



---

## **Small object detection in Remote sensing images**

*Submitted in partial fulfilment of the requirements for  
the award of the Degree*

---

**Under the Supervision of Prof. Buddhiraju Krishna Mohan**

**By: Shyam Dakhore**

**22M0324**

Centre for studies in resources engineering - IIT Bombay

June 3, 2024

---



# Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Digital Signature  
Shyam Jankiram Dakhore  
(22m0324)  
03-Jun-24 10:41:49 PM

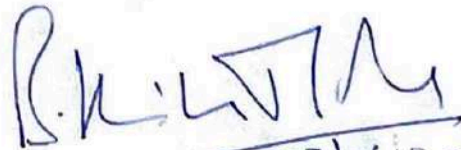
**Shyam Dakhore**

22M0324

Date: June 3, 2024

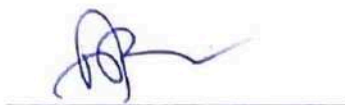
## Approval Sheet

It is certified that the work contained in this report titled **Small object detection in Remote sensing images** submitted by Mr. Shyam Dakhore (22M0324), may be accepted for evaluation.

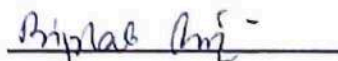
  
03/06/2024

**Prof.B.K Mohan**

Supervisor



**Chairman**



**Examiner**



**Examiner**

Date: 3<sup>rd</sup> June 2024  
Place: IIT Bombay

# Acknowledgements

First and foremost, I would like to express my deepest gratitude to my thesis supervisor, Prof. Buddhiraju Krishna Mohan, for his unwavering support, guidance, and scholarly advice. His profound knowledge, insightful suggestions, and continuous encouragement have been instrumental in shaping the direction and quality of my research work. I am truly grateful for his dedication, patience, and willingness to invest his time and efforts in reviewing and providing constructive feedback at each stage of the thesis. I would also like to extend my gratitude to the faculty members of the Centre of Studies in Resources Engineering for their valuable inputs, intellectual discussions, and academic mentorship. Their expertise and diverse perspectives have enriched my understanding of the subject matter and have been instrumental in broadening the scope of my research. I would also like to acknowledge the support and resources provided by Indian Institute of Technology, Bombay. The facilities, library resources, and access to relevant research materials have been instrumental in carrying out an in-depth study for this thesis. Once again, I express my sincere appreciation to all those who have supported me directly or indirectly in the successful completion of this M.Tech thesis. Their guidance, encouragement, and expertise have been invaluable in shaping my academic journey and research outcomes

Digital Signature  
Shyam Jankiram Dakhore  
(22m0324)  
03-Jun-24 10:42:39 PM

---

**Mr. Shyam Dakhore**

22M0324

## Abstract

In this MTP report, a comprehensive exploration of diverse object detection frameworks has been conducted, encompassing both anchor-free and anchor box-based methodologies. The investigation delves into prominent techniques such as center-point based and keypoint-based detection, as well as widely recognized approaches like YOLO and RCNN, which rely on anchor boxes for detection. The primary focus of the study centers on addressing the intricate challenge of detecting small objects within remote sensing images, a task hindered by their limited size and low contrast against complex backgrounds.

Furthermore, the research encompasses a thorough examination of various factors affecting small object detection. These factors include SAHI (Size, Aspect ratio, Height-to-width ratio, and Illumination) slicing, which significantly impacts the accuracy and reliability of detection. Additionally, the research discusses the vital need for efficient small object detection in applications ranging from surveillance to environmental monitoring and urban planning, emphasizing its importance in improving decision-making processes and augmenting overall system efficacy.

The insights gained from this extensive study are anticipated to advance the field of object detection, particularly in the challenging context of small object detection. The findings and methodologies presented herein can guide future research endeavors, aiding in the development of more precise and robust object detection systems across various domains and applications.

**Keyword:** Object Detection, YOLO, Deep Learning, Neural Network, Real-time, Accuracy, Speed, Efficiency, Architecture, Training Data, Label assignment, Optimization, Performance metrics.



# List of Figures

2.1	Network Architecture[13] . . . . .	7
2.2	This prediction encoded as an $B \times B \times (S * 5 + M)$ tensor[13] . . . . .	9
2.3	Loss Function[13] . . . . .	10
2.4	Object Detection Model[2] . . . . .	11
2.5	ELAN and E-ELAN Model[16] . . . . .	14
2.6	Slicing aided fine-tuning [1] . . . . .	17
2.7	Slicing aided hyper inference[1] . . . . .	17
2.8	Clustered object Detection (ClusDet) network [17] . . . . .	19
3.1	Visdrone Image . . . . .	25
3.2	xView Image . . . . .	28
3.3	xView Image . . . . .	29
4.1	Visdrone Image . . . . .	33
4.2	Metrics obtained during Training YOLOv5 on Visdrone Dataset . . . . .	34
5.1	Metrics obtained after Training on Visdrone Dataset . . . . .	40
5.2	Object Detection Using YOLOv8 (Pretrained Model) . . . . .	40
5.3	Object Detection Using SAHI+YOLOv8 (Pretrained Model) . . . . .	41
5.4	Object Detection Using YOLOv5 (Trained on Visdrone Dataset) . . . . .	41
5.5	Object Detection Using SAHI+YOLOv5 (Trained on Visdrone Dataset) . . . .	42
5.6	Vertical object detection a challenge . . . . .	42
5.7	Metrics obtained after Training on Xview data using yolov8x for 20 Epochs .	45
5.8	Prediction for Image 1 . . . . .	47

5.9 Prediction for Image 2 . . . . .	47
5.10 Prediction on YOLOv8x Trained on xView Dataset . . . . .	47
5.11 Confusion Matrix . . . . .	48
5.12 Confusion Matrix Normalized . . . . .	48
5.13 F1 Curve . . . . .	48
5.14 P Curve . . . . .	48
5.15 R Curve . . . . .	48
5.16 PR Curve . . . . .	48
5.17 Labels . . . . .	48
5.18 Labels Correlogram . . . . .	48
5.19 Metrics While Training . . . . .	48
5.20 Metrics While Training . . . . .	49
5.21 Prediction for Image 3 . . . . .	50
5.22 Prediction for Image 4 . . . . .	50
5.23 Prediction on YOLOv8n Trained on xView Dataset . . . . .	50
5.24 Metrics obtained after Training on Xview dataset using YOLOv8p2 for 20 epochs . . . . .	52
5.25 Prediction using YOLOv8n P2 Head . . . . .	52



# Contents

<b>Declaration</b>	<b>I</b>
<b>Acknowledgements</b>	<b>III</b>
<b>List of Figures</b>	<b>V</b>
<b>Contents</b>	<b>VII</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature review</b>	<b>2</b>
2.1 Traditional Object Detection Techniques . . . . .	3
2.2 Deep Learning based Object Detection . . . . .	4
2.3 One-stage vs Two-stage detection Approaches . . . . .	5
2.3.1 Two-Stage Object Detectors . . . . .	6
2.3.2 One-Stage Object Detectors . . . . .	6
2.4 YOLO-You Only Look Once . . . . .	6
2.4.1 Network Architecture . . . . .	8
2.4.2 Dataset used . . . . .	8
2.4.3 Working . . . . .	8
2.4.4 Limitations . . . . .	10
2.5 YOLO V4 . . . . .	11
2.5.1 Network Architecture . . . . .	11
2.5.2 Dataset Used . . . . .	12
2.5.3 Improvement over previous versions . . . . .	12

2.6	YOLO V7 . . . . .	12
2.6.1	Network Architecture . . . . .	13
2.6.2	Dataset used . . . . .	14
2.6.3	Improvement over previous versions . . . . .	14
2.7	R-CNN Family and Evolution . . . . .	15
2.7.1	Fast R-CNN . . . . .	16
2.7.2	Faster R-CNN . . . . .	16
2.8	Slicing-Enabled Enhanced Inference and Refinement . . . . .	16
2.9	Clustered Object Detection in Aerial Images . . . . .	18
2.10	Introduction to YOLO P2 . . . . .	19
2.10.1	The Evolution of YOLO . . . . .	19
2.10.2	Feature Pyramid Networks (FPN) . . . . .	19
2.10.3	YOLO and P2 Head . . . . .	20
2.10.4	Importance of P2 Head . . . . .	20
2.10.5	Conclusion . . . . .	20
<b>3</b>	<b>Problem Definition and Dataset</b>	<b>22</b>
3.1	Problem Definition . . . . .	23
3.2	Datasets . . . . .	25
3.2.1	Visdrone Dataset . . . . .	26
3.2.2	xView Dataset . . . . .	27
<b>4</b>	<b>Methodology</b>	<b>31</b>
4.1	Phase 1 . . . . .	31
4.1.1	Slicing Aided Hyper Inference . . . . .	31
4.1.2	Training YOLOv5 on the Visdrone Dataset . . . . .	32
4.1.3	Addressing Nadir View Challenges . . . . .	32
4.1.4	Summary . . . . .	33
4.2	Phase II . . . . .	34
4.2.1	Dataset Selection and Preprocessing . . . . .	35

4.2.2	Training and Fine-Tuning the Model . . . . .	35
4.2.3	Results and Performance Evaluation . . . . .	36
4.2.4	Challenges and Solutions . . . . .	36
4.2.5	Summary . . . . .	37
<b>5</b>	<b>Result and Discussion</b>	<b>39</b>
5.1	Model 1: YOLOv5 fine tuned on Visdrone Dataset . . . . .	39
5.1.1	Training Metrics . . . . .	39
5.1.2	Validation Metrics . . . . .	43
5.1.3	Summary . . . . .	44
5.2	Model 2: YOLOv8x fine tuned on xView Dataset . . . . .	44
5.2.1	Training Metrics . . . . .	44
5.2.2	Validation Metrics . . . . .	45
5.2.3	Summary . . . . .	46
5.3	Model 3: YOLOv8n fine tuned on xView Dataset Multi Class . . . . .	48
5.3.1	Training Metrics . . . . .	49
5.3.2	Validation Metrics . . . . .	50
5.3.3	Summary . . . . .	51
5.4	Model 4: YOLOv8n-P2 fine tuned on xView Dataset . . . . .	51
5.4.1	Training Metrics . . . . .	53
5.4.2	Validation Metrics . . . . .	53
5.4.3	Summary . . . . .	54
<b>6</b>	<b>Summary and Conclusion</b>	<b>55</b>
	<b>REFERENCES</b>	<b>A</b>

## CHAPTER

# 1 | Introduction

Object detection, a pivotal technology in computer vision, is designed to precisely locate and identify objects within images or videos. This process involves identifying objects, determining their locations, and correctly labeling them, contributing to diverse applications like counting items in a scene or enabling automated systems to comprehend and interact with their environment. The distinctive characteristics of each item class are leveraged by object detection methodologies, emphasizing the importance of capturing unique visual features for effective categorization.

Understanding the differentiation between object detection and other computer vision tasks is essential to avoid confusion. Image classification involves predicting the class of a single object in a picture, while object localization focuses on locating one or more items and defining bounding boxes around them. Consequently, object detection yields a more comprehensive understanding of an image compared to mere object recognition. Additionally, object segmentation, encompassing "item instance segmentation" or "semantic segmentation," extends this breakdown by highlighting precise pixels, effectively representing instances of recognized objects.

In the domain of object detection, small object detection stands as a distinct challenge, aiming to accurately detect and categorize objects that are comparatively diminutive in size within the image or video. The intricacies of this task underscore the necessity for robust visual characteristic extraction, essential for enhancing the precision and effectiveness of object detection, especially in scenarios where objects may be small or subtly defined.

## CHAPTER

# 2

## Literature review

Small object detection refers to the specialized task in object detection where the goal is to accurately detect and categorize objects that are relatively small in size within an image or video frame. These objects pose a unique challenge due to their reduced spatial dimensions, making them more challenging to discern compared to larger objects. Enhancing the detection and recognition of small objects is crucial for various applications, including surveillance, medical imaging, robotics, and micro-object analysis.

1. **Anchor-based Object Detection:** Anchor-based object detection involves predefined anchor boxes or reference boxes that are strategically placed across the image. These anchor boxes serve as templates for potential object locations and scales. The detection network then predicts adjustments to these anchor boxes to fit the actual objects in the image, helping in precise localization and classification.
2. **Non-anchor-based Object Detection:** In contrast, non-anchor-based object detection does not rely on predefined anchor boxes. Instead, it directly predicts object bounding boxes and class probabilities without using anchor box references. This approach often uses regression techniques to refine object locations and sizes directly.

