SMALL OBJECT DETECTION BY AUGMENTING THE DATASET AND INTEGRATING INTERACTIVE MODULES IN THE DEEP LEARNING ARCHITECTURE

(VERİSETİNİN ARTIRILMASI VE İNTERAKTİF MODÜLLERİN DERİN ÖĞRENME MİMARİSİNE ENTEGRE EDİLMESİ İLE KÜÇÜK NESNE TESPİTİ)

by

Elif Melis TAŞKIN, B.S.

Thesis

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

in the

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

of

GALATASARAY UNIVERSITY

September 2024

This is to certify that the thesis entitled

SMALL OBJECT DETECTION BY AUGMENTING THE DATASET AND INTEGRATING INTERACTIVE MODULES IN THE DEEP LEARNING ARCHITECTURE

prepared by **Elif Melis TAŞKIN** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering** at the **Galatasaray University** is approved by the

Examining Committee:

Prof. Dr. Tankut ACARMAN (Supervisor)
Department of Computer Engineering
Galatasaray University ----------

Assist. Prof. A. Teoman Naskali
Department of Computer Engineering
Galatasaray University ----------

Assist. Prof. Selin Nacaklı
Department of Computer Engineering
Bahçeşehir University ----------

Date: ----------

ACKNOWLEDGEMENTS

I am profoundly thankful for the unwavering support and guidance of my advisor, Prof. Dr. Tankut Acarman, throughout my master's thesis. Their expertise and patience have been invaluable, playing a pivotal role in the success of this thesis.

Considering that is one of the hardest times to work both at work and my thesis at the same time, I am also grateful to my parents and my close friends. Their belief in me has kept my spirits and motivation high during this process.

September 2024

Elif Melis TAŞKIN

TABLE OF CONTENTS

LIST OF SYMBOLS

AI	**:** Artifical Intelligence
YOLO	**:** You Only Look Once
CNN	**:** Convolutional Neural Networks
IoU	**:** Intersection over Union
R-CNN	**:** Region-based Convolutional Neural Networks
Mask R-CNN	**:** Mask Region-Based Convolutional Neural Network
COCO	**:** Common Objects in Context
LGIM	: Local Global Interactive Module
ECSI	**:** Enhancing Channel and Space Interaction
ECA	**:** Efficient Channel Attention
mAP	**:** Mean Average Precision
SSD	**:** Single Shot MultiBox Detector
RoI	**:** Region of Interest
SVM	**:** Support Vector Machines
BiFPN	**:** bi-directional feature pyramid network structure
HGA	**:** High Global Attention
LGA	**:** Low Global Attention
FC	**:** Fully Connected
MLGT	**:** Multi-Level Global Transformer
SPP	**:** Spatial Pyramid Pooling
BoF	**:** Bag of freebies
BoS	**:** Bag of specials
E-ELAN	**:** Extended Efficient Layer Aggregation Network
RPN	**:** Region proposal networks

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Recently, image processing, video processing applications have made significant progress via Graphical Processing Unit emerging technology and has been effectively used by several industries such as robotics, augmented reality, autonomous cars, and surveillance systems. Object identification is a fundamental challenge in computer vision, as it serves as the basis for several sophisticated applications. By accurately detecting and localizing objects, computer vision systems may perform complex tasks such as object identification, semantic comprehension, and scene understanding. Driven by improvements in deep learning algorithms and the availability of big annotated datasets, object detection has made tremendous strides over time. Deep neural networks, in particular CNNs, have exceeded earlier techniques relying on manually created features and classifiers. CNNs have demonstrated impressive object detecting capabilities, offering greater accuracy.

The main goal of this thesis is to investigate and assess various object multi-classification methods, with a particular emphasis on deep learning-based methods. In order to develop a lightweight and quick object detection model, the goal has also building a model that could effectively detect objects on the YOLO algorithm with new classes. In this thesis, a suite of models and architectures were tested and compared. The objectives are as follows:

- Determine how object identification techniques have changed over time, from traditional approaches to contemporary deep learning systems.
- Examine the basic ideas and elements of well-known deep learning object recognition models, such as R-CNN, YOLO and Mask R-CNN.
- Analyze the performance of various object detection algorithms while benchmarking with available datasets using statistical-based performance mettics such as accuracy, precision, recall, and run time.

- Examine current developments in object identification, including feature pyramid networks, attention mechanism integration, and one-stage vs. two-stage detectors, stand-alone algorithms.
- Examine object detection's difficulties and restrictions, such as occlusion, size variation, and tiny object detection.

This thesis contributes to:

- Transfer learning and the exploitation of an augmentation of an existing dataset in order to improve the detection performance and increase the statistical-based performance metric values
- Integrate new layers to the deep learning architecture as a new class in order to increase the mAP, accuracy metric and validation values.
- Introduce a new module as the Local Global Interactive Module (LGIM) module to extract dependencies that are, in a conventional sense, both local and global from the input data.
- Introduce a new module as the Enhancing Channel and Space Interaction (ECSI) module to combine as the two as High and Low Order Global Attention.
- Benchmark the COCO dataset and to augment the dataset based on 7 objects that maintains the detection of the object detection thesis
- Evaluate the newly created YOLOv8+ algorithm effect on different datasets and its performance with respect to the former detection algorithm.

The represented technique includes obtaining and examining pertinent books, articles, research papers, and open-source implementations. While benchmarking datasets such COCO, and ImageNet, a variety of object identification techniques are elaborated. To evaluate the benefits and drawbacks of various strategies, performance measurements, comparative analyses, and visualizations will be employed. For further dataset than COCO, new dataset will be applied for the object detection in higher percentage of detection and accuracy.

The goal of this thesis is to provide researchers, academicians, engineers and practitioners a thorough analyze of object detection techniques, their development, and their practical

applications. The conclusions and assessments made in this study will be an invaluable tool for computer vision scholars, professionals, and hobbyists. Additionally, the restrictions and difficulties found might direct ongoing research projects and spark creative ideas for improving object identification systems.

This thesis aims to contribute to the improvement and refinement of object detection techniques, enabling more precise and effective object recognition in a variety of real-world applications by examining the theoretical underpinnings, practical implementations, and recent advancements in object detection.

RÉSUMÉ

Récemment, le traitement d'images et les applications de traitement vidéo ont fait des progrès significatifs grâce à la technologie émergente de l'unité de traitement graphique et ont été efficacement utilisés par plusieurs industries telles que la robotique, la réalité augmentée, les voitures autonomes et les systèmes de surveillance. L'identification d'objets est un défi fondamental en vision par ordinateur, car elle sert de base à plusieurs applications sophistiquées. En détectant et en localisant avec précision les objets, les systèmes de vision par ordinateur peuvent effectuer des tâches complexes telles que l'identification d'objets, la compréhension sémantique et la compréhension de scènes. Poussée par les améliorations des algorithmes d'apprentissage profond et la disponibilité de grands ensembles de données annotées, la détection d'objets a fait d'énormes progrès au fil du temps. Les réseaux neuronaux profonds, en particulier les CNN, ont dépassé les techniques antérieures reposant sur des fonctionnalités et des classificateurs créés manuellement. Les CNN ont démontré des capacités de détection d'objets impressionnantes, offrant une plus grande précision.

L'objectif principal de cette thèse est d'étudier et d'évaluer diverses méthodes de multi-classification d'objets, en mettant l'accent sur les méthodes basées sur l'apprentissage profond. Afin de développer un modèle de détection d'objets léger et rapide, l'objectif est également de construire un modèle capable de détecter efficacement des objets sur l'algorithme YOLO avec de nouvelles classes. Dans cette thèse, une série de modèles et d'architectures ont été testés et comparés. Les objectifs sont les suivants :

- Déterminer comment les techniques d'identification d'objets ont évolué au fil du temps, des approches traditionnelles aux systèmes d'apprentissage profond contemporains.

- Examiner les idées et éléments de base des modèles de reconnaissance d'objets d'apprentissage profond bien connus, tels que R-CNN, YOLO et Mask R-CNN.
- Analyser les performances de divers algorithmes de détection d'objets tout en les comparant aux ensembles de données disponibles en utilisant des métriques de performance basées sur des statistiques telles que l'exactitude, la précision, le rappel et le temps d'exécution.
- Examiner les développements actuels dans l'identification d'objets, y compris les réseaux pyramidaux de caractéristiques, l'intégration des mécanismes d'attention et les détecteurs à une ou deux étapes, les algorithmes autonomes.
- Examiner les difficultés et les restrictions de la détection d'objets, telles que l'occlusion, la variation de taille et la détection d'objets minuscules.

