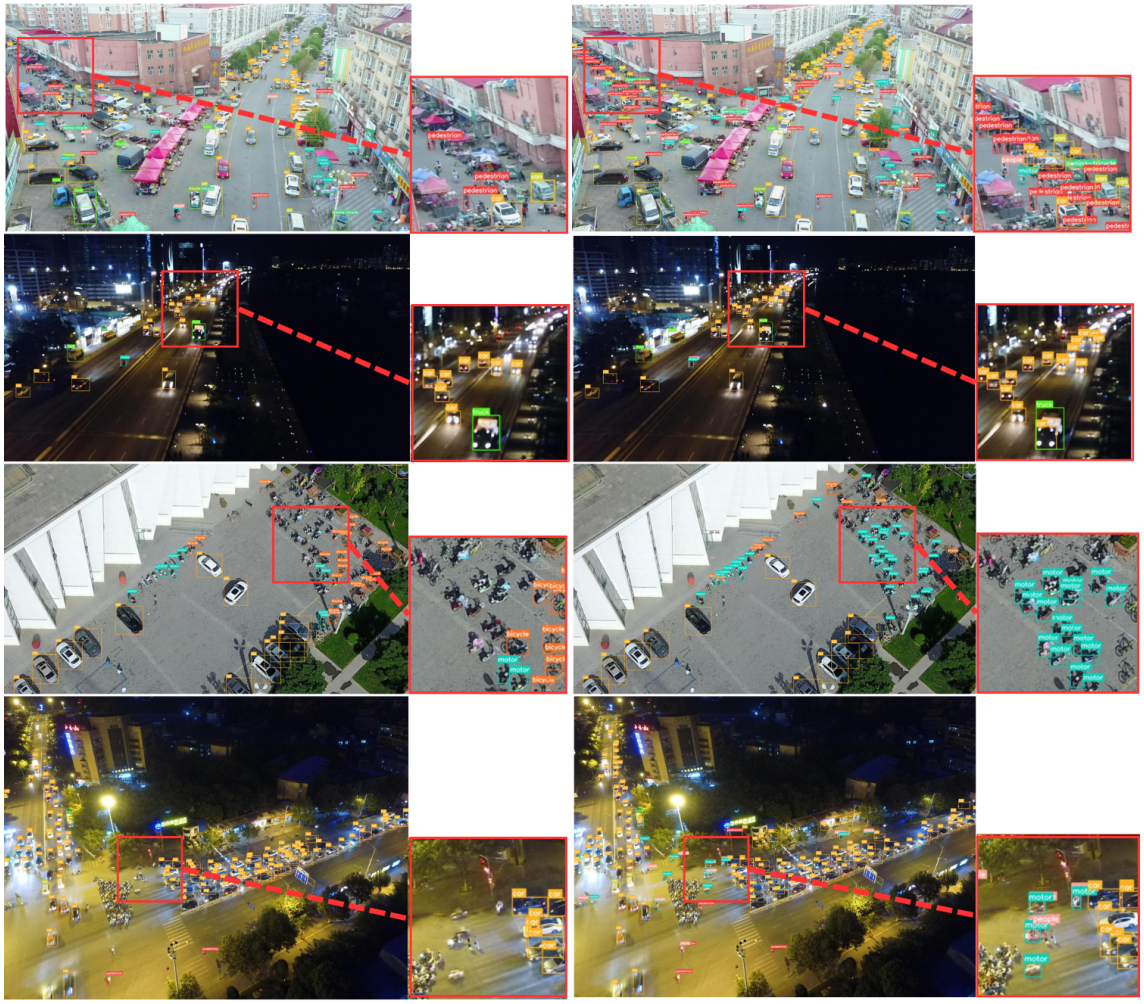
Table 10: Evaluation on each class of VisDrone2019-DET-test-dev

Based on results (Table 10) from ours model small object detection model:

- **Detection Performance**: The Car class shows the highest precision and recall, indicating strong detection capabilities. While the Tricycle class shows lower precision and recall, this is an area for potential improvement.
- **mAP Scores**: The mAP50 scores are generally high for classes with larger objects like Cars and Buses, while smaller objects like Bicycles and Tricycles have lower scores, especially in the more stringent mAP50:95 metric.
- **Data Imbalance**: There is a significant imbalance in the number of instances across classes, with Pedestrians having the highest instances. This could affect

the model's ability to learn and generalize across less-represented classes.