## 2.1 Traditional Object Detection Techniques

Traditional object detection techniques encompass various approaches that were prevalent before the advent of deep learning and convolutional neural networks (CNNs). These methods relied on handcrafted features and classical machine learning algorithms. Here are some key traditional object detection techniques:

- **Histogram of Oriented Gradients (HOG):** Utilizes local gradient information for object representation.
- **Haar-like Features:** Based on edge, line, and four-rectangle features for object detection.
- **Cascade Classifier:** Employs a series of progressively more complex classifiers for efficient detection.
- **SIFT (Scale-Invariant Feature Transform):** Detects and describes local features invariant to scaling and rotation.
- **SURF (Speeded-Up Robust Features):** Variant of SIFT for faster keypoint detection and description.
- **Region-based CNN (R-CNN):** Combines region proposals with CNN for object detection.
- **Fast R-CNN:** Enhances R-CNN by sharing convolutional layers for faster processing.
- **Faster R-CNN:** Integrates the region proposal network into the detection network for end-to-end training.
- **YOLO (You Only Look Once):** One-stage detector predicting bounding boxes and class probabilities simultaneously.
- **SSD (Single Shot Multibox Detector):** Efficient one-stage detector using multi-scale feature maps for detection.

While these traditional techniques were foundational and effective in their time, modern object detection approaches, especially deep learning-based methods, have largely surpassed them in terms of accuracy, robustness, and versatility, and are now the primary choice for many object detection tasks.

## 2.2 Deep Learning based Object Detection

Deep learning has revolutionized the field of object detection, significantly advancing the accuracy and efficiency of object detection systems. Deep learning-based object detection methods leverage neural network architectures, particularly convolutional neural networks (CNNs), to automatically learn and extract features from images for precise object localization and classification. Here are key approaches in deep learning-based object detection:

1. **Region-based Convolutional Neural Networks (R-CNN):** R-CNN[3] was one of the pioneering deep learning-based object detection approaches. It segments the image into regions of interest and applies a CNN to each region for feature extraction. Subsequently, it uses SVMs (Support Vector Machines) for classification.
2. **Fast R-CNN:** Fast R-CNN[3] improved upon R-CNN by introducing a single CNN to process the entire image, enabling more efficient training and testing. It also incorporated a Region of Interest (RoI) pooling layer for better handling of varying region sizes.
3. **Faster R-CNN:** Faster R-CNN[14] introduced a Region Proposal Network (RPN) that shares convolutional features with the object detection network, enabling faster proposal generation and improving overall detection speed.
4. **YOLO (You Only Look Once):** YOLO[13] is an innovative approach that treats object detection as a regression problem, predicting bounding boxes and class probabilities directly in a single pass through the network. It's known for its real-time processing capabilities and has subsequent versions like YOLOv2, YOLOv3, and YOLOv4, each with improvements.

5. **SSD (Single Shot Multibox Detector):** SSD[10] combines object localization and classification into a single forward pass of the network. It employs a set of default boxes to predict multiple bounding boxes and classes for various object sizes and aspect ratios in an image.
6. **Mask R-CNN:** An extension of Faster R-CNN, Mask R-CNN[5] adds a parallel branch to predict segmentation masks for each detected object. It's used for both object detection and instance segmentation, producing highly detailed object masks.
7. **RetinaNet:** RetinaNet[9] is designed to address the class imbalance issue in object detection. It introduces the Focal Loss to give more weight to difficult, under-represented examples, leading to improved performance, especially in detecting rare objects.
8. **EfficientDet:** EfficientDet[15] focuses on achieving better accuracy and efficiency by using a compound scaling method that scales up network architecture in multiple dimensions (depth, width, resolution) while maintaining efficiency.

These deep learning-based object detection methods have significantly improved the accuracy, speed, and versatility of object detection systems. They find extensive applications in fields like autonomous driving, robotics, surveillance, medical imaging, and more, where precise object detection is critical for decision-making and analysis.

## 2.3 One-stage vs Two-stage detection Approaches

Modern object detection techniques can be broadly categorized into two main types: one-stage object detectors and two-stage object detectors. These techniques, largely based on deep learning, extract essential features from input images or video frames to perform object detection. The key objectives of an object detector involve identifying any number of objects (potentially none) in the input and classifying each object while estimating their sizes through bounding boxes.



### 2.3.1 Two-Stage Object Detectors

In the case of two-stage object detectors, bounding box regression is employed for object candidate refinement. Deep features are initially utilized to propose approximate regions in the image likely to contain objects. Subsequently, these features are leveraged for object classification. The two-stage architecture encompasses object region proposal, achieved either through deep networks or traditional computer vision techniques, followed by object categorization via bounding-box regression to gather characteristics from the proposed region. While two-stage approaches achieve maximum detection accuracy, they tend to be slower due to the multiple inference stages required for each image.

### 2.3.2 One-Stage Object Detectors

Contrastingly, one-stage object detectors do not involve a separate object region proposal phase and can estimate bounding boxes directly over the input images. This approach is faster and suitable for real-time applications. One-stage object detectors prioritize inference speed, making them extremely quick, though they may struggle with accurately distinguishing collections of small items or objects with irregular shapes. Prominent one-stage detectors include YOLO (You Only Look Once), SSD (Single Shot Multibox Detector), and RetinaNet. Recent advancements include YOLOv4-Scaled, YOLOv7, and YOLOR (2021) models. One-stage algorithms excel in speed and have simpler structural designs, allowing end-to-end training.

## 2.4 YOLO-You Only Look Once

YOLO is a real-time object detection system, it is very demanding and constantly the subject of research. Every year, a new version of YOLO is released with minor or significant changes that increase the system's speed and accuracy. So far, there have been 8 versions of YOLO, with version 8 being the most recent released on 10<sup>th</sup> January 2023.

In 2015, Redmon et al[13] introduced the first version of YOLO, known as YOLO V1.

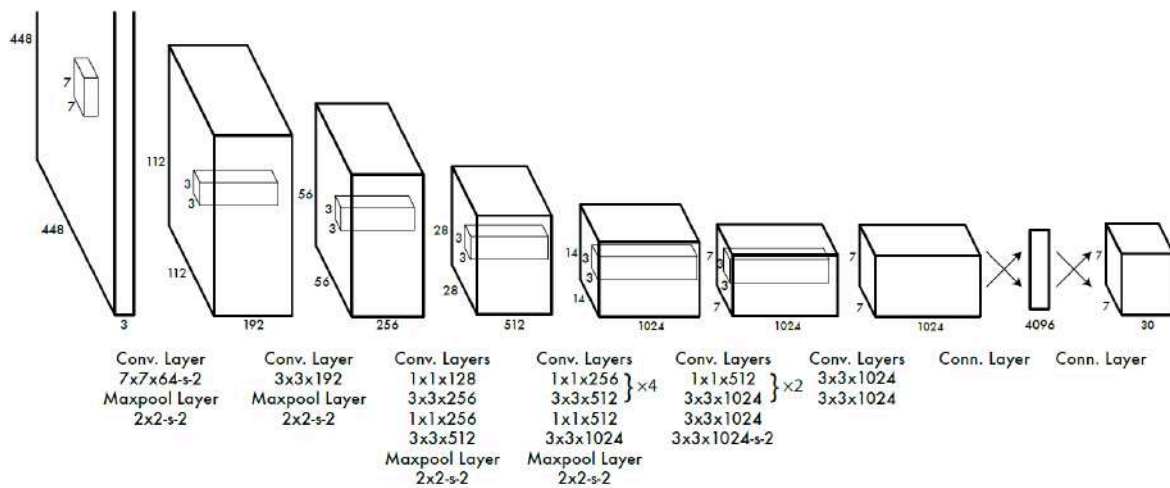


Figure 2.1: Network Architecture[13]

Prior to YOLO, object detection using convolutional neural networks (CNNs) like R-CNN relied on Region Proposal Networks (RPNs) to generate bounding box proposals, followed by running a classifier on the proposals, and finally applying post-processing to remove duplicate detections and refine the bounding boxes. The R-CNN network required separate training for each level, was slow and challenging to optimize.[13]

The driving force behind YOLO's architecture as illustrated in [Fig. 2.1] was to develop a convolutional neural network that could be trained end-to-end, was easy to optimize, and could operate in real-time. YOLO's approach to object detection involves treating it as a single regression task, where the network predicts bounding box coordinates and class probabilities directly from the image pixels.

The popularity of YOLO stems from its ability to achieve high accuracy in real-time object detection. Its efficiency lies in its "you only look once" approach, which means that it only requires a single forward propagation pass through the neural network to generate predictions. The algorithm uses non-max suppression to ensure that each object is only detected once, and then outputs the recognized objects along with their respective bounding boxes.

### 2.4.1 Network Architecture

The YOLO V1 model as illustrated in [Fig. 2.1] comprises of 24 convolutional layers and 2 fully connected layers. To reduce the feature space from previous layers, the model also includes alternating  $1 \times 1$  convolutional layers. The convolutional layers are pre-trained on the ImageNet classification task at a resolution of  $224 \times 224$  for half of the training process, after which the resolution is doubled for object detection. The network architecture of YOLO V1 is based on the GoogLeNet model, which is used for image classification.[13]

Redmon et al also developed a faster version of YOLO for speedy object detection, known as Fast YOLO. Its network architecture has fewer convolutional layers (9 instead of 24), and the number of filters is reduced in each layer. The ultimate output of the network is a  $7 \times 7 \times 30$  tensor of predictions.

### 2.4.2 Dataset used

ImageNet Data set which contains 1000 class competition data set was used for training the CNN network.

The PASCAL Visual Object Classes (VOC) dataset was used in YOLO version 1 for evaluating. This data set consists of images of various objects categorized into 20 classes, including animals, vehicles, and household objects.

PASCAL VOC 2007, 2010 and 2012 were the three versions used for model evaluation. PASCAL VOC 2007 was used to evaluate all models but to check the generalizability of the models on person detection Redmon et al used PASCAL VOC 2010 on people art model and PASCAL VOC 2012 on picasso model.

### 2.4.3 Working

The YOLO algorithm as illustrated in [Fig. 2.2] divides an image into an  $B \times B$  grid, where each grid cell becomes responsible for detecting an object if the object's center lies within that cell. Each grid cell generates predictions for  $B$  bounding boxes, and assigns a confidence score to each box, which represents the algorithm's level of confidence in identifying an object

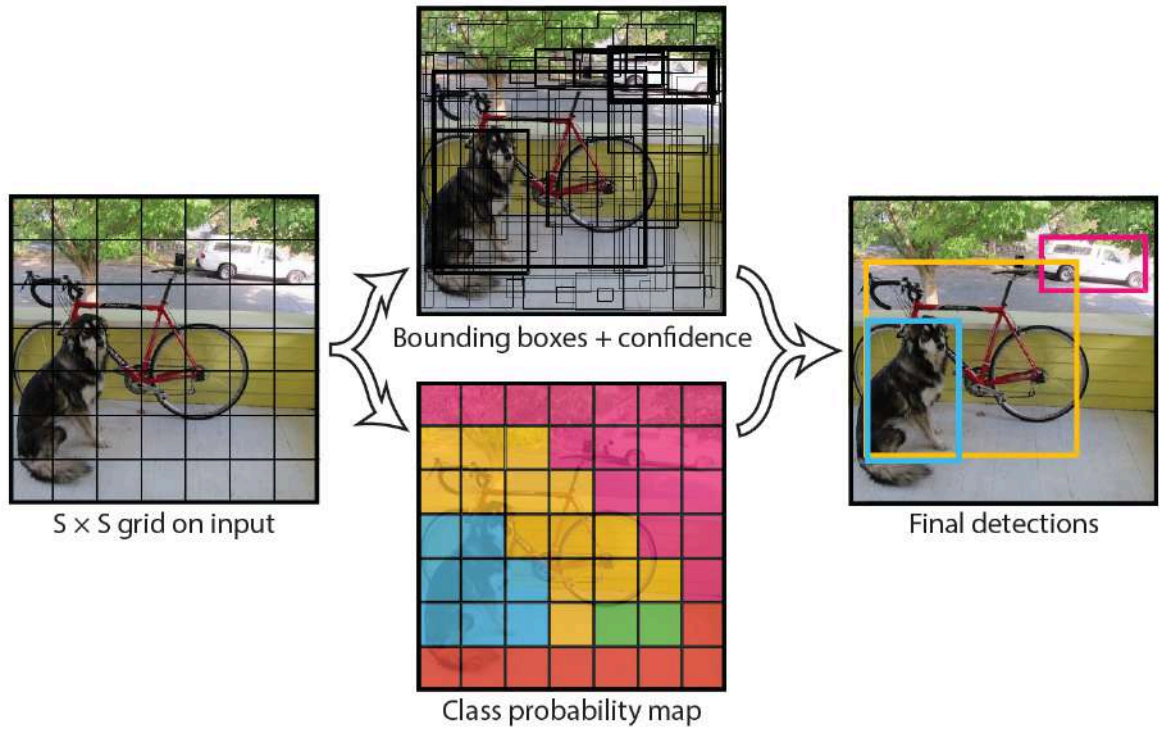


Figure 2.2: This prediction encoded as an  $B \times B \times (S * 5 + M)$  tensor [13]

within the box. Only the bounding boxes that have a confidence score greater than a preset threshold value of 0.25 are included in the final output. [13].