Cette thèse contribue à :

- Transfert d'apprentissage et exploitation d'une augmentation d'un ensemble de données existant afin d'améliorer les performances de détection et d'augmenter les valeurs de métriques de performance basées sur les statistiques
- Intégrer de nouvelles couches à l'architecture d'apprentissage profond en tant que nouvelle classe afin d'augmenter les valeurs mAP, de métrique de précision et de validation.
- Introduire un nouveau module en tant que module Local Global Interactive Module (LGIM) pour extraire les dépendances qui sont, au sens conventionnel, à la fois locales et globales des données d'entrée.
- Introduire un nouveau module en tant que module Enhancing Channel and Space Interaction (ECSI) pour combiner les deux en tant qu'attention globale d'ordre élevé et faible.
- Étalonner l'ensemble de données COCO et augmenter l'ensemble de données sur la base de 7 objets qui maintiennent la détection de la thèse de détection d'objets
- Évaluer l'effet de l'algorithme YOLOv8+ nouvellement créé sur différents ensembles de données et ses performances par rapport à l'ancien algorithme de détection.

La technique représentée comprend l'obtention et l'examen de livres, d'articles, de documents de recherche et d'implémentations open source pertinents. Lors de l'évaluation

comparative d'ensembles de données tels que COCO et ImageNet, diverses techniques d'identification d'objets sont élaborées. Pour évaluer les avantages et les inconvénients de diverses stratégies, des mesures de performance, des analyses comparatives et des visualisations seront utilisées. Pour un ensemble de données plus large que COCO, un nouvel ensemble de données sera appliqué pour la détection d'objets avec un pourcentage de détection et une précision plus élevés.

L'objectif de cette thèse est de fournir aux chercheurs, universitaires, ingénieurs et praticiens une analyse approfondie des techniques de détection d'objets, de leur développement et de leurs applications pratiques. Les conclusions et évaluations faites dans cette étude seront un outil précieux pour les chercheurs, les professionnels et les amateurs de vision par ordinateur. De plus, les restrictions et les difficultés rencontrées pourraient orienter les projets de recherche en cours et susciter des idées créatives pour améliorer les systèmes d'identification d'objets.

Cette thèse vise à contribuer à l'amélioration et au raffinement des techniques de détection d'objets, permettant une reconnaissance d'objets plus précise et plus efficace dans une variété d'applications du monde réel en examinant les fondements théoriques, les implémentations pratiques et les avancées récentes dans la détection d'objets.

ÖZET

Son zamanlarda, görüntü işleme, video işleme uygulamaları, ortaya çıkan Grafiksel İşleme Birimi teknolojisi aracılığıyla önemli ilerlemeler kaydetti ve robotik, artırılmış gerçeklik, otonom arabalar ve gözetleme sistemleri gibi çeşitli endüstriler tarafından etkili bir şekilde kullanıldı. Nesne tanımlama, çeşitli karmaşık uygulamalar için temel teşkil ettiği için bilgisayar görüşünde temel bir zorluktur. Nesneleri doğru bir şekilde algılayarak ve yerelleştirerek, bilgisayar görüşü sistemleri nesne tanımlama, anlamsal kavrama ve sahne anlama gibi karmaşık görevleri gerçekleştirebilir. Derin öğrenme algoritmalarındaki gelişmeler ve büyük açıklamalı veri kümelerinin kullanılabilirliği ile yönlendirilen nesne algılama, zamanla muazzam ilerlemeler kaydetti. Derin sinir ağları, özellikle CNN'ler, elle oluşturulan özelliklere ve sınıflandırıcılara dayanan önceki teknikleri aştı. CNN'ler, daha fazla doğruluk sunarak etkileyici nesne algılama yetenekleri gösterdi.

Bu tezin temel amacı, özellikle derin öğrenmeye dayalı yöntemlere vurgu yaparak çeşitli nesne çoklu sınıflandırma yöntemlerini araştırmak ve değerlendirmektir. Hafif ve hızlı bir nesne algılama modeli geliştirmek için, hedef aynı zamanda yeni sınıflarla YOLO algoritmasında nesneleri etkili bir şekilde algılayabilen bir model oluşturmaktır. Bu tezde, bir dizi model ve mimari test edildi ve karşılaştırıldı. Amaçlar şunlardır:

- Nesne tanımlama tekniklerinin geleneksel yaklaşımlardan çağdaş derin öğrenme sistemlerine kadar zaman içinde nasıl değiştiğini belirleyin.
- R-CNN, YOLO ve Mask R-CNN gibi iyi bilinen derin öğrenme nesne tanıma modellerinin temel fikirlerini ve öğelerini inceleyin.
- Doğruluk, kesinlik, geri çağırma ve çalışma zamanı gibi istatistik tabanlı performans ölçütlerini kullanarak mevcut veri kümeleriyle kıyaslama yaparken çeşitli nesne algılama algoritmalarının performansını analiz edin.

- Özellik piramit ağları, dikkat mekanizması entegrasyonu ve tek aşamalı ve iki aşamalı dedektörler, bağımsız algoritmalar dahil olmak üzere nesne tanımlamadaki güncel gelişmeleri inceleyin.
- Tıkanıklık, boyut değişimi ve küçük nesne algılama gibi nesne algılamanın zorluklarını ve kısıtlamalarını inceleyin.

Bu tez şunlara katkıda bulunur:

- Algılama performansını iyileştirmek ve istatistik tabanlı performans metriği değerlerini artırmak için transfer öğrenimi ve mevcut bir veri setinin bir artırımının kullanılması
- mAP, doğruluk metriği ve doğrulama değerlerini artırmak için yeni katmanları derin öğrenme mimarisine yeni bir sınıf olarak entegre edin.
- Giriş verilerinden hem yerel hem de küresel olan bağımlılıkları geleneksel anlamda çıkarmak için Yerel Küresel Etkileşimli Modül (LGIM) modülü olarak yeni bir modül tanıtın.
- İkisini Yüksek ve Düşük Dereceli Küresel Dikkat olarak birleştirmek için Kanal ve Uzay Etkileşimini Geliştirme (ECSI) modülü olarak yeni bir modül tanıtın.
- COCO veri setini kıyaslayın ve nesne algılama tezinin algılanmasını koruyan 7 nesneye dayalı veri setini artırın
- Yeni oluşturulan YOLOv8+ algoritmasının farklı veri setleri üzerindeki etkisini ve önceki algılama algoritmasına göre performansını değerlendirin. Temsil edilen teknik, ilgili kitapları, makaleleri, araştırma makalelerini ve açık kaynaklı uygulamaları edinmeyi ve incelemeyi içerir. COCO ve ImageNet gibi veri kümelerini kıyaslarken, çeşitli nesne tanımlama teknikleri ayrıntılı olarak açıklanır. Çeşitli stratejilerin faydalarını ve dezavantajlarını değerlendirmek için performans ölçümleri, karşılaştırmalı analizler ve görselleştirmeler kullanılacaktır. COCO'dan daha fazla veri kümesi için, daha yüksek algılama ve doğruluk yüzdesinde nesne algılama için yeni veri kümesi uygulanacaktır.

Bu tezin amacı, araştırmacılara, akademisyenlere, mühendislere ve uygulayıcılara nesne algılama teknikleri, bunların gelişimi ve pratik uygulamaları hakkında kapsamlı bir analiz sağlamaktır. Bu

çalışmada yapılan sonuçlar ve değerlendirmeler, bilgisayarlı görme bilim insanları, profesyoneller ve hobiciler için paha biçilmez bir araç olacaktır. Ek olarak, bulunan kısıtlamalar ve zorluklar devam eden araştırma projelerini yönlendirebilir ve nesne tanımlama sistemlerini iyileştirmek için yaratıcı fikirlere yol açabilir.

Bu tez, nesne algılama tekniklerinin iyileştirilmesine ve iyileştirilmesine katkıda bulunmayı, nesne algılamadaki teorik temelleri, pratik uygulamaları ve son gelişmeleri inceleyerek çeşitli gerçek dünya uygulamalarında daha hassas ve etkili nesne tanıma olanağı sağlamayı amaçlamaktadır.