Overall, the model shows promising results, particularly in detecting medium and small objects, but may benefit from further training or data augmentation for classes with fewer instances and lower detection metrics. Improving the balance of data and fine-tuning the model parameters could enhance overall performance.



(a) Yolov5s results (b) Our results

Figure 19: Visualization of the detection results obtained on the VisDrone2019-DET dataset. The left side presents the results of the baseline YOLOv5. The right side shows the results of our method. The object categories are represented by different colors. To compare the detection details more conveniently, the contents of the red square boxes on the images are enlarged.

6 VISUALIZATION ANALYSIS

According to the result of confusion matrix 20, the top 3 classes with the best mAP50 (>50% mAP50) prediction results on valid set are car, pedestrian and bus. Car class has the best results (83% mAP50) because it has the largest number of objects, and the bounding box size is quite large, which is convenient for extracting features and predictions of the model. The bus class only accounts for a very modest portion of the total number of instances, but the prediction results are the second best with 55% mAP50. Awning-tricycle, tricycle and bicycle received quite bad prediction results at 14%, 24% and 20% mAP50 respectively because of the limited number of objects, only a few pixels in size.

Figure 20: Confusion matrix result

After conducting a comparative analysis results between samples on test datasets have proposed to exhibit the superior performance of the model in the identification of diminutive targets simultaneously there are a few perspectives that appear error from certain points of view.

The camera angle of drones also has some influence on the model's ability to recognize and make accurate predictions for images. Generally, images taken straight down always have superior results. When taking pictures from top to bottom, small objects in the image will have clearer shapes and positions than when taken from other angles. This makes it easier for the model to recognize small objects.