$$\text{Confidence score} = Pr(obj) * IOU_{truth} \quad (2.1)$$

Where-

$Pr(obj)$  - stands for probability that an object exists in the predicted bounding boxes

$IOU$  - stands for intersection over union

$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

Predicting a bounding box means predicting 5 values:  $[x_c, y_c, w_c, h_c]$  and confidence score.  $(x_c, y_c)$  are the co-ordinates of the center of the predicted box relative to its corresponding grid cell and  $(w_c, h_c)$  are the width and height of the box relative to the image.

$$\begin{aligned}
& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

Figure 2.3: Loss Function[13]

## 2.4.4 Limitations

1. YOLO's bounding box predictions are spatially limited because each grid cell can only predict two boxes and one class. This limitation makes it challenging for the model to predict adjacent items. Additionally, YOLO faces difficulty in detecting small objects that appear in clusters, such as flocks of birds.[13]
2. The model's ability to predict bounding boxes is limited by its reliance on learning from data, making it difficult to generalize to objects with unusual aspect ratios or configurations. Additionally, due to the use of multiple down-sampling layers in the model architecture, the data used for predicting bounding boxes is relatively coarse.
3. During training, YOLO uses a loss function as illustrated in [Fig. 2.3] that estimates detection performance; however, the function treats errors in small bounding boxes and large bounding boxes equally. In reality, a small error in a large box is not as significant as a small error in a small box, which has a more significant impact on Intersection over Union (IOU). As a result, the model's primary source of error is inaccurate localizations.

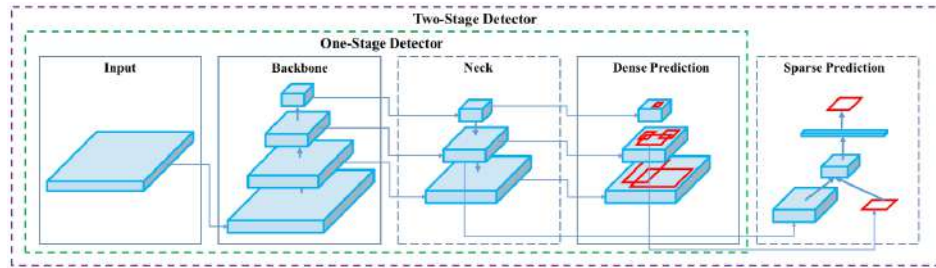


Figure 2.4: Object Detection Model[2]

## 2.5 YOLO V4

To enhance the precision and swiftness of the model, YOLOv4 utilizes numerous advanced techniques such as a spatial pyramid pooling module, a custom-designed backbone network, and the mish activation function. Data augmentation techniques and training strategies are also employed to boost the model's robustness and generalization. As a result of these efforts, YOLOv4 has achieved top-tier results on multiple benchmark datasets and is commonly implemented in various fields.

### 2.5.1 Network Architecture

As we can see in the figure a typical object detection model consist of Input layer, backbone,neck and head which includes dense prediction layer and sparse prediction layer. If the detector only includes dense predictor layer then it is called as One-Stage Detector while if it includes Dense and sparse predictor layer then it is called Two-Stage Detector

The YOLO V4 architecture as illustrated in [Fig. 2.4] comprises of CSPDarknet53 as its backbone, SPP (Spatial Pyramid Pooling) and PAN (Path Aggregation Network) modules as its neck, and YOLO v3 as the head. In addition to these layers, the model employs various "bag of freebies" techniques that alter only the training strategy or increase the training cost, and "bag of specials" techniques that consist of plugin modules and post-processing methods. Although these techniques may slightly increase the inference cost, they can significantly enhance the accuracy of object detection.[2]

"With 27.6 million parameters, CSPDarknet53 comprises 29 convolutional layers of 3 x 3

dimensions and has a receptive field of  $725 \times 725$ .”[2]

### 2.5.2 Dataset Used

To evaluate the classifier’s accuracy, we employ the ImageNet dataset, while the MS COCO dataset is utilized to gauge the detector’s precision.

### 2.5.3 Improvement over previous versions

YOLO v4 uses various bag of freebies and bag of specials to increase the accuracy and speed as follows:

The Bag of Freebies (BoF) for the backbone in YOLOv4 includes CutMix and Mosaic data augmentation, Class label smoothing, and DropBlock regularization. For the backbone’s Bag of Specials (BoS), it uses Mish activation, Cross-stage partial connections (CSP), and Multi-input weighted residual connections (MiWRC). In the detector, YOLOv4’s Bag of Freebies (BoF) incorporates CIOU-loss, CmBN, Mosaic data augmentation, DropBlock regularization, Self-Adversarial Training, eliminating grid sensitivity, using multiple anchors for a single ground truth, Cosine annealing scheduler, optimal hyperparameters, and random training shapes. Meanwhile, the Bag of Specials (BoS) for the detector includes Mish activation, SPP-block, SAM-block, PAN path-aggregation block, and DIOU-NMS.[2]

## 2.6 YOLO V7

This Technique proposes various trainable bag-of-freebies techniques that enhance the accuracy of real-time object detection without augmenting the inference cost. It introduces novel solutions to two challenges encountered in the development of object detection methods: re-parameterization and dynamic label assignment across output layers. In addition, the authors propose ”extend” and ”compound scaling” techniques for real-time object detectors, which maximize parameter and computation usage. The approach results in a 40% reduction in



parameters and a 50% decrease in computation requirements, while still achieving higher detection accuracy and faster inference than existing real-time object detection methods.[16]

### 2.6.1 Network Architecture

#### E-ELAN

Current research on designing efficient neural network structures mainly concentrates on parameters and computational density. Nonetheless, recent analysis has examined the impact of other factors, such as input/output channel ratios and gradient path. The Extended Efficient Layer Aggregation Network (Extended-ELAN) as illustrated in [Fig. 2.5] introduces a new architecture that employs expand, shuffle, and merge cardinality to improve the network's learning capabilities while retaining the gradient path. The use of group convolution expands the channel and cardinality of computational blocks, and feature maps are shuffled and concatenated into groups, which are then added together using merge cardinality. This approach enables different groups of computational blocks to learn more varied features while preserving the original ELAN architecture. By reducing parameter and computational demands while improving detection accuracy, E-ELAN presents a promising new approach to neural network design.

#### Model Scaling

Model scaling will adjust different properties of the model to create models of different sizes to accommodate different inference rates. The EfficientNet model takes width, depth, and resolution into account, while Scaled-YOLOv4 sets the level. Previous methods had analyzed the impact of common mesh and group networks on calculations and parameters while measuring width and depth in PlainNet and ResNet architectures. However, the application of this method to the stepwise model may cause changes in the process after the calculation of the blocks during the depth analysis. Therefore, it is necessary to consider different scaling factors together in a concatenation-based model. The proposed compound scaling method maintains the initial properties and optimal structure of the model when scaling depth and



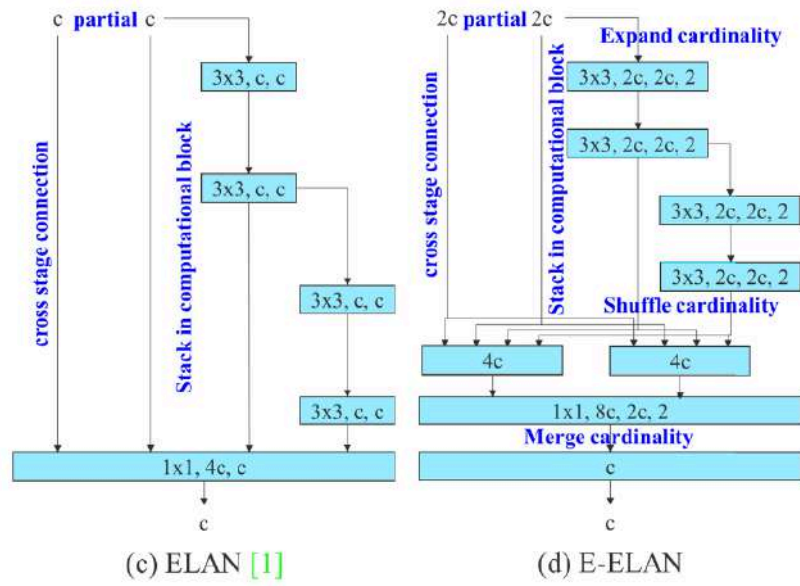


Figure 2.5: ELAN and E-ELAN Model[16]

width factors. This approach ensures a change in the output channel of a computational block while performing width factor scaling on the transition layers.

### 2.6.2 Dataset used

The training process of YOLOv7 involved using only the MS COCO dataset and did not involve any usage of pre-trained weights or other datasets.

### 2.6.3 Improvement over previous versions

#### Planned reparameterized convolution

The RepConv algorithm has shown impressive results in VGG, but it loses accuracy when applied to ResNet, DenseNet, and other models. The authors examine the integration of RepConv with various networks through the analysis of gradient flow propagation paths and introduce a modified version called RepConvN that eliminates the identity connection. They propose a planned re-parameterized convolution architecture for models with residual or concatenation connections, in which the replaced convolutional layer should not have an identity connection. The authors present an example of this architecture in PlainNet and

ResNet and plan to present a complete experiment in residual-based and concatenation-based models in the ablation study session.

### **Coarse for auxiliary and fine for lead loss**

In deep network training, the technique of deep supervision involves adding an auxiliary head to the middle layers of the network to assist in training, which has been commonly used. By incorporating an auxiliary head along with an assistant loss, the model's performance can be improved, even for well-converged architectures like ResNet and DenseNet. The approach to label assignment in deep network training has also evolved, with recent methods using the quality and distribution of the network's prediction output to generate reliable soft labels. These soft labels are then taken into account along with the ground truth to execute label assignment.

### **Other trainable bag of freebies**

This section presents a list of trainable bag-of-freebies that were used in training, but not proposed by the authors. Which includes batch normalization in conv-bn-activation topology, combining implicit knowledge in YOLOR with convolution feature map, and using EMA model as the final inference model. The purpose of these freebies is to improve the performance of the system

## **2.7 R-CNN Family and Evolution**

The R-CNN family of models revolutionized object detection through a two-stage approach that combines region proposal generation with deep Convolutional Neural Networks (CNNs). This family comprises R-CNN, Fast R-CNN, and Faster R-CNN, each improving upon its predecessor.

### 2.7.1 Fast R-CNN

Fast R-CNN addressed R-CNN's slow computation drawback by processing the entire image in a single forward pass instead of individual region proposals. It introduced a region of interest (RoI) pooling layer, aligning feature maps from region proposals with input feature maps. The extracted region-wise features are then fed into fully connected layers for classification and bounding box regression.

### 2.7.2 Faster R-CNN

Faster R-CNN enhanced efficiency by introducing an integrated region proposal network (RPN). This fully convolutional network shares convolutional layers with the object detection network, directly generating region proposals. The proposals from RPN are further processed for RoI pooling, object classification, and regression, similar to Fast R-CNN.

The core innovation of the R-CNN family lies in integrating region proposal generation and CNN-based feature extraction, facilitating precise object detection in complex scenes. This approach focuses computation on potential regions of interest, significantly improving speed and accuracy compared to traditional sliding window approaches.

Since the introduction of the R-CNN family, numerous variations and improvements have emerged, such as Mask R-CNN, extending object detection to instance segmentation by incorporating a parallel mask prediction branch. These models serve as the foundation for modern object detection frameworks, driving advancements in the field.

## 2.8 Slicing-Enabled Enhanced Inference and Refinement

Addressing the challenge of detecting small objects within high-resolution images, a general architecture based on slicing was proposed. This technique involves partitioning the input images into overlapping patches of variable sizes during both the fine-tuning and inference stages. The slicing methodology allows for the generation of larger pixel regions concerning small objects compared to the original images .