1. INTRODUCTION

In this economically growing world, it is a fact that object detection is widely used for many real-world applications. Some examples to highlight would be the detection of cancer cases for early diagnosis in the healthcare sector; or the auto-pilot feature in the transportation sector which utilizes relevant subjects like computer vision, machine learning, and video camera raw image inputs. CNNs are then used to the raw video to track and detect objects. The concept of object detection is used for many applications like face recognition, pedestrian detection, vehicle detection, counting humans, text detection, and medical imaging. The object is defined by designating a colored box called the bounding box. Defining the model structure and adjusting the parameters is the very first step of object detection. Further steps consist of testing and training. Training is the step for basically teaching the algorithm to learn by itself. Once trained, it will be able to define an object that has been processed before. Testing step is necessary for the algorithm to test itself in percentages of success rate. The sample numbers differ for each step. For generalization purposes, sample rates used are 60% for training, 20% validation and 20% for testing. Finally, evaluation of the results, assessment of accuracy rates take place as the last step for object detection. Success rates and cases are observed for the selected algorithm. When the training step of the algorithm reaches a valid level, it directs itself to the testing stage. Important to note that this consumes a lot of memory and computational power. This is the general perspective for object detection using a custom dataset.

There is a stage where one-staged or two-staged algorithm has been done, compared and decided. For two-staged algorithm, R-CNN and Mask R-CNN has been performed. For one-staged algorithm, YOLOv5 and YOLOv8 has been performed and made comparison with each other in order to decide on which algorithm type should be used. It has been expertise how successful and fast in terms of developing the one-staged algorithms. The further development is done with using the latest YOLO which is YOLOv8 and newly algorithm is named as YOLOv8+.

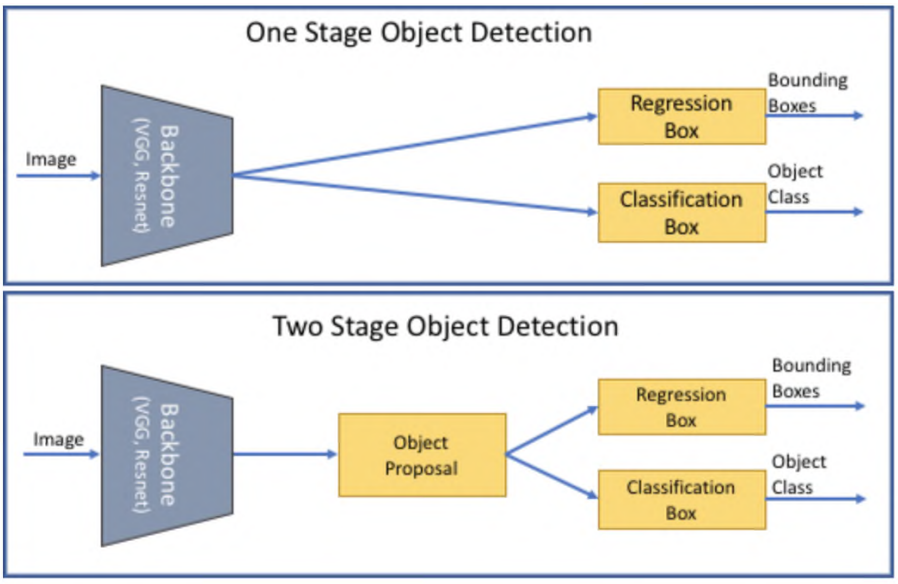

Fig 1.1: Block diagram demonstrating the difference between one stage and two stage algorithm.

As can be seen in Figure 1.1, the task at hand and the properties of the data determine whether a one-stage or a two-stage approach performs better. When a more complex or sophisticated method is required, two-stage algorithms are frequently employed, with each step is focused on a distinct part of the issue. On the contrary, one-stage algorithms are less complicated and may be better when a straightforward approach is preferred.

In this thesis, research is focused on improving detection rate of small objects. And as a real-life use case, items forgotten on the overheads of the passenger seats are detected in order to generate warning to the driver of the coach bus to take the forgotten items before leaving the bus. And its dataset consists of the images taken samples from inside the bus, places overheads luggage parts. The objects are taken captures with the usage of cameras mounted inside the roof of the bus and their captures were recorded by a computer. This system helped for the creation of new dataset and augmenting the existing COCO dataset. An increase in performance metric levels has been observed by using the new dataset which consists of 7 classes, which are as follows:

- backpack,
- book,

- laptop,
- phone,
- eye glass,
- umbrella,
- luggage.

There are 1799 images in the main dataset, and more images are synthetically generated with auto-orient, static crop, resize and greyscale. The number of images reaches at the total of 4497 images in this increased dataset. It was divided with respect to the percentage values of train, test and validation files.

In order to compare the newly designed architecture for object identification and detection with the benchmark performance of the most advanced deep learning model, a mixture of widely known datasets has been used. A variety of techniques and architectural modifications, such as feature pyramid networks for merging high- and low-level features and sophisticated data augmentation, have been used to improve training and performance. Convolution layers are used as the main key for attention module implementation to obtain channel-wise attention. For the final stage, two new architectures have been created for better training performance.

The first newly introduced architecture; ECSI is introduced as the procedures move ahead. The two components referred to as High and Low Order Global Attention are combined in the Enhancing Module. Neural networks, especially transformer models, employ both low-order and high-order global attention techniques to find connections between various input data segments. Low-Order Global Attention is a model that combines data from all around the input space in a comprehensible way. The second architecture that has been introduced named LGIM. It consists of local and global parts as of using the development for fastening the process for detection.

Above-mentioned methods are explained in detail with video and image samples from inside a bus. This is followed by explaining how YOLOv5, YOLOv8, R-CNN and Mask R-CNN methods are being used and improved using normal COCO and newly created datasets and adding newly introduced architecture YOLOv8+ to the tables as well. The comparison for COCO dataset and newly created dataset is the two titles for comparison for this research.

2. LITERATURE REVIEW

This literature review aims to form a comparison between common object detection algorithms by analyzing several papers on the subject. First of all, this subject in general has gained immense popularity with the advent of prompt commanded AI. The algorithms studied by these papers are mostly the same considering their popularities.

Object detection using the COCO dataset [Jain et al.,2022] and Python is used for various technics. The COCO dataset is a widely used object identification benchmark dataset used by AI and Computer Vision designers for various PC vision projects. The discovery of the item is completed through two tasks: grouping and detection. The course classifier is used in grouping the articles, with approximately 87% of accuracy achieved.

The aim is to conduct a thorough examination of SSD, Faster-RCNN, and YOLO. Faster R-CNN is a convolutional neural network-based method for object localization that is cohesive and exhibits increased speed. The use of YOLO is being discussed, which predicts the edge position and rank of multiple competitors. The paper concludes with a comprehensive investigation on a PASCAL VOC dataset, demonstrating a high level of YOLO usage.

Also a new method has been introduced for recognizing objects in photos using a single deep brain architecture, known as the SSD. SSD relies on the entire disposal of the cycle that generates a proposition and uses multiscale convolutional bouncing box yields linked to several element maps. The Tiny SSD, a solitary shot recognizing profound convolutional brain organization, aims to make ongoing object recognition easier. In addition to providing important insights on CNN subjects including its hybrid nature, fragmented highlight maps, interaction redundancy, and different pooling layers, the paper addresses advanced learning procedures for object detection frameworks. SSD has several features, such as a feed-forward organization with a fixed-size collection of jumping boxes, and is simple to set up and include into frameworks.

In the article named; "*Detection of Bolts and Nuts of Automobile Sheet Metal Parts Based on YOLOv7"*. This paper presents a target detection method based on YOLOV7 for automatic detection of bolts and nuts on automobile sheet metal parts. Traditional detection methods often rely on manual work, leading to slow detection speed and low accuracy. The paper uses a target

detection method based on YOLOV7 to achieve a final mAP0.5 of 94.89% and a mAP0.5:0.9 of 75.26%. The model's detection effect increases from 69.7% to 94.89% when trained on a coco dataset. The paper also discusses the rapid development of industrial technology in China, particularly computer technology and image processing technology, which has led to the development of automated detection methods for automobile sheet metal parts. The paper presents a deep learning method for detecting nuts and bolts on automobile sheet metal parts, focusing on the problem of large error and low accuracy in traditional detection methods.

The AlexNet network model was first proposed in 2012, leading to the development of deep learning algorithms. The transformer model, a fourth type of deep learning model, has been applied to various fields. Nicolas Carion proposed a simple CNN+ transformer framework (DETR) for object detection, which is an end-to-end model based on ensemble prediction. Xizhou Zhu proposed a variability DETR network model based on DETR to address slow training speed and poor detection effect of small objects. Despite their popularity, transformers framework models are limited for general researchers due to their large model parameter framework. The YOLO series of object detection algorithms is widely used due to their simple model structure and small model structure parameters. The detection of bolts and nuts is based on the YOLOV7 target detection model. With 300 epochs, the sheet metal part data is being trained and it increases 69.7% to 94.97% accuracy.