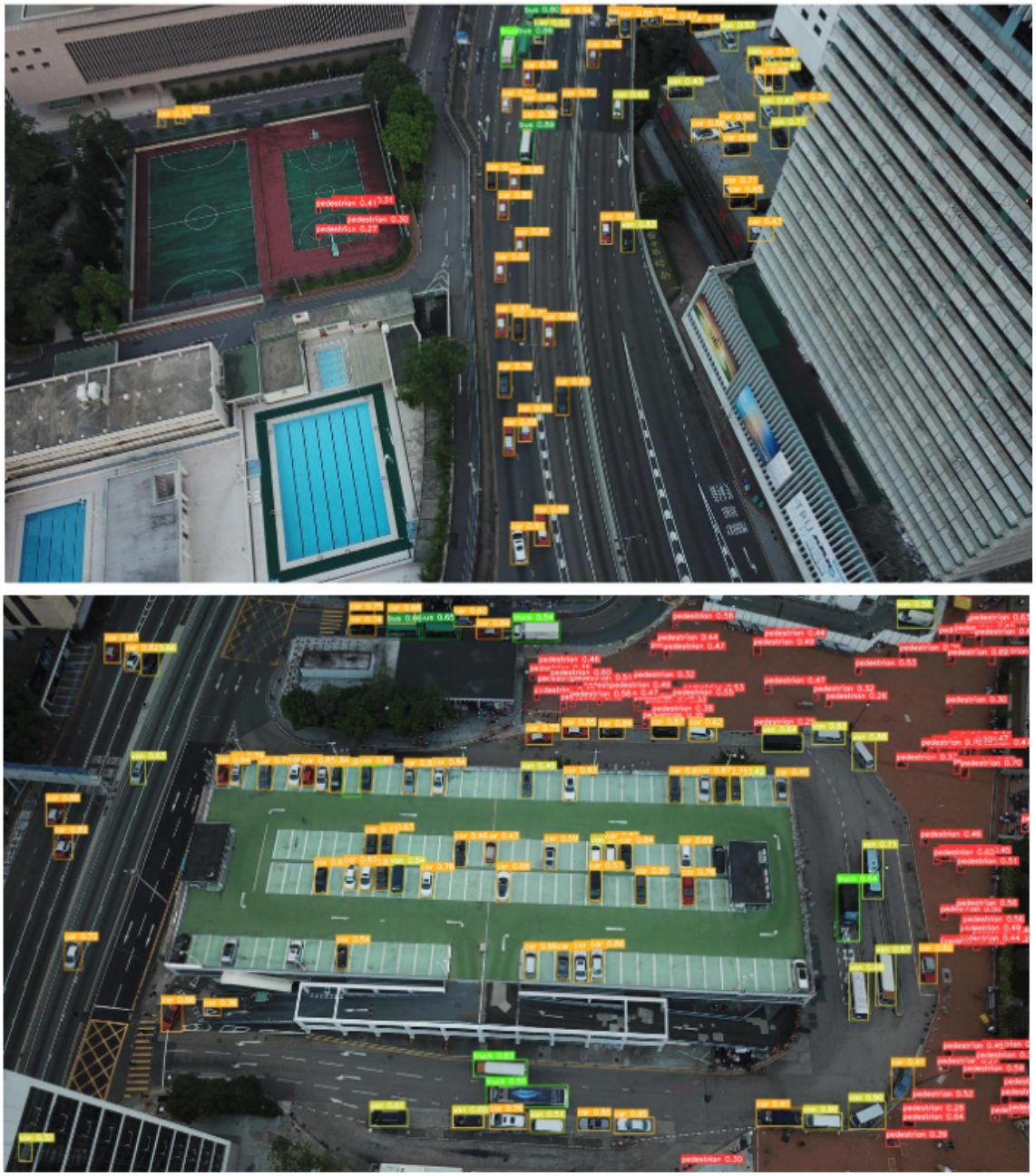


Figure 21: Image which captures straight from top to bottom

Scene Complexity leading to difficulty distinguishing individuals in an overcrowded scene. When individuals stand close together and overlaped, the model has difficulty distinguishing between them. Important information of the object may also be concealed, lowering the model's detectability.

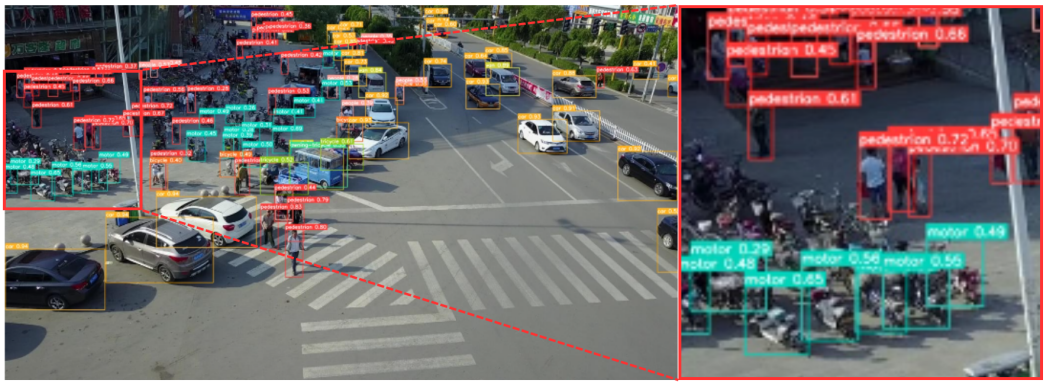


Figure 22: Individuals overlap

The given model exhibits limitations in detecting remote objects. This is primarily attributed to the lack of annotations or the complexity involved in accurately annotating these objects. As a result, these objects are often neglected during the learning process, thereby inhibiting the model's ability to learn and understand the inherent characteristics of these objects.

Figure 23: Model detection on images

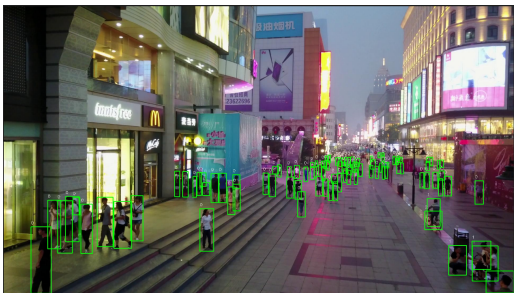


Figure 24: True Labels of images

In general, the model performs well with images that have a variety of object scales and average resolutions. Moreover, the model is capable of accurately capturing image information from top-down angles, resulting in impressive results. However, with angles that are obscured or overlapped, coupled with low-resolution images, the extraction of information becomes deficient and inaccurate, causing the model to not perform as expected. The incorporation of attention mechanisms and channel information enhancement modules enables the model to handle small object

detection tasks with flexibility. Additionally, various noise reduction and image enhancement techniques that contribute to the model's overall performance should also be considered to improve the model. To further improve the model's capabilities, it is crucial to augment the training dataset with challenging cases, including low-quality images and small-sized objects. This will enhance the model's robustness and generalization ability across diverse image conditions.