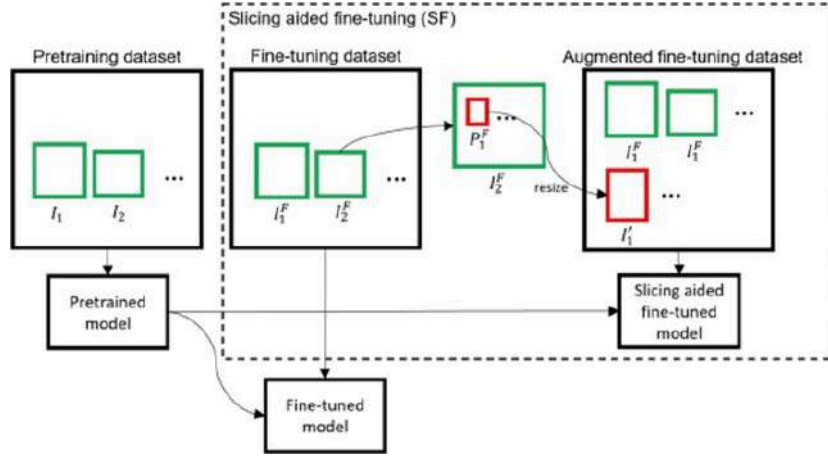


Figure 2.6: Slicing aided fine-tuning [1]

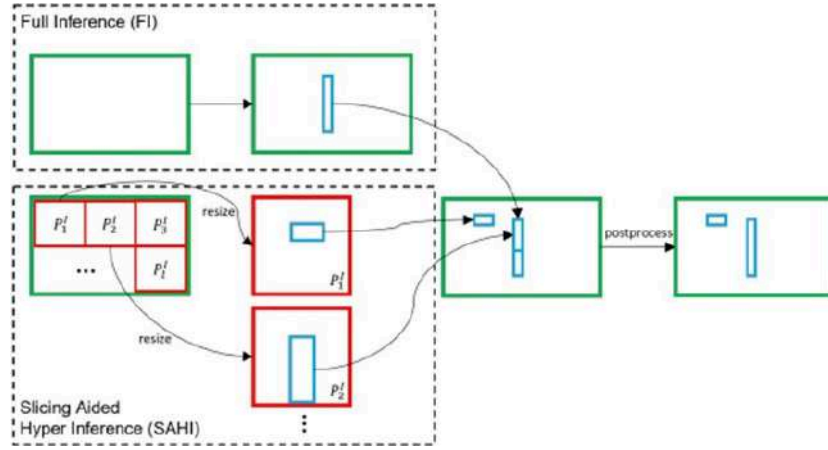


Figure 2.7: Slicing aided hyper inference[1]

During the fine-tuning phase, the patches are resized while preserving the aspect ratio, resulting in augmented images with relatively larger object sizes. These augmented images, alongside the original ones, are utilized to fine-tune the object detector.

In the inference stage, the original query image is similarly segmented into overlapping patches. Similar to the fine-tuning process, these patches are resized while maintaining the aspect ratio. Each patch undergoes its own object detection forward pass. Optionally, a full inference using the original image can be executed to detect larger objects. The prediction outcomes from overlapping patches and the full inference (if applied) are then amalgamated back into their original dimensions. Experiments conducted on diverse detectors and datasets have demonstrated that this slicing strategy can enhance the average precision (AP) by up

to 6.8%. Specifically, it offers an additional 14.5% enhancement in AP for small objects. It is essential to note that employing higher resolution images and larger feature maps during training escalates computational and memory requisites. However, the proposed slicing approach permits a linear increase in computation time while maintaining constant memory demands. When adjusting the patch sizes, careful consideration should be given to the target platform to strike a balance between computation and memory constraints.

## 2.9 Clustered Object Detection in Aerial Images

Detecting objects in aerial images poses significant challenges, primarily due to two key factors. First, target objects, such as pedestrians, are minute in size, making them difficult to distinguish from the surrounding background at a pixel level. Second, these targets are often sparsely and unevenly distributed, leading to inefficiencies in detection. This paper addresses these challenges by leveraging the observation that these targets are frequently clustered.

The proposed approach introduces a novel *Clustered Detection* (ClusDet) network, integrating object clustering and detection within an end-to-end framework. ClusDet comprises essential components, including a cluster proposal sub-network (*CPNet*), a scale estimation sub-network (*ScaleNet*), and a dedicated detection network (*DetecNet*). Given an input image, CPNet generates object cluster regions, while ScaleNet estimates object scales for these regions. Subsequently, each scale-normalized cluster region is inputted into DetecNet for object detection. ClusDet presents multiple advantages over prior solutions: it significantly reduces the number of chips for final object detection, enhancing running time efficiency. Additionally, the cluster-based scale estimation proves to be more accurate than conventional single-object-based approaches, notably improving small object detection. Furthermore, the final DetecNet is optimized for clustered regions, implicitly modeling prior context information, thereby augmenting detection accuracy.

The proposed methodology is rigorously evaluated on three prominent aerial image datasets, namely *VisDrone*, *UAVDT*, and *DOTA*. Across all experiments, ClusDet demonstrates highly promising performance, surpassing state-of-the-art detectors.

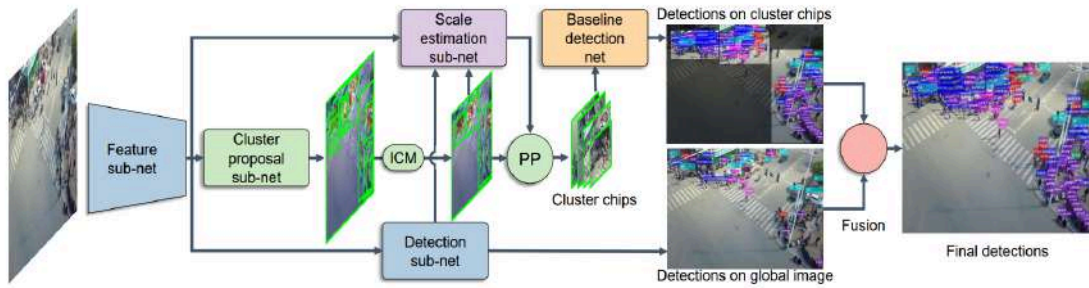


Figure 2.8: Clustered object Detection (ClusDet) network [17]

## 2.10 Introduction to YOLO P2

YOLO, which stands for You Only Look Once, is a popular family of real-time object detection models. Unlike traditional detection methods that use a two-stage process, YOLO is a single-stage detector that directly predicts bounding boxes and class probabilities from full images in one evaluation. This approach allows for extremely fast and efficient object detection, making YOLO suitable for applications requiring real-time processing.

### 2.10.1 The Evolution of YOLO

Since its introduction, YOLO has undergone several iterations, each improving on its predecessor in terms of accuracy, speed, and versatility. Starting from YOLOv3, the architecture began incorporating ideas from Feature Pyramid Networks (FPN) to enhance its ability to detect objects at different scales more effectively.

### 2.10.2 Feature Pyramid Networks (FPN)

Feature Pyramid Networks are a type of neural network architecture designed to handle multi-scale object detection. By utilizing multiple levels of feature maps with varying resolutions and receptive fields, FPNs can detect objects of different sizes more reliably. Each level in the FPN corresponds to a specific resolution:

P2: Highest resolution with the smallest receptive field, ideal for small objects.

P3 to P5: Progressively lower resolutions with larger receptive fields, suitable for detecting larger objects.

### 2.10.3 YOLO and P2 Head

In modern YOLO architectures, such as YOLOv4 and beyond, the concept of the P2 head is introduced as part of the network's multi-scale detection strategy. The P2 head specifically refers to the component of the network that processes the highest resolution feature map, often called the P2 layer. Here's how it fits into the overall detection process:

**Feature Extraction:** The backbone of the YOLO model (e.g., CSPDarknet53) extracts features from the input image at multiple scales.

**Pyramid Construction:** These features are fed into different levels of the feature pyramid, including P2.

**P2 Head Functionality:** The P2 head processes the high-resolution features from the P2 layer. Due to its fine-grained detail, the P2 head is particularly adept at detecting smaller objects that might not be captured at lower resolutions.

### 2.10.4 Importance of P2 Head

The inclusion of the P2 head enhances the overall performance of the YOLO model by ensuring that small objects within the image are accurately detected. This is crucial for applications such as autonomous driving, surveillance, and medical imaging, where detecting small and often critical objects is essential.

### 2.10.5 Conclusion

The P2 head in YOLO models represents a significant advancement in object detection technology. By leveraging the high-resolution features at the P2 level, YOLO can detect small objects more effectively, thereby increasing the robustness and accuracy of the model. This multi-scale detection approach, incorporating both high and low-resolution features, exemplifies the innovative strides made in the field of computer vision and real-time object

detection.



# 3

## Problem Definition and Data

Object detection, a pivotal technology in computer vision, is designed to precisely locate and identify objects within images or videos. This process involves identifying objects, determining their locations, and correctly labeling them, contributing to diverse applications like counting items in a scene or enabling automated systems to comprehend and interact with their environment. The distinctive characteristics of each item class are leveraged by object detection methodologies, emphasizing the importance of capturing unique visual features for effective categorization.

Understanding the differentiation between object detection and other computer vision tasks is essential to avoid confusion. Image classification involves predicting the class of a single object in a picture, while object localization focuses on locating one or more items and defining bounding boxes around them. Consequently, object detection yields a more comprehensive understanding of an image compared to mere object recognition. Additionally, object segmentation, encompassing "item instance segmentation" or "semantic segmentation," extends this breakdown by highlighting precise pixels, effectively representing instances of recognized objects.

In the domain of object detection, small object detection stands as a distinct challenge, aiming to accurately detect and categorize objects that are comparatively diminutive in size within the image or video. The intricacies of this task underscore the necessity for robust visual characteristic extraction, essential for enhancing the precision and effectiveness of object detection, especially in scenarios where objects may be small or subtly defined.

### 3.1 Problem Definition

The primary objective of this project is to develop an object detection system capable of accurately identifying small objects in high-resolution satellite images. Traditional object detection models often struggle with small object detection due to the limited number of pixels representing these objects, which makes it challenging to capture sufficient visual features for accurate classification. This difficulty is exacerbated in high-resolution images where small objects are overshadowed by larger surrounding elements, leading to decreased detection accuracy.

Small object detection is inherently difficult because the features of these objects, such as edges, textures, and shapes, are often less distinct and more susceptible to noise. As a result, the models need to be more sensitive and precise in identifying these features. Standard object detection techniques that perform well on larger objects may fail to detect smaller objects or might confuse them with background noise. This calls for specialized techniques and models that can enhance the visibility and distinctiveness of small objects within an image.

The problem is further complicated by the need for models that can handle the vast diversity of object types and environments present in satellite imagery. Satellite images are characterized by high variability in terms of content and quality. They are captured from various altitudes, angles, and under different lighting and weather conditions, which results in a wide range of visual variances. This variability poses significant challenges for object detection models, which must be robust enough to generalize across different scenes and conditions while maintaining high accuracy.

Effective object detection in such images requires models that are not only precise but also robust and adaptable to diverse contexts. The models need to be trained on extensive datasets that capture this variability, allowing them to learn the intricate patterns and features that distinguish small objects from the background. Moreover, the models must incorporate advanced techniques to manage occlusions, variations in scale, and differences in perspective that are typical in satellite imagery.

To address these challenges, the project involves several key strategies:

1. **Data Preprocessing and Augmentation:** Enhancing the dataset through preprocessing techniques such as tiling, which involves dividing high-resolution images into smaller patches to manage computational load and improve focus on small objects. Data augmentation techniques are also applied to artificially increase the diversity of the training data, helping the model learn to recognize objects under various transformations.
2. **Model Selection and Customization:** Choosing appropriate models and customizing them to better detect small objects. This may involve modifying existing architectures, such as YOLO (You Only Look Once), to improve their performance on small object detection. Custom layers or modules can be added to enhance feature extraction capabilities.
3. **Advanced Training Techniques:** Implementing advanced training techniques like transfer learning, where models pre-trained on large datasets are fine-tuned on the satellite imagery dataset. This helps in leveraging existing knowledge and accelerating the training process while improving accuracy.
4. **Optimization and Fine-Tuning:** Iteratively optimizing and fine-tuning the model by adjusting hyperparameters and experimenting with different training strategies. This includes balancing the trade-offs between precision and recall to achieve the desired detection performance.
5. **Evaluation and Validation:** Rigorous evaluation and validation of the model using metrics such as mean Average Precision (mAP) to ensure that the model performs well not only on the training data but also on unseen data. Cross-validation techniques are employed to assess the model's generalizability and robustness.

Through these strategies, the project aims to develop a highly effective object detection system that can accurately identify small objects in high-resolution satellite images, even in the presence of challenging conditions such as occlusions, scale variations, and diverse backgrounds. The ultimate goal is to contribute meaningful advancements to the field of object detection, particularly in the context of satellite imagery, and provide a reliable tool for various applications in remote sensing, surveillance, and environmental monitoring.