In another article named "*Object Detection On Bottles Using the YOLO Algorithm"*; Using the YOLOv3 and COCO datasets, this project attempts to develop an easily practiced basic architectural model as well as a computer that can identify bottles. Regression is the foundation of the YOLO method, which uses a single algorithm to forecast both the picture's bounding box and trajectory. This study makes use of the COCO dataset, which has 123,287 pictures, 886,284 labels, and 8,880 labels for bottles. One of the things in the dataset is the bottle, and recycling bottle trash can be facilitated using object identification. YOLO is well regarded for its accuracy and real-time speed in identifying bottles. Here, COCO dataset is being used for garbage as it has a significant issue worldwide, needs to be detected worldwide as it has a number of 19 tons of waste daily. Technology can help reduce plastic pollution by monitoring waste, such as plastic bottles, with efficiency and low operation costs. The YOLO algorithm, which can detect objects with 94% accuracy, is an excellent solution for detecting plastic waste. YOLO can work on low-quality

cameras and can view videos in real time at 45 FPS.YOLO algorithm and the COCO dataset, which includes 8,880 labels for bottles. The YOLO algorithm is chosen due to its 87% accuracy rate.

In the article (Konola et al., 2023) YOLOv5 and YOLOv7, for detecting objects in live video footage. The YOLOv5 uses anchor boxes to detect objects in images and is successful in prediction based on probabilities. However, it has limitations such as misclassification when objects are small in size, fails in angle deflection, and has limited scalability. The YOLOv7 is a real-time object detection model that uses instance segmentation. The model is implemented with a live video to detect whether a two-wheeler rider is wearing a helmet, as wearing a helmet is a safety measure while driving on busy roads in India. The YOLOv7 has achieved higher mean average accuracy than YOLOv5, and its performance is crucial in traffic surveillance applications. To analyze the performance between YOLOv5 and YOLOv7, improve the accuracy and speed of object detection in various applications by introducing new architecture designs, network optimizations, and training strategies.

Helmet detection is crucial for traffic police to ensure riders' safety. YOLO uses a single neural network to predict bounding boxes and associated class probabilities directly from complete images in a single assessment. This integrated architecture is fast and can process images in real-time, with the fundamental YOLO model processing at 45 frames per second. YOLOv5 is an updated version of the YOLO object detection algorithm, developed by Ultralytics in 2020. It is built on a deep convolutional neural network trained on a large dataset of annotated images. YOLOv5 is fast, capable of real-time object detection, making it suitable for applications like autonomous vehicles and security systems. Its architecture is based on a modified version of the Efficient Net backbone, achieving high accuracy with fewer parameters.

In the article (Shan et al.,2023), it has new improvements based on BiFPN. Furthermore, the YOLOv5's decoupling head enhances convergence rate and detection performance. Hence, these changes on algorithm of YOLOv5 had increased the model’s performance. Considering mAP score, in Yolov5s it is 92.9%, in Yolov5s+ECA 92.2%, in Yolov5s+SE 95.0%, in Yolov5x 95.7% and finally with Improved Yolov5 95.9%. It increased all the values for mAP scores. For considering the improvement of BiFPN, the mAP score is increased from in YOLOv5 92.9% to in YOLOv5-BiFPN 95%.

The article (Cai et al.,2022) is based on the percentage of using COCO dataset. Since it increases the usage as 1% COCO, 2% COCO, 5% COCO, 10% COCO and COCO-Full; in any method that they have used such as Baseline, CSD, STAC, Instant-teaching, Unbiased Teacher or

'Ours', the values tend to increase. Even when they use full of COCO dataset, the results compared for all methods are the highest in COCO-Full. In the model overview, they added on teachers and student module Convolutional Block Attention Module (CBAM) and added a connection between student from teacher module with Exponential Moving Average (EMA).

In the conference paper, the given (Li et al.,2020) data preparation part is as data augmentation which is essential to the object detection procedures in order to increase mAP from 65.5% to 74.3%. For default box forms, the mAP increases from 71.6% to 74.3%. In this paper, Laptop and Suitcase objects are defined same as in this object detection thesis. The system models used are F-CNN, YOLO and SSD512. The mAP values are respectively, 73.2, 66.4 and 76.8. [30]

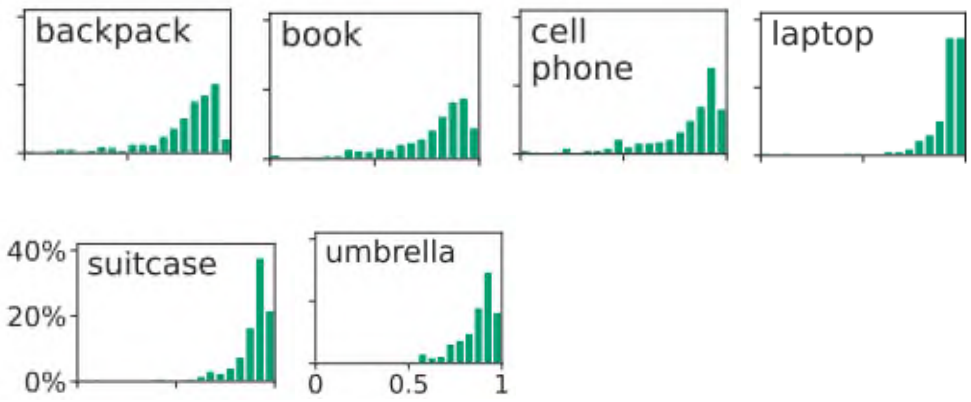

Fig 2.1: IoU over percentage of masks

Considering these values in following way, the values represent mAP results are better in laptop or suitcase (luggage) and umbrella compared to book and backpack. The mAP scores are respectively 73%, 77%, 76%, 66% 62%. It is clear that mAP scores are lower than 80%. In overall, considering all the dataset in COCO used, the used algorithm is YOLOv5, and the mAP score is 74.1%. With

a mean average performance percentage of 86.4%, YOLOv8 outperforms YOLOv7 in additional ways (Kang&Kim,2023).

Table 2.1: mAP, Precision, Recall and F1 scores for all dataset

Objects	**mAP**	**Precision**	**Recall**	**F1 Score**
backpack	62%.	0.5738	0.6732	0.5554
umbrella	76%	0.6945	0.7446	0.6736
book	66%	0.6027	0.6946	0.6352
glasses	69%	0.6767	0.6121	0.6834
laptop	73%	0.6237	0.6736	0.6936
luggage	77%	0.7123	0.7263	0.7587
phone	71%	0.7043	0.7124	0.7045

This table is quite important to represent the objects that has been used to see their results based on mAP, precision, recall and F1 scores with the usage of YOLOv5 with COCO dataset. The scores are low with the percentage in detail, as mentioned on YOLOv5. Since YOLOv5 will be thoroughly analyzed in comparison with YOLOv8, this results or study provides clarity on Table 2.1.

3. THE METHODOLOGY

3.1.Object Detection and Methods

Object detection is a crucial task in computer science, laying the foundation for other vision tasks. There are two main categories of object detection algorithms: artificial and deep learning. Machine Learning algorithms rely on manual feature extraction, while deep learning algorithms use network models to automatically extract features.

Object identification is a difficult process in determining with variety of methods where objects appear in an image and with which class each item belongs. It has gained significant attention due to its close relationship with video examination and image acquisition. Traditional object recognition techniques are based on carefully assembled highlights and shallow teachable designs. To gain comprehensive object recognition knowledge, it is essential to group similar images and evaluate the thoughts and regions of articles featured in each image. Object detection involves perceiving, recognizing, and finding objects with precision.

Considering an important method; CNN is the key part for this subtitle, as it learns from an input image the structural information which means that high- and low-level characteristics. The term "convolution neural network" describes how the network applies the convolutional mathematical process. One particular kind of specialized linear procedure is convolution. Neural networks having at least one layer that employ convolution rather than general matrix multiplication are known as convnets. A convolution procedure on a 2D image I with kernel K may be shown with an output;

The input image is processed by a kernel using simple matrix multiplication and addition, producing a lower-dimensional output that is more semantically accessible in a specific feature than the original image. As convolution layer takes the image in raw format and convolves the part one by one as it suggests from the filter; it moves forward to pooling layers.

Different filtering points out different convolved images as in output. For further stages, it will be presented for this convolution operation in new dataset creation part. The datasets are created with the effect of new convolution layers and the number of images on new created dataset increases. Whether all the data are being recognized by YOLO, shows how well-created this dataset is.