7 DISCUSSIONS

Following a detailed comparison among various models, we have found that our proposed model has advantages in detecting small targets. When compared to other target detection algorithms like SSD, YOLOv5, YOLOv7, and YOLOv8, our model achieved the highest 44.2% mAP50 and 27.3% mAP50:95 on VisDrone2019-DET-test-dev.

Integrating CBAM, Involution, and C3TR modules, individually or combined, significantly improves YOLOv5's detection accuracy. Combining these attention mechanisms with the C3 module proved more effective than using them separately. The CBAM module, focusing on both channel and spatial information, synergizes well with C3TR for global information extraction. Additionally, incorporating an Involution block between the backbone and neck enhances feature map channel information, leading to performance improvement.

However, there are still some limitations and areas for improvement. A significant variation in the Average Precision (AP) values across different target types is noticeable. This disparity persists even after model enhancements, and we can see that the cause for the difference in AP values between different object categories is due to imbalanced data. In general, models trained on categories with a larger number of samples tend to perform better than those with fewer samples.

Moreover, while the addition of modules can enhance the detection accuracy of target objects, it also increases the model's parameter count, which may not be suitable for the model's actual deployment.

In summary, our proposed alterations, including the CBAM, Involution, and C3TR modules, considerably improve the detection accuracy of the YOLOv5 model, especially when combined with the C3 module. These attention mechanisms effectively extract both local and global features, resulting in improved performance. However, there are still limitations that need to be addressed in future work.

A primary challenge is the disparity in AP values across different target types, likely caused by imbalanced data distribution. Future efforts will concentrate on data augmentation techniques or class-balanced loss functions to address this issue and achieve more consistent detection performance across all object categories.

Furthermore, while the addition of these modules improves accuracy, it also increases the model's size, which could pose challenges for real-world deployment on devices with limited resources. Therefore, future research will explore methods to reduce the number of parameters while maintaining model detection accuracy, such

as Pruning and Distillation, to strike a balance between model detection accuracy and the number of parameters.

8 CONCLUSIONS AND PERSPECTIVES

In conclusion, this report proposes several enhancements to the YOLOv5 object detection model to improve its performance in detecting small objects in drone imagery. The key modifications include:

Adding a prediction head to better handle the large-scale variance of objects. Integrating a Channel Feature Fusion with Involution (CFFI) block between the backbone and neck to reduce information loss when combining multi-scale features. Applying a Convolutional Block Attention Module (CBAM) after the C3 block to focus on the most important spatial and channel-wise features. Using a C3 structure with a Transformer block (C3TR) at the end of the backbone to better capture contextual information. Employing Soft Non-Maximum Suppression to improve bounding box scoring for densely packed objects. Extensive experiments on the VisDrone2019 dataset demonstrate the effectiveness of these modifications. The enhanced model achieves state-of-the-art performance, outperforming other single-stage detectors like YOLOv8s by a significant margin in terms of mAP50 (44.2% vs 31.6%). We attribute the performance gains to the effective integration of attention mechanisms like CBAM and involution, which help the model focus on important features for small objects. The C3TR module also enhances the utilization of contextual information for accurate object localization.

While the improved model shows superior results, some limitations remain, such as performance variations across object categories due to data imbalance. Future work could explore data augmentation techniques, class-balanced loss functions, and model compression methods to address these issues while maintaining high accuracy.

Overall, the proposed enhancements effectively boost YOLOv5's capability to detect small objects in complex drone imagery, highlighting the potential for deployment in real-world UAV applications.

Conflicts of Interest: The authors declare no conflict of interest.

9 REFERENCES

References

[1] Duo Li et al. 'Involution: Inverting the Inherence of Convolution for Visual Recognition'. In: June 2021, pp. 12316–12325. DOI: 10.1109/CVPR46437.2021.01214.