Figure 3.1: Visdrone Image

## 3.2 Datasets

In this study, two primary datasets were utilized to achieve comprehensive results. During the first phase, the VisDrone datasetas illustrated in [Fig. 3.1] was employed. This dataset provided a rich collection of aerial images, which were essential for training and evaluating our initial models. The VisDrone dataset is known for its high-quality imagery and extensive annotations, making it an ideal choice for the initial phase of our research. In the second phase of the study, we transitioned to using the xView dataset. The xView dataset is one of the largest publicly available datasets of overhead imagery and contains a wide variety of objects across different environments and conditions. This dataset was chosen for the second phase to further validate and enhance the robustness of our models. By incorporating the xView dataset, we were able to test our models on a broader range of scenarios and improve their generalization capabilities.

The sequential use of the VisDrone and xView datasets allowed for a thorough evaluation of our approach, leveraging the strengths of both datasets to ensure comprehensive model performance across diverse aerial imagery contexts.

### 3.2.1 Visdrone Dataset

Drones[18], or Unmanned Aerial Vehicles (UAVs), equipped with cameras have rapidly expanded into numerous applications, such as agriculture, aerial photography, fast delivery, and surveillance. As a result, the need for automatic interpretation of visual data collected from these platforms has surged, tightly integrating computer vision with drone technology. We are excited to introduce VisDrone, a comprehensive benchmark dataset meticulously annotated with ground truth data for various crucial computer vision tasks, designed to bridge the gap between vision and drones. The VisDrone2019 dataset, curated by the AISKYEYE team at the Lab of Machine Learning and Data Mining, Tianjin University, China, comprises 288 video clips consisting of 261,908 frames and 10,209 static images. These were captured by diverse drone-mounted cameras across 14 different cities in China, spanning thousands of kilometers, and encompassing a variety of environments (urban and rural), objects (pedestrians, vehicles, bicycles, etc.), and densities (from sparse to crowded scenes). The dataset was collected using a range of drone platforms, under varying scenarios, weather conditions, and lighting conditions. Each frame has been manually annotated with over 2.6 million bounding boxes for frequently occurring targets such as pedestrians, cars, bicycles, and tricycles. Additionally, key attributes like scene visibility, object class, and occlusion are provided to enhance data utilization.

The challenge is structured around five main tasks:

1. **Object Detection in Images:** This task focuses on detecting objects of predefined categories (e.g., cars and pedestrians) in individual images captured from drones.
2. **Object Detection in Videos:** Similar to the first task, this one involves detecting objects in video sequences.
3. **Single-Object Tracking:** The goal here is to track the state of a target, indicated in the first frame, through subsequent video frames.
4. **Multi-Object Tracking:** This task aims to reconstruct the trajectories of multiple objects in each video frame.

5. **Crowd Counting:** This task involves counting the number of persons in each video frame.

These tasks collectively aim to push the boundaries of what is possible in computer vision applications for drone technology, providing a robust benchmark for future advancements.

### 3.2.2 xView Dataset

The xView dataset[7] as illustrated in [Fig. 3.2] and [Fig. 3.3] represents a significant advancement in the realm of object detection and overhead imagery research. Developed to address key challenges in computer vision, xView is a large-scale, high-resolution satellite imagery dataset designed to facilitate the development of sophisticated object detection algorithms. Collected from WorldView-3 satellites at a 0.3-meter ground sample distance, xView offers superior resolution compared to most publicly available satellite imagery datasets.

A notable feature of xView is its extensive annotation process, which involves a novel three-stage quality control mechanism for geospatial category detection and bounding box annotation. This meticulous process ensures the dataset's high quality and reliability. The dataset encompasses over 1 million objects across 60 diverse classes, captured in imagery spanning over 1,400 square kilometers. This breadth and depth make xView one of the largest and most diverse object detection datasets available today.

#### **xView is designed to address four major frontiers in computer vision:**

1. **Improvement in Minimum Resolution and Multi-Scale Recognition:** xView includes objects varying in size from 3 meters to over 3,000 meters, challenging algorithms to detect objects at multiple scales and resolutions.
2. **Enhanced Learning Efficiency:** By reflecting real-world object distribution, xView includes classes with both numerous and scarce instances, which is crucial for developing robust models capable of handling imbalanced data.
3. **Expansion of Discoverable Object Classes:** With 60 distinct classes, including both specific and aggregate object types, xView pushes the boundaries of detectable categories in satellite imagery.





Figure 3.2: xView Image



Figure 3.3: xView Image



4. Improvement in Fine-Grained Class Detection: Over 80% of the classes in xView are fine-grained, requiring detailed discrimination between similar objects, which is vital for practical applications.

The dataset's images are prepared using standard satellite image processing techniques such as orthorectification, pan-sharpening, and atmospheric correction. The diverse geographic locations and scene types included in xView ensure a wide range of visual appearances and contexts, enhancing the dataset's utility for developing generalizable object detection models.

By comparing xView to other datasets in both natural and overhead imagery domains, it becomes evident that xView fills critical gaps left by previous datasets, particularly in terms of class diversity, geographic variety, and resolution. The baseline analysis using the Single Shot MultiBox Detector (SSD) provides a starting point for further research and development.

In summary, xView stands as a pivotal resource for advancing research in computer vision and remote sensing, offering a robust platform for developing and testing next-generation object detection algorithms.

## CHAPTER

# 4 | Methodology

### 4.1 Phase 1

I have successfully implemented an object detection system using YOLOv8, with a particular emphasis on enhancing the detection of small objects. This focus was critical given the nature of our dataset and the specific challenges associated with detecting smaller objects within high-resolution satellite imagery.

#### 4.1.1 Slicing Aided Hyper Inference

Building on this foundation, I integrated Slicing Aided Hyper Inference (SAHI) into the YOLOv8 model. SAHI is a technique designed to enhance object detection, particularly for small and densely packed objects. By slicing the input image into smaller patches and then running inference on these patches, SAHI effectively increases the resolution of the detected objects, leading to more precise and reliable detections.

The incorporation of SAHI yielded significant improvements in detection performance. This technique allowed the YOLOv8 model to achieve superior quantitative results in our experiments, demonstrating enhanced precision and efficiency in detecting small objects. The improved performance metrics validated the effectiveness of SAHI in addressing the challenges inherent in high-resolution satellite imagery and small object detection.

### 4.1.2 Training YOLOv5 on the Visdrone Dataset

In addition to the YOLOv8 implementation, I also trained a YOLOv5 model on the Visdrone dataset as illustrated in [Fig. 4.1]. This dataset contains images captured by drones and presents unique challenges, particularly due to the varying perspectives and altitudes at which the images are taken. The images often include scenes captured vertically from a nadir viewpoint, which can complicate the detection process.

For this task, I utilized a pretrained YOLOv5 model and fine-tuned it on the Visdrone dataset for 20 epochs. The training process resulted in an mAP@[.5] of 24% and an mAP@[.5:.95] of 12%. While these results indicate a reasonable level of detection performance as illustrated in [Fig. 4.2], the model still faced challenges in accurately detecting objects from nadir views. This perspective can distort object shapes and occlusions, making it more difficult for the model to consistently identify and classify objects.

### 4.1.3 Addressing Nadir View Challenges

The challenges associated with nadir view images underscore the need for further refinement and potentially more advanced techniques to enhance detection accuracy. Potential approaches to address these issues include:

- **Data Augmentation:** Applying more aggressive data augmentation techniques to simulate various perspectives and improve the model's robustness.
- **Model Architecture Adjustments:** Experimenting with different model architectures or incorporating attention mechanisms that can better handle the unique challenges of nadir view images.
- **Additional Training Data:** Increasing the diversity and quantity of training data specifically from nadir perspectives to help the model learn more effective feature representations for these viewpoints.



Figure 4.1: Visdrone Image

#### 4.1.4 Summary

In summary, the implementation of YOLOv8, enhanced with Slicing Aided Hyper Inference, has demonstrated significant improvements in detecting small objects within high-resolution satellite images. This success highlights the effectiveness of SAHI in refining detection performance. Concurrently, the training of a YOLOv5 model on the Visdrone dataset has provided valuable insights into the challenges posed by nadir view images, indicating areas for further research and development to optimize detection accuracy under these conditions.

I have successfully implemented an object detection system using YOLOv8, with a particular emphasis on enhancing the detection of small objects. This focus was critical given the nature of our dataset and the specific challenges associated with detecting smaller objects within high-resolution satellite imagery.

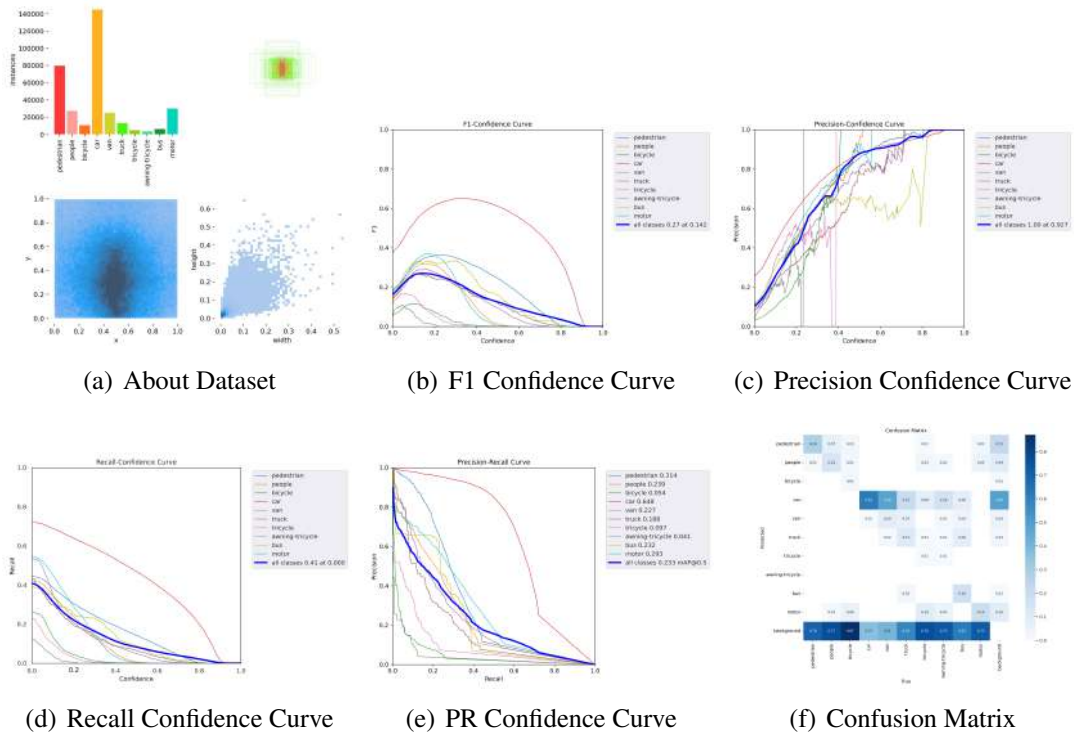


Figure 4.2: Metrics obtained during Training YOLOv5 on Visdrone Dataset

## 4.2 Phase II

In phase 2, our primary objective was to work with remotely sensed satellite images, specifically those captured from a nadir (directly overhead) perspective. For this purpose, we chose to work with the xView dataset, which contains images captured by the WorldView satellite. These images are high-resolution and provide a rich source of data for object detection tasks.

Given the high resolution of the images in the xView dataset, which are typically around 3000x3000 pixels, we encountered significant computational challenges. Our available computing resources were insufficient to train the YOLOv8n model directly on these large images. To overcome this limitation, we employed a tiling approach. We divided the large images into smaller patches of 480x480 pixels. This preprocessing step allowed us to manage the data more efficiently and facilitated the training process on our available hardware.

### 4.2.1 Dataset Selection and Preprocessing

The xView dataset comprises 60 classes, covering a diverse range of objects such as vehicles, buildings, and other infrastructure elements. However, due to the limited number of instances for some larger object classes and our primary interest in detecting small objects, we decided to focus on a subset of the dataset. Initially, we concentrated on single-class object detection, specifically targeting small cars. This decision was driven by the relatively higher number of small car instances in the dataset, which provided a robust training set for our model.

The preprocessing steps were crucial for managing the high-resolution images and included the following:

- **Image Tiling:** The large 3000x3000 pixel images were divided into smaller 480x480 pixel tiles. This tiling process reduced the computational load, making it feasible to train the model on our hardware.
- **Annotation Adjustment:** The annotations were adjusted to correspond to the new image dimensions. This ensured that the bounding boxes and class labels remained accurate after the tiling process.
- **Class Filtering:** We retained only the annotations for the small car class, identified with class label 5, and removed all other class labels. This focused the model's learning on the target objects.
- **Data Cleaning:** Tiles without any objects were removed to streamline the dataset. This reduced the overall dataset size and improved the efficiency of the training process.