Another method is anchoring boxes. This theme has become important with YOLO algorithm as it suggest object detection in single detection. There would be numbers and numbers of anchor boxes, and all predict the value of the object whether it is the object or not. It has a value regarding the percentage, highest being 1.0 and would go around 0.5-0.7-0.9 etc. Whether the value is greater than 0.5, it takes into account for using as an object detection as considering more accurately predict the items' ground-truth boundary boxes. Creating numbers of anchor boxes is how these region samplings are carried out. Anchor boxes are mainly centered boxes that are centered on every pixel in the image and have different aspect ratios and sizes. The anchor boxes in figure 5 below have the same center coordinates but vary in size and aspect ratio.

IoU is such an important metric that helps viewer to make an easy comment, that shows how valid the algorithm is, object is detected in high percentage. This metric also uses localization as a metric and shows the relation between the ground-truth bounding box and predicted bounding box. IoU is basically the ration between these values. This metric indicates the proximity of the two bounding boxes. The junction area of the two boxes divided by their union area gives the value for the IoU. If the value is considered high, can be judged as successful.

3.2. YOLOv1 - YOLOv2 - YOLOv3

A one-stage detector is YOLO (Redmon et al,2016). Rather of classifying objects based on area suggestions, it views object identification as a regression issue. The fundamental idea is to partition the input into a (S S) grid, where each cell regresses the bounding box position and the confidence level that results from the object center falling within it. It presents YOLO, a unified and real-time item detection system. In a single forward run through the neural network, YOLO is intended to forecast bounding boxes and class probabilities for several objects in an image simultaneously. With this method, YOLO can detect objects in real time, which makes it effective and appropriate for a range of uses. Each image is predicted by the YOLO model to have numerous bounding boxes, but each box vector indicates the likelihood that an item will be in the cell. Using non-max suppression and the computation of the IOU score, the vectors with an accurate bounding box of

the item among them are identified. The loss function shown below is used to train the model. For box and class prediction, it is primarily a regression loss function, as previously indicated.

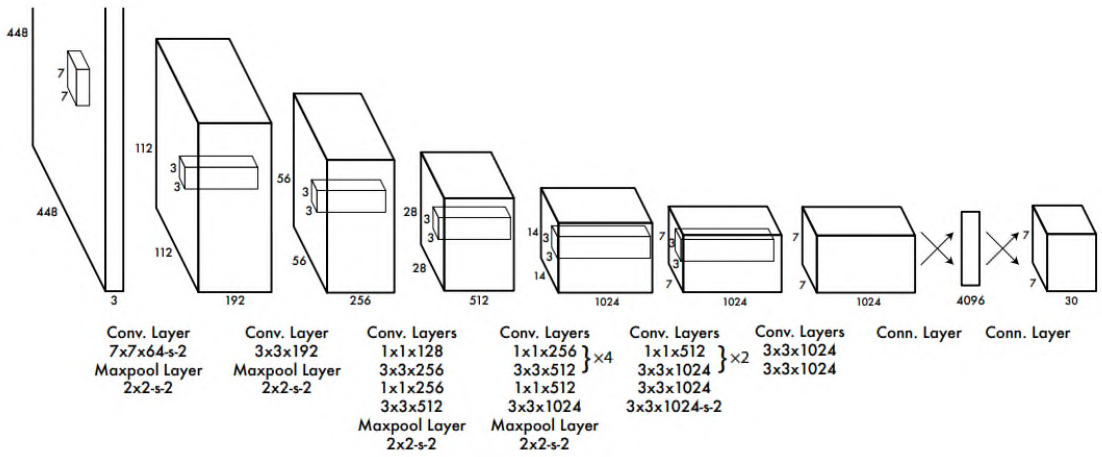

Fig 3.1: The YOLOv1 architecture for feature extraction, then reduces the layer and transforms the operation to forecast grid-wise.

There is a common problem that could be faced such as; multiple number of grids may be placed in many grids since YOLOv1 utilizes a 7 x 7 grid. The solution to this issue is called non-maximum suppression.

Since it is the beginning of the technological development for YOLO, as YOLOv1 is the oldest one and for this algorithm there is only one case to recognize the object's center is contained within a grid cell. YOLOv2 have improvements from YOLOv1 architecture on several topics that can be considered as huge differences for technological development of object detection history. As a normal condition for YOLOv1, it cannot identify every complete item or several items in a grid cell, nor can it identify small things. YOLOv2 aims to improve memory and localization while preserving classification accuracy in order to solve this issue. The changes can be listed as;

- anchor boxes
- batch normalization
- higher resolution classifier
- fine-grained features

- multi-scale training
- Darknet19.

Every grid cell has the ability to identify multiple objects by using the k bounding boxes. Using the training set bounding boxes, YOLOv2 applied the k-means clustering technique to determine the ideal set of anchor boxes based on the IoU measure. These values fall between 0 and 1 because it forecasts bounding box coordinates in relation to the grid cell's position. It forecasts four coordinates - t_x, t_y, t_w, t_h - for every bounding box. Given that the grid cell's upper left corner is (c_x, c_y) and the anchor box's dimensions are p_w and p_h. The YOLOv1 and YOLOv2 loss functions are the same. Different input picture sizes are used to carry out a multiscale training. For YOLOv2, Darknet-19 serves as the foundation.

YOLOv3 surpasses all prior iterations in terms of performance. It contains multi-scale detection, a more robust feature extractor network, and some modifications to the loss function compared to earlier versions. A multiscale feature, like a tractor, and a multiscale detector include two distinct parts of the network. Darknet-53 is the feature extractor in YOLOv3. YOLOv3 (Redmon&Farhadi,2018) gathers features from the final three residual blocks of the network as a multi-scale detector.

3.3.YOLOv4-YOLOv5-YOLOv7-YOLOv8

YOLOv5 is different from its predecessors, as it utilizes PyTorch instead of Darknet. Its main framework is CSPDarknet53, which integrates gradient changes into feature maps and resolves redundant gradient information in large backbones. This reduces inference performance, improves accuracy, and decreases model size by lowering the parameters. For enhanced information flow, YOLOv5 uses the Path Aggregation Network (PANet) as its neck. PANet employs a unique feature pyramid network (FPN) with multiple top-down and bottom-up layers, improving the propagation of low-level features and increasing localization accuracy in the lower layers.

Additionally, YOLOv5 shares the same head as YOLOv4 and YOLOv3, producing three distinct feature map outputs for multi-scale prediction. This enhances the model's efficiency in predicting objects of various sizes. The image is first processed by CSPDarknet53 for feature extraction and

then sent to PANet for feature fusion. The final results are generated by the YOLO layer. The design of the YOLOv5l algorithm is based on the YOLOv3 structure and introduces the Focus layer, which replaces the first three layers of YOLOv3 with a single layer in YOLOv5. Additionally, Conv refers to a convolution layer, and C3 consists of three convolution layers and a module cascaded by multiple bottlenecks.

Spatial pyramid pooling (SPP) is used to overcome the fixed size limitation of the network. The previous layer's fusion in the closest node is upsampled using the upsample layer. The Concat layer is used to slice the previous layer, and the final three Conv2d layers are detection modules used at the network's head.

Model architecture of legacy YoloV4 and Yolov7, and YoloV8 multi-classification models

The YOLO series of object detection models, which includes YOLOv4 and YOLOv7, is renowned for its quickness and accuracy in real-time object recognition [16]. Every successive iteration brings with it a number of enhancements over the previous one, and these developments have informed the creation of subsequent models such as YOLOv8. The key distinctions between YOLOv4 and YOLOv7 are outlined here, along with how these advancements impact the creation of YOLOv8.