[2] Sanghyun Woo et al. 'CBAM: Convolutional Block Attention Module'. In: *CoRR* abs/1807.06521 (2018). arXiv: 1807.06521. URL: http://arxiv.org/abs/1807.06521.

[3] Ross B. Girshick et al. 'Rich feature hierarchies for accurate object detection and semantic segmentation'. In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[4] Ross B. Girshick. 'Fast R-CNN'. In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[5] Shaoqing Ren et al. 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[6] Joseph Redmon et al. 'You Only Look Once: Unified, Real-Time Object Detection'. In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[7] Wei Liu et al. 'SSD: Single Shot MultiBox Detector'. In: *CoRR* abs/1512.02325 (2015). arXiv: 1512.02325. URL: http://arxiv.org/abs/1512.02325.

[8] Tsung-Yi Lin et al. 'Focal Loss for Dense Object Detection'. In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[9] Dawei Du et al. 'VisDrone-DET2019: The Vision Meets Drone Object Detection in Image Challenge Results'. In: Oct. 2019. DOI: 10.1109/ICCVW.2019.00030.

[10] Ethan Harris et al. 'Understanding and Enhancing Mixed Sample Data Augmentation'. In: *CoRR* abs/2002.12047 (2020). arXiv: 2002.12047. URL: https://arxiv.org/abs/2002.12047.

[11] Fardad Dadboud et al. 'Single-Stage UAV Detection and Classification with YOLOV5: Mosaic Data Augmentation and PANet'. In: Nov. 2021, pp. 1–8. DOI: 10.1109/AVSS52988.2021.9663841.

[12] Quan Zhou et al. *DPNet: Dual-Path Network for Real-time Object Detection with Lightweight Attention*. Sept. 2022.

[13] Zhaowei Cai & Nuno Vasconcelos. *Cascade R-CNN: Delving into High Quality Object Detection*. 2017. arXiv: 1712.00726 [cs.CV].

[14] Alexey Bochkovskiy, Chien-Yao Wang & Hong-Yuan Mark Liao. 'YOLOv4: Optimal Speed and Accuracy of Object Detection'. In: *CoRR* abs/2004.10934 (2020). arXiv: 2004.10934. URL: https://arxiv.org/abs/2004.10934.

[15] Tsung-Yi Lin et al. 'Feature Pyramid Networks for Object Detection'. In: *CoRR* abs/1612.03144 (2016). arXiv: 1612.03144. URL: http://arxiv.org/abs/1612.03144.

[16] Shu Liu et al. 'Path Aggregation Network for Instance Segmentation'. In: *CoRR* abs/1803.01534 (2018). arXiv: 1803.01534. URL: http://arxiv.org/abs/1803.01534.

[17] Shiyi Tang, Shu Zhang & Yini Fang. *HIC-YOLOv5: Improved YOLOv5 For Small Object Detection*. 2023. arXiv: 2309.16393 [cs.CV].

[18] Xingkui Zhu et al. 'TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object Detection on Drone-captured Scenarios'. In: *CoRR* abs/2108.11539 (2021). arXiv: 2108.11539. URL: https://arxiv.org/abs/2108.11539.

[19] Chaoyue Sun et al. 'YOLOv5s-DSD: An Improved Aerial Image Detection Algorithm Based on YOLOv5s'. In: *Sensors* 23 (Aug. 2023), p. 6905. DOI: 10.3390/s23156905.

[20] Yihao Luo et al. 'CE-FPN: enhancing channel information for object detection'. In: *Multimedia Tools and Applications* 81.21 (2022), pp. 30685–30704.

[21] Duo Li et al. *Involution: Inverting the Inherence of Convolution for Visual Recognition*. 2021. arXiv: 2103.06255 [cs.CV].

[22] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[23] Navaneeth Bodla et al. 'Improving Object Detection With One Line of Code'. In: *CoRR* abs/1704.04503 (2017). arXiv: 1704.04503. URL: http://arxiv.org/abs/1704.04503.

[24] Zhaohui Zheng et al. *Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression*. 2019. arXiv: 1911.08287 [cs.CV].

[25] Stevo Bozinovski. 'Reminder of the first paper on transfer learning in neural networks, 1976'. In: *Informatica* 44.3 (2020).