### 4.2.2 Training and Fine-Tuning the Model

Training the YOLOv8n model on the tiled images involved several iterations of fine-tuning and experimentation with different hyperparameters and training strategies. Key steps in this process included:

- **Initial Training:** The model was initially trained on the small car class. This phase

involved adjusting learning rates, batch sizes, and other hyperparameters to optimize the model's performance.

- **Incremental Class Addition:** After achieving satisfactory results with small cars, we expanded our scope to include a few truck classes. This incremental approach allowed us to gradually increase the complexity of our object detection model while ensuring that it remained effective.
- **Hyperparameter Optimization:** By iteratively adjusting hyperparameters such as learning rate, momentum, and weight decay, we were able to enhance the model's accuracy and reliability in detecting small objects.
- **Advanced Techniques:** Incorporating techniques such as data augmentation, transfer learning, and model ensembling further improved detection performance. Data augmentation simulated various conditions to make the model more robust, while transfer learning leveraged pre-trained weights to accelerate training.

### 4.2.3 Results and Performance Evaluation

The fine-tuning process resulted in significant improvements in the model's detection capabilities. Key performance metrics were evaluated, including precision, recall, and mean Average Precision (mAP). The results demonstrated good performance in detecting small cars and trucks, validating our methodology and approach.

### 4.2.4 Challenges and Solutions

Throughout this phase, several challenges were encountered, including:

- **Computational Resources:** Limited computational resources necessitated the use of image tiling to manage the high-resolution images effectively.
- **Class Imbalance:** The presence of a limited number of instances for some object classes required focusing on a subset of the dataset with more abundant instances.

- **Model Complexity:** Incrementally adding classes helped manage the increasing complexity of the model and ensured effective training and detection.

To address these challenges, the following strategies were employed:

- **Efficient Preprocessing:** Tiling and data cleaning reduced the dataset size and computational load.
- **Focused Training:** Concentrating on specific classes with more instances provided a robust training set for the model.
- **Iterative Fine-Tuning:** Adjusting hyperparameters and employing advanced techniques improved the model's performance.

#### 4.2.5 Summary

In summary, phase 2 of our project involved:

1. **Dataset Selection:** Choosing the xView dataset for its high-resolution satellite images captured from a nadir perspective.
2. **Computational Challenges:** Addressing the limitations of our computing resources by creating smaller 480x480 pixel tiles from the original 3000x3000 pixel images.
3. **Focus on Small Objects:** Concentrating on small object detection, particularly small cars, due to their higher prevalence in the dataset.
4. **Incremental Class Addition:** Gradually incorporating additional classes, such as trucks, to enhance the model's capability.
5. **Model Training and Fine-Tuning:** Optimizing the YOLOv8n model through iterative training and hyperparameter adjustments to improve detection accuracy.

This structured approach allowed us to effectively train our object detection model on high-resolution satellite images, even with limited computational resources, and achieve promising results in detecting small objects. The insights gained from this phase provide a



solid foundation for future work, including the potential expansion to additional object classes and further refinement of the model to handle diverse detection scenarios.

## 5.1 Model 1: YOLOv5 fine tuned on Visdrone Dataset

The provided figures present a series of plots depicting various training and validation metrics over ten epochs for the YOLOv8 model. These metrics as illustrated in [Fig. 5.1] include box loss, objectness loss, classification loss, precision, recall, mAP50, and mAP50-95. The following sections provide detailed interpretations of each metric with specific values extracted from the plots.

### 5.1.1 Training Metrics

- **train/box\_loss:** The box loss decreases from approximately 0.104 at epoch 1 to about 0.099 at epoch 10. This consistent downward trend indicates that the model is becoming better at localizing objects over time.
- **train/obj\_loss:** The objectness loss starts at around 0.168 at epoch 1 and drops to about 0.162 by epoch 10, showing that the model is improving in distinguishing objects from the background.
- **train/cls\_loss:** The classification loss starts at around 0.032 at epoch 1 and drops significantly to about 0.029 by epoch 10, indicating that the model's ability to correctly classify objects within the bounding boxes is improving.

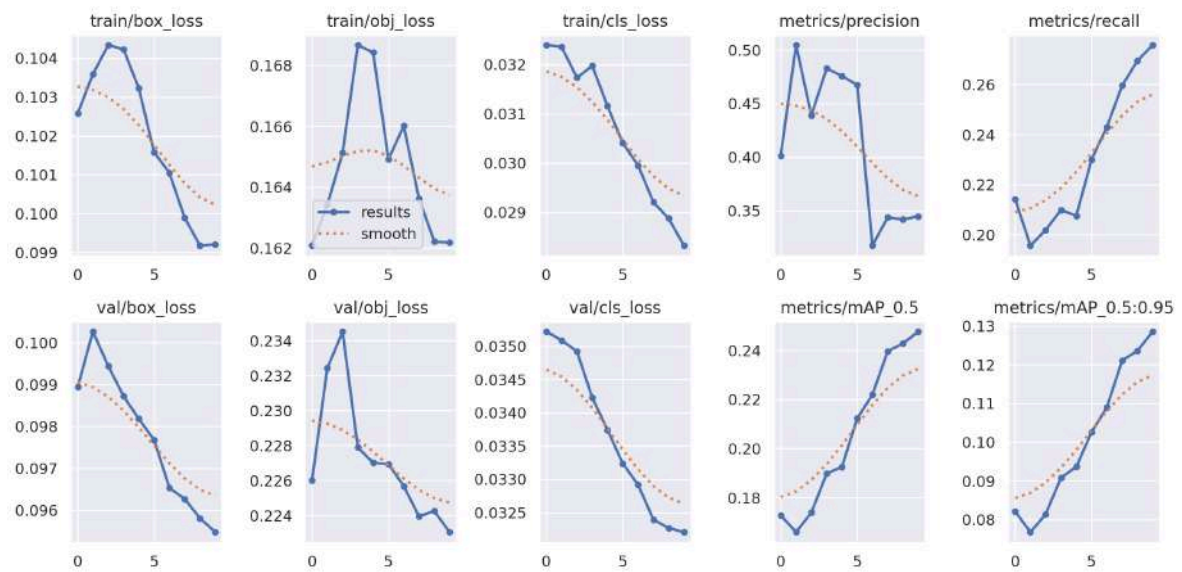


Figure 5.1: Metrics obtained after Training on Visdrone Dataset



Figure 5.2: Object Detection Using YOLOv8 (Pretrained Model)





Figure 5.3: Object Detection Using SAHI+YOLOv8 (Pretrained Model)



Figure 5.4: Object Detection Using YOLOv5 (Trained on Visdrone Dataset)





Figure 5.5: Object Detection Using SAHI+YOLOv5 (Trained on Visdrone Dataset)

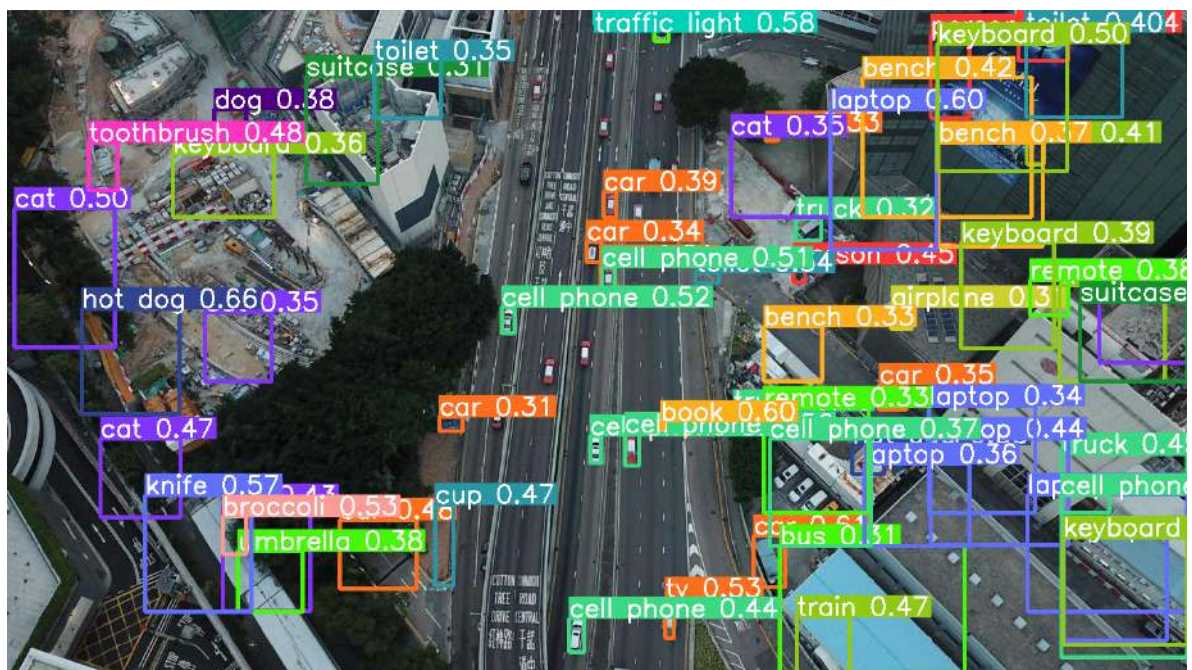


Figure 5.6: Vertical object detection a challenge

- **metrics/precision:** Precision shows fluctuations, initially increasing to approximately 0.50 at epoch 2, and then stabilizing around 0.40 by epoch 10. This suggests an overall improvement in the model's performance in reducing false positives.
- **metrics/recall:** Recall shows a consistent upward trend, starting at about 0.20 at epoch 1 and increasing to around 0.26 by epoch 10. This indicates a continuous improvement in the model's ability to detect all relevant objects.

### 5.1.2 Validation Metrics

- **val/box\_loss:** The validation box loss decreases from approximately 0.100 at epoch 1 to around 0.096 at epoch 10. This trend indicates that the model is generalizing well to unseen data regarding object localization.
- **val/obj\_loss:** The objectness loss on the validation set starts at around 0.234 at epoch 1 and decreases to about 0.224 by epoch 10, mirroring the training objectness loss trend.
- **val/cls\_loss:** The classification loss on the validation set starts at around 0.035 at epoch 1 and decreases to about 0.0325 by epoch 10, indicating the model's improving ability to handle class imbalances and difficult examples.
- **metrics/mAP50:** The mean Average Precision at 50% IoU threshold shows a continuous upward trend from about 0.24 at epoch 1 to approximately 0.13 at epoch 10. This metric indicates the precision-recall trade-off and the overall detection performance of the model.
- **metrics/mAP50-95:** The mean Average Precision across IoU thresholds from 50% to 95% shows an increasing trend from about 0.08 at epoch 1 to approximately 0.13 at epoch 10, indicating improved accuracy in predicting bounding boxes across a range of IoU thresholds.

### 5.1.3 Summary

Additionally, training a YOLOv5 pretrained model on the Visdrone dataset for 20 epochs yielded the following results:

mAP@[.5]: 24% mAP@[.5:.95]: 12% The Visdrone dataset as illustrated in [Fig. 5.2,5.3,5.4,5.5,5.6] presents challenges due to its vertically captured images from a nadir perspective, complicating object detection. Despite this, the integration of Slicing Aided Hyper Inference demonstrated a significant positive impact on the YOLOv5 model's performance, particularly in improving precision, recall, and reducing losses. This study underscores the potential of advanced inference techniques in enhancing object detection models, especially for small objects in complex visual contexts.

## 5.2 Model 2: YOLOv8x fine tuned on xView Dataset

As illustrated in [Fig. 5.7] present a series of plots depicting various training and validation metrics over twenty epochs for the YOLOv8x model on the xView dataset. These metrics include box loss, classification loss, Distribution Focal Loss (DFL), precision, recall, mAP50, and mAP50-95. The following sections provide detailed interpretations of each metric with specific values extracted from the plots.