YOLOv4

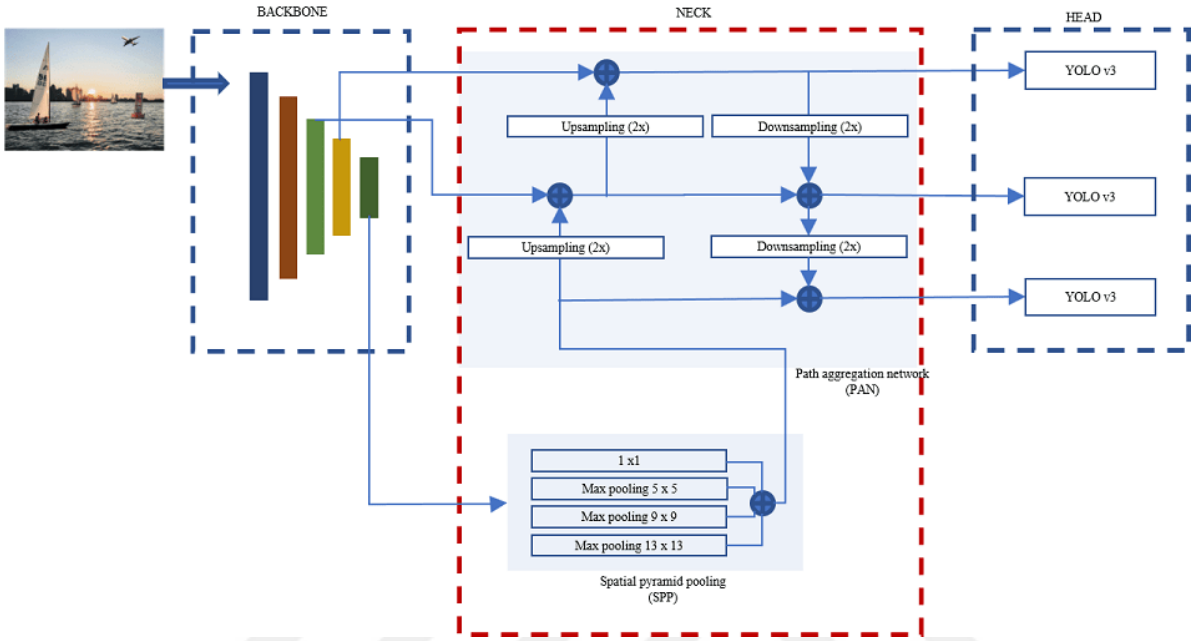

Fig 3.2: YOLOv4 in Synopsis and Principal Enhancements

In the YOLO architecture, there are main 3 parts. For the neck and backbone, CSPDarknet53 is the backbone of YOLOv4 is CSPDarknet53, which lowers computing costs and enhances learning capacity.

SPP layers increase the robustness of the model by assisting in the detection of objects at various sizes. Another development is, Path Aggregation Network, or PANet, is a technology used in the neck that improves information flow, which improves feature representation and object identification precision. A variety of training methodologies such as BoF including data augmentation, mosaic augmentation, and hyperparameter optimization were presented by YOLOv4. BoS uses methods to increase accuracy even further without lengthening the inference time. With its excellent speed-accuracy balance, YOLOv4 is deployed for real-time object identification tasks, especially in situations with limited resources.

With new ideas added to the YOLO series together with lessons learnt from earlier versions, YOLOv8 is an upgrade above YOLOv5. Following YOLOv4, YOLOv5 was developed as an expansion of the YOLO series. Because of its developer-friendly ecosystem, performance, and ease of use, it gained a lot of popularity. YOLOv8 expands upon YOLOv5 and YOLOv7 by including

fresh functionalities and cutting-edge machine learning techniques, such as enhanced architecture and improved training pipelines.

YOLOv7

With YOLOv7, real-time object identification accuracy is significantly increased without incurring higher inference costs. As demonstrated earlier in the benchmarks, YOLOv7 can efficiently accomplish quicker inference speed and greater detection accuracy, while cutting around 40% of parameters and 50% of the computation of state-of-the-art real-time object detections when compared to other known object detectors. In a general perspective, YOLOv7 offers a quick and resilient network architecture that enhances the speed of the label assignment and model training processes, improves the accuracy of object detection performance, and facilitates more efficient feature integration. Consequently, compared to other deep learning models, YOLOv7 requires computational gear that is several times less expensive. Without any pre-learned weights, it can be trained significantly more quickly on tiny datasets.

E-ELAN is by streamlining the way layers aggregate data, this architectural improvement strengthens the network's capacity for learning. Model Scaling in YOLOv7 brought effective scaling techniques that modify the network's depth, breadth, and resolution, enabling it to function effectively in a variety of processing settings, from high-end GPUs to mobile devices. With the introduction of novel methods like coarse-to-fine lead head, which enhances the performance of the head layers during training, YOLOv7 made substantial strides in training tactics. YOLOv7 outperformed earlier YOLO versions in precision and recall parameters, achieving state-of-the-art accuracy while retaining high speed.

As a general summary considering YOLOv4, YOLOv7 and connection these two to YOLOv8; to boost object identification performance, YOLOv4 included a strong backbone, sophisticated training algorithms, and feature aggregation approaches. With improved layer aggregation, scaling strategies, and more sophisticated training methodologies, YOLOv7 greatly improved the architecture and achieved state-of-the-art performance. With the connection and developments on these two YOLO algorithms, YOLOv8 expands upon these developments by combining and improving upon the most effective elements of YOLOv4 and YOLOv7 to provide an object detection model that is stronger, more adaptable, and more effective. YOLOv8's architecture and

performance have been significantly impacted by the ongoing advancements from YOLOv4 to YOLOv7, which has made it an excellent option for real-time object identification jobs in a variety of contexts.

YOLOv8

The most recent version of the well-liked object identification system, YOLOv8, exhibits remarkable results in a number of criteria. Notable modifications to the neural network design are made by YOLOv8. The model is an attractive option for a variety of computer vision jobs because of its higher accuracy and speed, which are achieved with optimized structures.

Improvements to YOLOv8's training procedure lead to better convergence and quicker training periods. The model is more effective at adjusting to various datasets and settings as a result of these modifications. There are several variations of YOLOv8 (YOLOv8-S, YOLOv8-M, YOLOv8-L, and YOLOv8-X), each designed to meet certain needs. With the variety of options for these variations provide, customers may select the model that best suits their application requirements and available computing power.

One of the most important point to mention which will be used as a common detection method as mAP; YOLOv8 obtains mAP values ranging from 0.685 to 0.835 on COCO, a well-known object identification benchmark. It depends on the model variation and dataset. This is a quite scaling metric that, this thesis will compare the mAP scores of YOLOs as in YOLOv5 and YOLOv8.

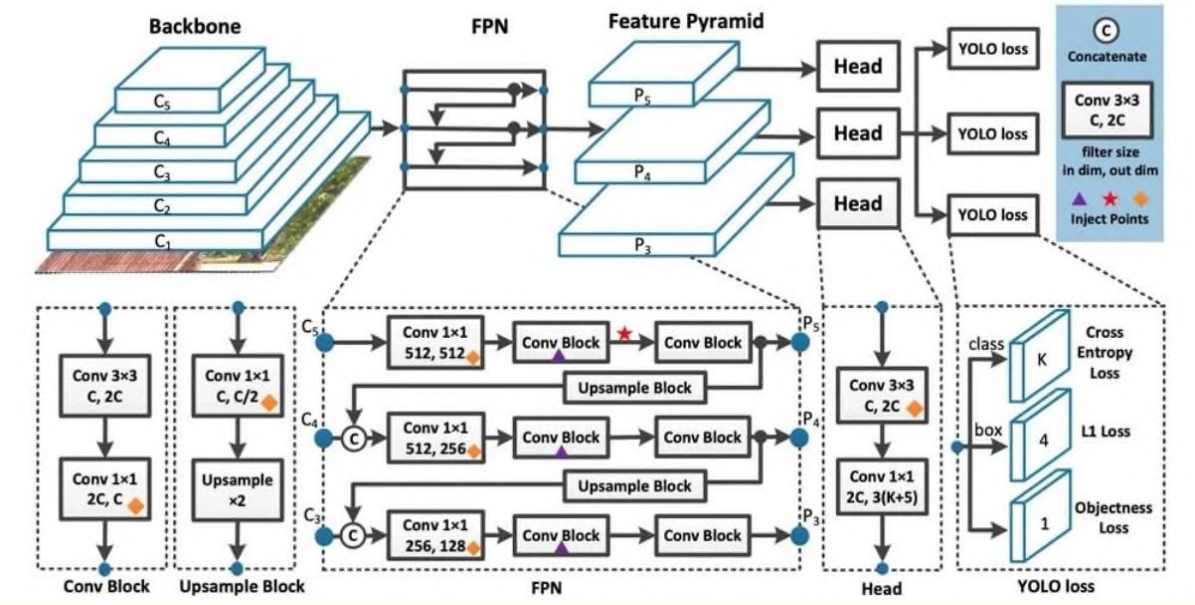

Fig 3.3: Transformation of convolutional computation in YOLOv8

It is useful to compare YOLOv5 to other models as its proven performance, versatility, and ease of use make it a solid benchmark. This comparison aids in demonstrating the developments and potency of new models in the field of object detection.

Because of its ability to balance speed and accuracy, YOLOv5 is extensively used in the object detection field and is frequently used as a benchmark for future models. When YOLOv5 was first released, it offered cutting-edge performance, setting a high bar that later models had to either equal or surpass. The user-friendliness of YOLOv5 is prioritized in its design, which includes intuitive codebases, clear documentation, and strong community support. This enables benchmarking against new models to be possible.

The popularity of YOLOv5 paved the way for YOLOv8. The strengths and limits shown in YOLOv5 provided as inspiration for the advancements in YOLOv8, which included improved performance on smaller models, improved architecture, and adaptability for various tasks (e.g., object recognition, segmentation, and classification). In many aspects (including model export formats and some processes), YOLOv8 is backward-compatible with YOLOv5, which facilitates the upgrading process for developers who want to go from YOLOv5 to YOLOv8. Thanks to improvements in its design, YOLOv8 performs better than YOLOv5 in terms of accuracy, speed,

and model flexibility. However, because YOLOv5 has been in operational environments for a longer period of time, it still leads in several aspects, such community support, maturity, and stability for particular use cases.

3.4. R-CNN

Here, considering CNN case, this is the combined case for regions. Deep network is used in detail based on specific details; color, texture, shape similarity. The R-CNN architecture has two stages as; region proposal and classification. From these points, it creates boxes around the objects that are being detected with a huge selective search. As the last steps, it follows RoI with SVM. Each class needs single SVM that is used for classifying each region. As considered, there are nearly 2000 regions detected for each image in order not to miss any detail that necessity to be examined through object detection process. In order to estimate the bounding box's center coordinates, width, and height during training, R-CNN appends a bounding box regression to the region proposal.

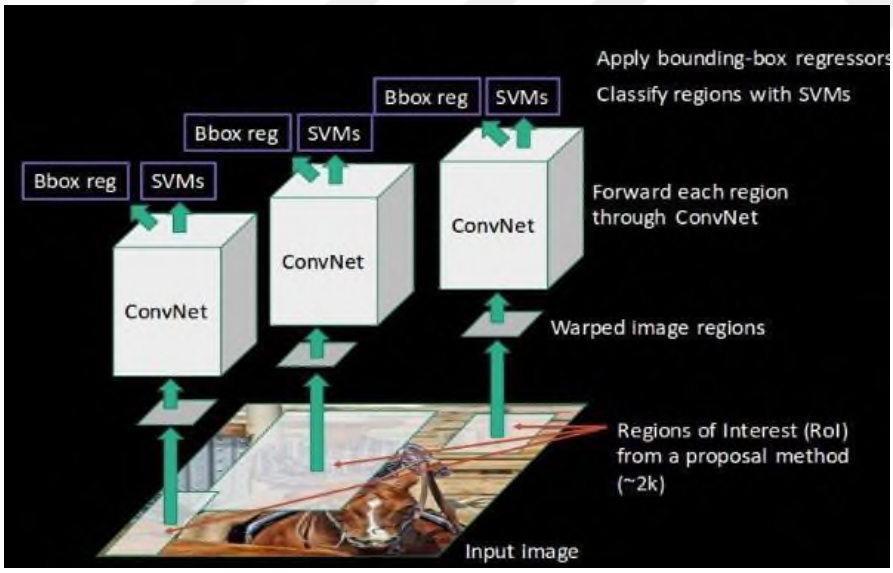

Fig 3.4: Method appliance for image with Region of Interests containing SVM's

R-CNN propose many regions where objects may be found and then pass into the convolution layer for class and bounding box prediction.

Drawback case is definitely with the region detection and memory case. Of course, for detection the training and testing stage consists of numbers of images. As the number of image increases to illustrate 1000, the number of region detected will be multiplied with the number of image that is being used. 1000*2000 is going to be a huge number as it slows down the method quite much.

One-stage detection is used in RetinaNet. Resent is used as the backbone for feature extraction since it allows us to access deeper network topologies. Two distinct subnetworks are utilized in tandem for regression and classification. Focus loss was included, and this enhanced the forecast. The issue of class imbalance is addressed by focal loss, particularly in one stage detectors. Compared to the number of foreground classes or positive models, the number of background classes or negative instances is significantly larger. The cross-entropy loss significantly raises the loss values for uncommon types when averaged across several negative samples because of this class imbalance. A modified cross entropy loss is called focal loss. It further reduces the loss of well-classified example loss, allowing our model to concentrate more on

Focal Loss, which adds a factor $(1-p\ t)\ \gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples, putting more focus on hard, misclassified examples. Focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples. The highest accuracy object detectors to date are based on a two-stage approach popularized by R-CNN, where a classifier is applied to a sparse set of candidate object locations. One-stage detectors applied over a regular, dense sampling of possible object locations have the potential to be faster and simpler but have trailed the accuracy of two-stage detectors thus far. The authors propose to address this class imbalance by reshaping the standard cross entropy loss so that it down-weights the loss assigned to well-classified examples. The novel focal loss focuses training on a sparse set of hard examples and prevents the vast number of easy negatives from overwhelming the detector during training.

3.5. COCO Dataset Attributes

The model is trained to predict the class and the bounding box in a single pass using a single head model. The cross-entropy kind of loss function is appropriate for classification, whereas regression often refers to square error loss. All losses were regarded as mse losses with a regularized parameter in the YOLOV1 study. It seems excessive to expect a single model to provide an accurate outcome

for every function. Rather than making predictions straight from the extraction of tensor multi-head features, parallel computes the bounding box and the class inside it independently, referred to as the classification head and regression head. Since the classification head's loss function differs from the regression head's, model learning becomes easy.

The dataset that’s been created including own taken photos inside the bus. The buses overhead luggage part, there is a place which people put their stuff as laptop, backpack, suitcase etc. The dataset also contains these samples taken from real-life, the objects are put as a real-life objects and samples as well. In part 5.1, Camera configurations and Hardware setup part, the detail for camera setup and how the videos are being recorded hardware part explained in detail. The dataset is being created both in my objective and created dataset.

The dataset in the beginning is observed with COCO since it has over 328K image samples that has a huge community in YOLO object detection algorithm. For YOLOv5 and YOLOv8 which two algorithms are going to be examined through this thesis, has already a COCO dataset with number of classes 80. In base it is always given as;

nc=80

names: ['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light',
'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple',
'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch',
'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cellphone',
'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear',
'hair drier', 'toothbrush']

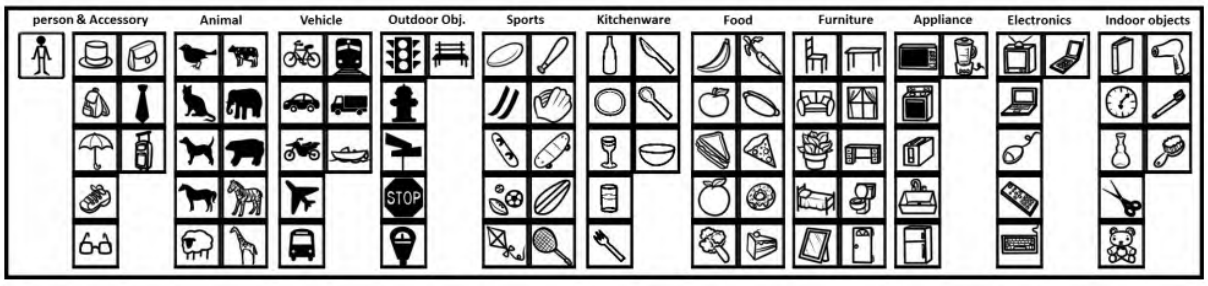

Fig 3.5: COCO Dataset is grouped into 11 categories and 91 categories are below listed

nc is number of classes and names are in detail the name of classes all written down. For COCO dataset; the train, test and validation folders would be written as this and the number of image samples are for %32 train, %13,6 validation and %54,3 test given as main number of samples in COCO. Here, the used number of percentages are the same; 70% test, 20% train and 10% validation.

```
# COCO 2017 dataset http://cocodataset.org

# download command/URL (optional)
download: bash ./scripts/get_coco.sh

# train and val data as 1) directory: path/images/, 2) file: path/images.txt, or 3) list: [path1/images/, path2/images/]
train: ./coco/train2017.txt  # 118287 images
val: ./coco/val2017.txt  # 5000 images
test: ./coco/test-dev2017.txt  # 20288 of 40670 images, submit to https://competitions.codalab.org/competitions/20794

# number of classes
nc: 80
```

Fig 3.6: COCO dataset is seen as sample for number of images inside them

Here, custom dataset is done for limited number of object detections. Since 80 classes of detection is not needed for the samples taken inside the bus, so the dataset is decreased to 7. These objects are classified as “Backpack”, “umbrella”, “book”, “glasses”, “laptop”, “luggage” and “phone”. The annotation instructions for the dataset gives a better proportion and were asked to minimize the border box sizes. Bigger bounding boxes were not preferred over smaller ones since they removed parts of the item that were not needed, like the ends. The human boundary boxes included no personal items. Only the parts of the item that were not obscured were displayed when occlusion occurred. When training ML models, especially in the context of deep learning, data augmentation

and preparation are closely related processes. These two steps do not concentrate on separate areas, they both help to raise the caliber and efficacy of the training process.