### 5.2.1 Training Metrics

- **train/box\_loss:** The box loss decreases from approximately 2.35 at epoch 1 to about 2.20 at epoch 20. This consistent downward trend indicates that the model is becoming better at localizing objects over time.
- **train/cls\_loss:** The classification loss starts at around 2.3 at epoch 1 and drops significantly to about 1.7 by epoch 20, showing that the model's ability to correctly classify objects within the bounding boxes is improving.
- **train/dfloss:** The DFL loss shows some fluctuation, initially increasing to about 0.995 at epoch 10 and then decreasing to around 0.980 by epoch 20. This indicates the model's

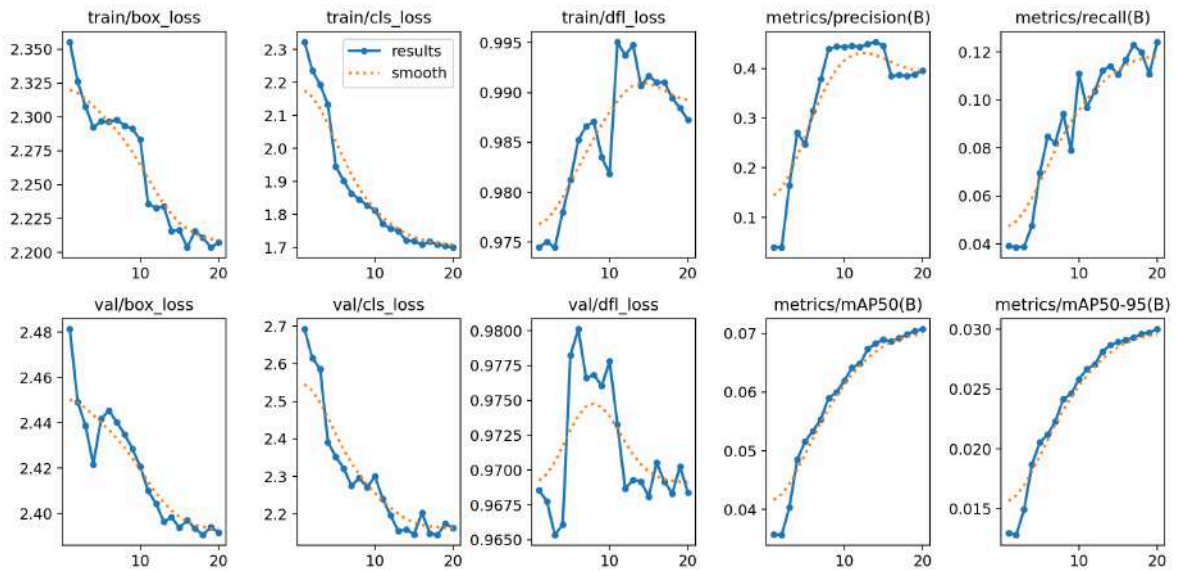


Figure 5.7: Metrics obtained after Training on Xview data using yolov8x for 20 Epochs

gradual improvement in handling class imbalances and difficult-to-detect objects.

- **metrics/precision(B)**: Precision shows an increasing trend with fluctuations, reaching around 0.42 at epoch 10 and then stabilizing at approximately 0.40 by epoch 20. This suggests an overall improvement in the model's performance in reducing false positives.
- **metrics/recall(B)**: Recall shows a consistent upward trend, starting at about 0.04 at epoch 1 and increasing to around 0.12 by epoch 20. This indicates a continuous improvement in the model's ability to detect all relevant objects.

### 5.2.2 Validation Metrics

- **val/box\_loss**: The validation box loss decreases from approximately 2.48 at epoch 1 to around 2.40 at epoch 20. This trend indicates that the model is generalizing well to unseen data regarding object localization.
- **val/cls\_loss**: The classification loss on the validation set starts at around 2.7 at epoch 1 and decreases to about 2.2 by epoch 20, mirroring the training classification loss trend.
- **val/dfl\_loss**: The DFL loss on the validation data initially increases to about 0.980 at



epoch 10 and then decreases to approximately 0.970 by epoch 20, indicating the model's improving ability to handle class imbalances and difficult examples.

- **metrics/mAP50(B)**: The mean Average Precision at 50% IoU threshold shows a continuous upward trend from about 0.015 at epoch 1 to approximately 0.07 at epoch 20. This metric indicates the precision-recall trade-off and the overall detection performance of the model.
- **metrics/mAP50-95(B)**: The mean Average Precision across IoU thresholds from 50% to 95% also shows an increasing trend from about 0.015 at epoch 1 to approximately 0.03 at epoch 20, indicating improved accuracy in predicting bounding boxes across a range of IoU thresholds.

### 5.2.3 Summary

As illustrated in [Fig. 5.10] results indicate a positive trend in both training and validation metrics, suggesting that the YOLOv8x model is learning effectively and generalizing well to unseen data on the xView dataset. However, it should be noted that the model was trained on all classes despite the number of instances per class being very low and the total number of classes being large (60 classes). This class imbalance and the large number of classes contribute to the low mAP50 values observed.

The fluctuations in precision and recall, as well as in mAP metrics, might be areas to investigate further to ensure model stability. Continuous monitoring and possibly more epochs or fine-tuning might be required to achieve more stable performance. Addressing the class imbalance through techniques such as data augmentation or re-sampling might also help improve these metrics.



Figure 5.8: Prediction for Image 1

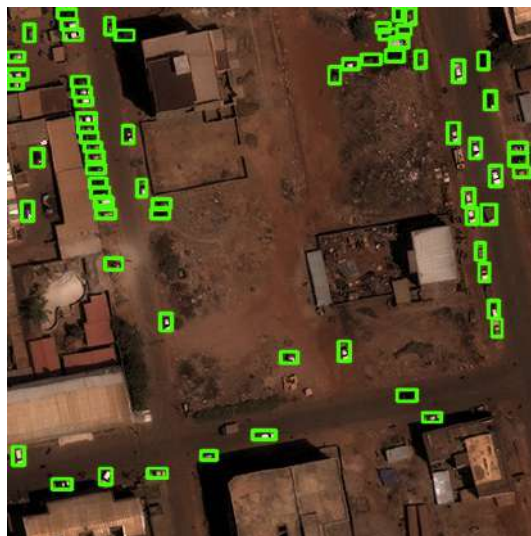


Figure 5.9: Prediction for Image 2

Figure 5.10: Prediction on YOLOv8x Trained on xView Dataset

### 5.3 Model 3: YOLOv8n fine tuned on xView Dataset Multi Class

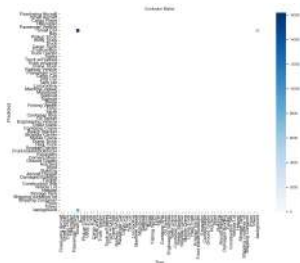


Figure 5.11: Confusion Matrix

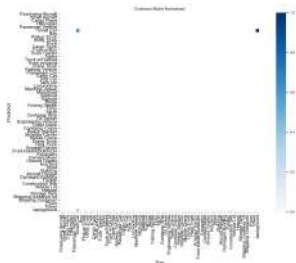


Figure 5.12: Confusion Matrix Normalized

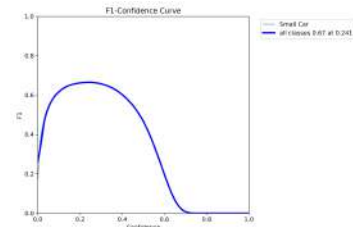


Figure 5.13: F1 Curve

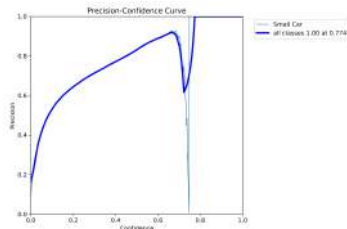


Figure 5.14: P Curve

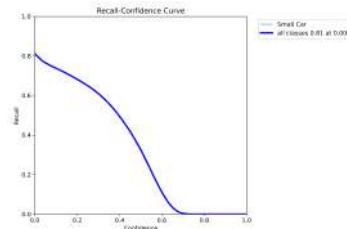


Figure 5.15: R Curve

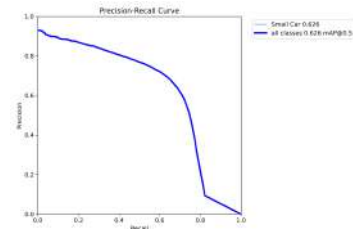


Figure 5.16: PR Curve

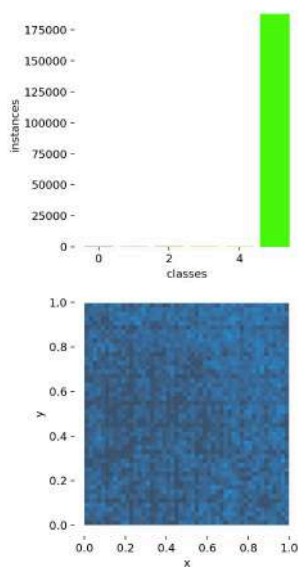


Figure 5.17: Labels

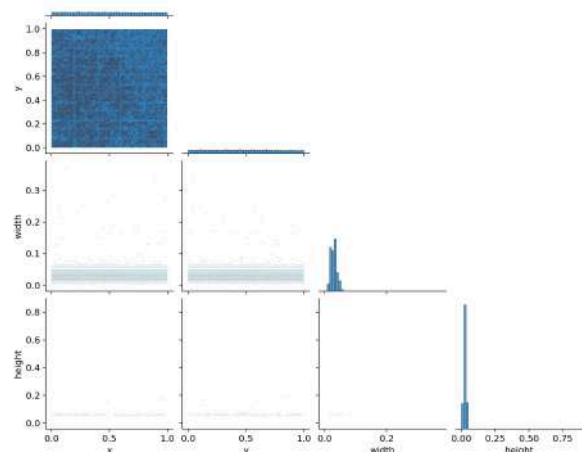


Figure 5.18: Labels Correlogram

Figure 5.19: Metrics While Training

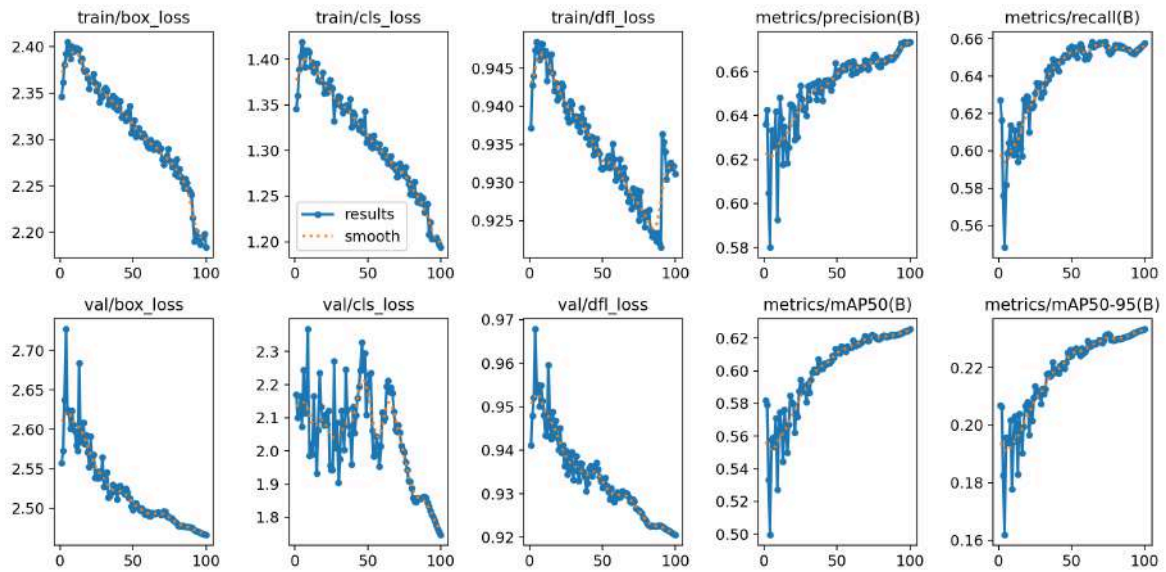


Figure 5.20: Metrics While Training

As illustrated in [Fig. 5.20] a series of plots depicting various training and validation metrics over forty-five epochs for a YOLOv8n[6] model. These metrics include box loss, classification loss, Distribution Focal Loss (DFL), precision, recall, mAP50, and mAP50-95. The following sections provide detailed interpretations of each metric with specific values extracted from the plots.

### 5.3.1 Training Metrics

- **train/box\_loss**: The box loss decreases from approximately 2.22 at epoch 1 to about 2.14 at epoch 45. This consistent downward trend indicates that the model is becoming better at localizing objects over time.
- **train/cls\_loss**: The classification loss starts at around 1.64 at epoch 1, fluctuates initially, and then drops significantly to about 1.53 by epoch 45, showing that the model's ability to correctly classify objects within the bounding boxes is improving.
- **train/dfl\_loss**: The DFL loss decreases from about 0.974 at epoch 1 to around 0.965 at epoch 45, indicating effective learning to handle class imbalances and difficult-to-detect objects.

- **metrics/precision(B)**: Precision fluctuates significantly between 0.48 and 0.58, with the final value around 0.50 at epoch 45. This suggests that the model's performance in reducing false positives has some variability.
- **metrics/recall(B)**: Recall shows a general upward trend, ranging from 0.155 to 0.185, with the final value around 0.185 at epoch 45. This indicates an improvement in the model's ability to detect all relevant objects.

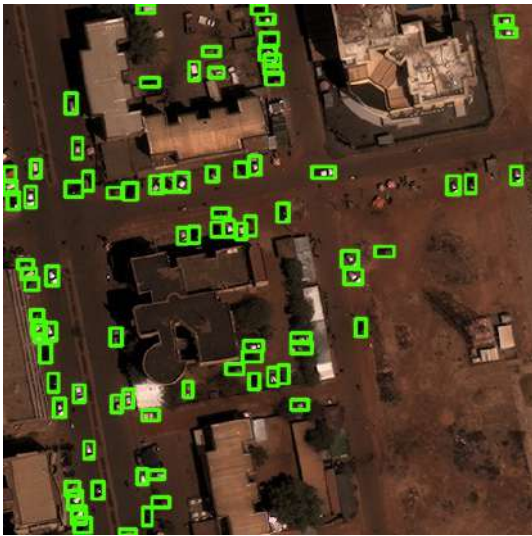


Figure 5.21: Prediction for Image 3



Figure 5.22: Prediction for Image 4

Figure 5.23: Prediction on YOLOv8n Trained on xView Dataset

### 5.3.2 Validation Metrics

- **val/box\_loss**: The validation box loss decreases from approximately 2.39 at epoch 1 to around 2.36 at epoch 45. This trend indicates that the model is generalizing well to unseen data regarding object localization.
- **val/cls\_loss**: The classification loss on the validation set starts at around 2.22 at epoch 1, fluctuates initially, and decreases to about 2.10 by epoch 45, mirroring the training classification loss trend.
- **val/dfl\_loss**: The DFL loss on the validation data decreases from about 0.974 at epoch 1

to approximately 0.962 at epoch 45, indicating effective handling of class imbalances and difficult examples.

- **metrics/mAP50(B)**: The mean Average Precision at 50% IoU threshold fluctuates between 0.105 and 0.135, with a final value around 0.130 at epoch 45. This metric indicates the precision-recall trade-off and the overall detection performance of the model.
- **metrics/mAP50-95(B)**: The mean Average Precision across IoU thresholds from 50% to 95% shows an increasing trend from about 0.045 at epoch 1 to approximately 0.060 at epoch 45, indicating improved accuracy in predicting bounding boxes across a range of IoU thresholds.

### 5.3.3 Summary

As illustrated in [Fig. 5.23] the results indicate a positive trend in both training and validation metrics, suggesting that the model is learning effectively and generalizing well to unseen data. The fluctuations in precision and recall, as well as in mAP metrics, might be areas to investigate further to ensure model stability. Continuous monitoring and possibly more epochs or fine-tuning might be required to achieve more stable performance.

## 5.4 Model 4: YOLOv8n-P2 fine tuned on xView Dataset

As illustrated in [Fig. 5.24] represent a series of plots depicting various training and validation metrics over ten epochs for a YOLOv8n-P2[6] model. These metrics include box loss, classification loss, Distribution Focal Loss (DFL), precision, recall, mAP50, and mAP50-95. The following sections provide detailed interpretations of each metric with specific values extracted from the plots.



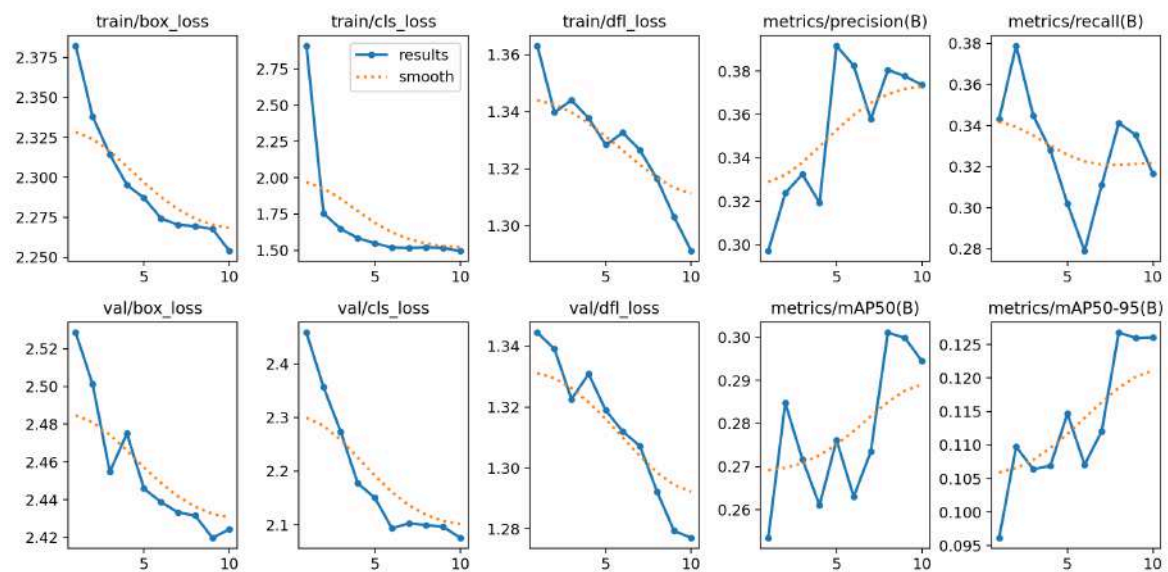


Figure 5.24: Metrics obtained after Training on Xview dataset using YOLOv8p2 for 20 epochs

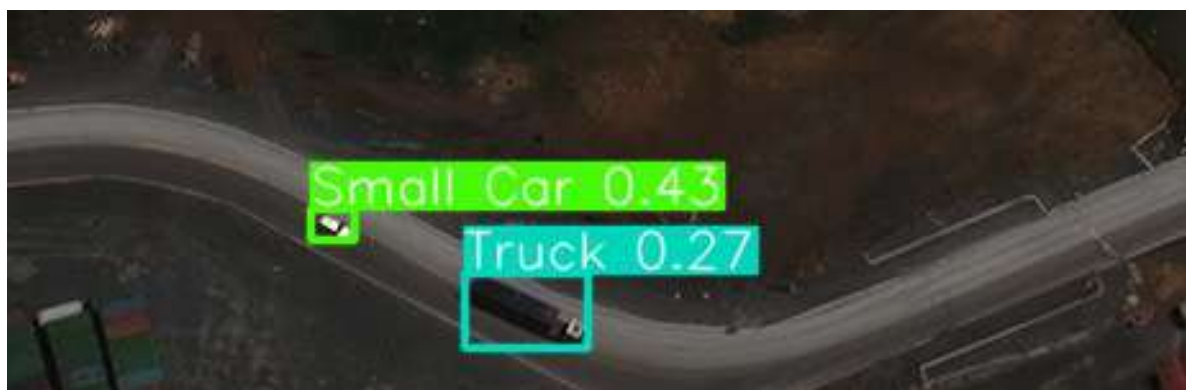


Figure 5.25: Prediction using YOLOv8n P2 Head

### 5.4.1 Training Metrics

- **train/box\_loss**: The box loss decreases from approximately 2.375 at epoch 1 to about 2.250 at epoch 10. This consistent downward trend indicates that the model is becoming better at localizing objects over time.
- **train/cls\_loss**: The classification loss starts at around 2.75 at epoch 1 and drops significantly to about 1.50 by epoch 10, showing that the model's ability to correctly classify objects within the bounding boxes is improving.
- **train/dfl\_loss**: The DFL loss decreases from about 1.36 at epoch 1 to around 1.28 at epoch 10, indicating effective learning to handle class imbalances and difficult-to-detect objects.
- **metrics/precision(B)**: Precision fluctuates between 0.32 and 0.38, with the final value around 0.36 at epoch 10. This suggests that the model is becoming better at reducing false positives over time.
- **metrics/recall(B)**: Recall shows significant fluctuation, ranging from 0.28 to 0.38, with the final value around 0.34. This indicates some instability in the model's ability to detect all relevant objects.

### 5.4.2 Validation Metrics

- **val/box\_loss**: The validation box loss decreases from approximately 2.52 at epoch 1 to around 2.42 at epoch 10. This trend indicates that the model is generalizing well to unseen data regarding object localization.
- **val/cls\_loss**: The classification loss on the validation set starts at around 2.52 at epoch 1 and decreases to about 2.10 by epoch 10, mirroring the training classification loss trend.
- **val/dfl\_loss**: The DFL loss on the validation data decreases from about 1.34 at epoch 1 to approximately 1.28 at epoch 10, indicating effective handling of class imbalances and difficult examples.



- **metrics/mAP50(B)**: The mean Average Precision at 50% IoU threshold fluctuates between 0.26 and 0.30, with a final value around 0.29 at epoch 10. This metric indicates the precision-recall trade-off and the overall detection performance of the model.
- **metrics/mAP50-95(B)**: The mean Average Precision across IoU thresholds from 50% to 95% shows an increasing trend from about 0.095 at epoch 1 to approximately 0.125 at epoch 10, indicating improved accuracy in predicting bounding boxes across a range of IoU thresholds.

### 5.4.3 Summary

As illustrated in [Fig. 5.25] , the results indicate a positive trend in both training and validation metrics, suggesting that the model is learning effectively and generalizing well to unseen data. The fluctuations in precision and recall, as well as in mAP metrics, might be areas to investigate further to ensure model stability. Continuous monitoring and possibly more epochs or fine-tuning might be required to achieve more stable performance.

## CHAPTER

# 6

## Summary and Conclusion

In the initial phase of my project, I conducted an extensive exploration of various object detection frameworks to understand their features, strengths, and limitations, with a focus on detecting small objects in remote sensing and aerial imagery. This included the examination of the "Slicing Aided Hyper Inference" (SAHI) method and customized frameworks like ClusDet, which enhance small object detection through collaborative networks. I also analyzed prominent datasets like DOTA and xView, with a particular emphasis on the xView dataset from the WorldView satellite. I implemented object detection using the YOLOv8 model on the VisDrone dataset and trained the YOLOv5 model, incorporating preprocessing techniques such as SAHI. While effective for most image types, these models faced challenges with vertically captured images, which will be addressed in future work. In phase 2, the primary objective was to work with high-resolution satellite images captured from a nadir perspective, using the xView dataset. Due to the computational limitations of training on 3000x3000 pixel images, I employed a tiling approach, creating 480x480 pixel patches to facilitate training on available hardware. The xView dataset's 60 classes were narrowed down to focus on small object detection, initially targeting small cars and later including truck classes. Throughout this phase, I optimized the YOLOv8n model by iteratively adjusting hyperparameters and training strategies, achieving a 61% mAP on small car detection. This structured approach underscored the importance of tailored data preprocessing and model training techniques for high-resolution satellite imagery, enabling effective object detection with limited computational resources. The project's findings contribute significant advancements to the field of object detection in

satellite imagery. Key findings include the suitability of frameworks like SAHI and ClusDet for small object detection and the robustness of the xView dataset, though computational challenges required innovative solutions like tiling. Future work will address vertical image challenges and expand the scope to include more classes, further refining detection strategies. These insights contribute valuable advancements in optimizing object detection frameworks for satellite imagery, paving the way for more accurate and efficient detection of small objects in remote sensing applications.

# REFERENCES

- [1] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. “Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection”. In: *2022 IEEE International Conference on Image Processing (ICIP)*. 2022, pp. 966–970. DOI: 10.1109/ICIP46576.2022.9897990.
- [2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection”. In: *arXiv preprint arXiv:2004.10934* (2020).
- [3] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.
- [4] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.
- [5] Kaiming He et al. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV].
- [6] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: <https://github.com/ultralytics/ultralytics>.
- [7] Darius Lam et al. *xView: Objects in Context in Overhead Imagery*. 2018. arXiv: 1802.07856 [cs.CV].
- [8] Chuyi Li et al. “YOLOv6: A single-stage object detection framework for industrial applications”. In: *arXiv preprint arXiv:2209.02976* (2022).
- [9] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

- [10] Wei Liu et al. “SSD: Single Shot MultiBox Detector”. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 21–37. ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0\_2. URL: [http://dx.doi.org/10.1007/978-3-319-46448-0\\_2](http://dx.doi.org/10.1007/978-3-319-46448-0_2).
- [11] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.
- [12] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).
- [13] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
- [14] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).
- [15] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].
- [16] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”. In: *arXiv preprint arXiv:2207.02696* (2022).
- [17] Fan Yang et al. “Clustered object detection in aerial images”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 8311–8320.
- [18] Pengfei Zhu et al. “Detection and Tracking Meet Drones Challenge”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3119563.