To deep down into these stages, first step is Data Preparation. The more comprehensive process of preparing your raw data for use in ML models is known as data preparation. There are multiple steps such as;

Data collection which is compiling information from several sources. Data cleaning includes addressing missing numbers, eliminating or fixing errors, and handling noisy data. Data transformation involves transforming features into a format that is appropriate for training a model by normalizing, scaling, or encoding them. Splitting the data into test, validation, and training sets is known as data splitting. Feature engineering involves developing fresh features or altering current ones to more effectively reveal underlying trends in the data. For the data enrichment process, a subset of data preparation known as data augmentation's main point is to modify the available data in order to artificially increase the diversity of the training set. It usually becomes relevant when the first stages of data preparation are finished. Data augmentation include rotating a picture a few degrees or inserting noise to strengthen the model. In the sequential process, augmenting data is frequently a phase in the larger process of preparing data. More diversified training data can be produced by using augmentation techniques after the data has been cleaned and processed.

In a general consideration, data augmentation and preparation go together and quite linked to each other. While data augmentation increases the diversity of the dataset and strengthens the model's ability to generalize to new data, data preparation guarantees the accuracy and usability of the data.

Here, custom dataset is done for limited number of object detections. Since 80 classes of detection is not needed for the samples taken inside the bus, so the dataset is decreased to 7. Backpack, umbrella, book, glasses, laptop, luggage and phone. The annotation instructions for the dataset gives a better proportion and were asked to minimize the border box sizes. Bigger bounding boxes were not preferred over smaller ones since they removed parts of the item that were not needed, like the ends. The human boundary boxes included no personal items. Only the parts of the item that were not obscured were displayed when occlusion occurred. The model is trained to predict the class and the bounding box in a single pass using a single head model. The cross-entropy kind

of loss function is appropriate for classification, whereas regression often refers to square error loss.

For Colab Library's loss function, when utilizing Ultralytics YOLOv5 or YOLOv8 in Google Colab, the loss function is preset inside the library itself. Right out of the box, the framework optimizes for object detection jobs. Here is its loss function,

$$\text{Loss}_{\text{classification}} = \sum_{i=1}^{C} \text{yi} \log(\hat{\text{y}}\text{i}) \quad (1)$$

y_i is the ground truth label (1 if the object belongs to class i, otherwise 0).
$\hat{y}_i$ is the predicted probability for class i.

$$\text{Loss}_{\text{regression}}(x, y) = \begin{cases} 0.5(x-y)^2, & if\ |x-y| < 1 \\ |x-y| - 0.5, & otherwise \end{cases} \quad (2)$$

The portion of the loss function that penalizes mistakes in the bounding boxes' anticipated center coordinates *(x, y)* is denoted as *(x, y)* which is indicated by these coordinates.
In the YOLOv5 and YOLOv8 Ultralytics models, the loss function consists of a mix of Bounding box localization loss, loss of Confidence (Objectivity) and loss of Classification. These are implemented automatically when the training code has been running and are preconfigured and tuned to perform tasks like object detection. The model training pipeline takes care of its computation and optimization.
The loss function for the regression head, which forecasts the bounding box coordinates, which is named MSE. All losses were regarded as MSE losses with a regularized parameter in the YOLOv1 study. It seems excessive to expect a single model to provide an accurate outcome for every function. Rather than making predictions straight from the extraction of tensor multi-head features, parallel computes the bounding box and the class inside it independently, referred to as the classification head and regression head. Since the classification head's loss function differs from the regression head's, model learning becomes easy.

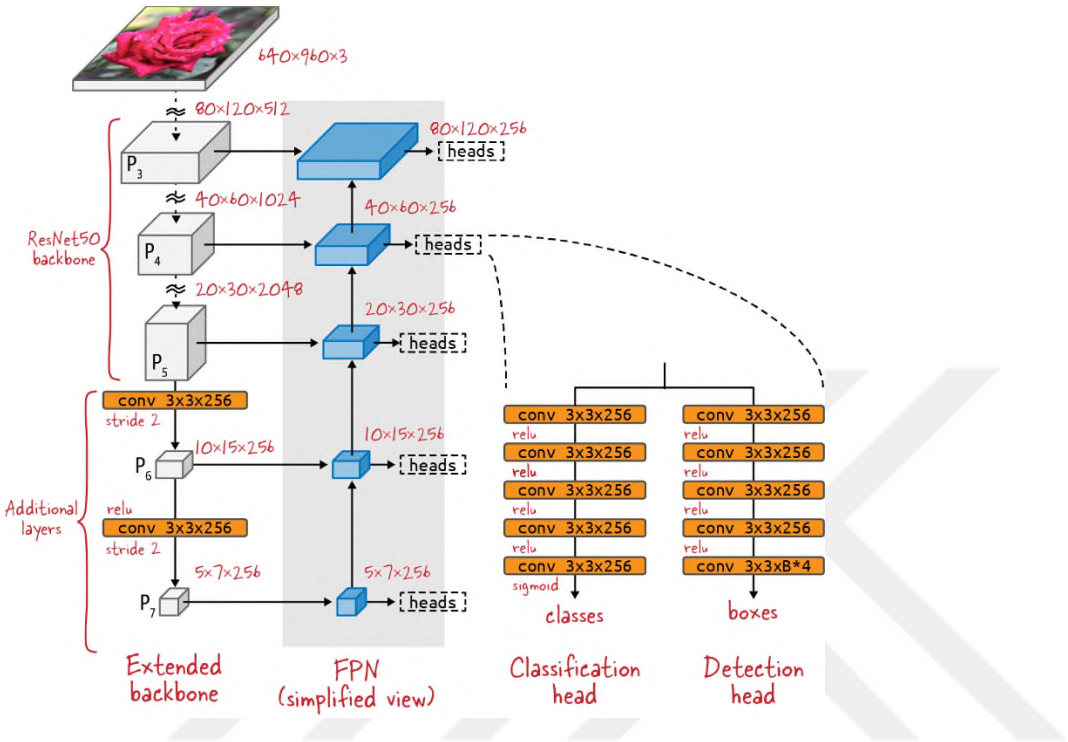

Fig 3.7: Classification and detection head YOLO

Detecting small objects is the main aim and action in this thesis. The goal is to employ cameras to identify forgotten things from passengers on coach bus shelves as they disembark and to alert the driver to any items that may have been neglected. The dataset that's been created including own taken photos inside the bus. The buses overhead luggage part, there is a place where passengers put their stuff as laptop, backpack, suitcase etc. The dataset also contains these samples taken from real-life, the objects are put as a real-life objects and samples as well. In part 5.1, Camera configurations and Hardware setup part, the detail for camera setup and how the videos are being recorded hardware part explained in detail. The dataset is being created both in my objective and created dataset.

3.6. Data Preparation

There were also samples in the dataset that one image has 3 objects inside in order to be trained better. The dataset has 1799 images inside, has divided with a known division percentage, training 80%, testing 10% and validation as 10%. All the images were annotated properly with a wider angle to be able to detect background properly. The images were preprocessed with these steps; auto-orientation, static-crop, resize and grayscale. These preprocessing steps are done in order to the dataset to be trained in harder cases and make a higher percentage to be known while the object is 90^{o} vertical or horizontal, has gray colors or stretched as well.

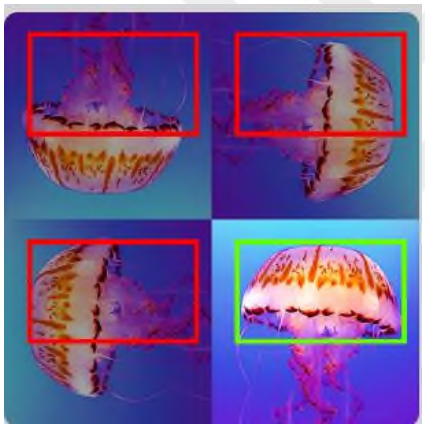

Fig 3.8: Auto-orientation for preprocessing scale

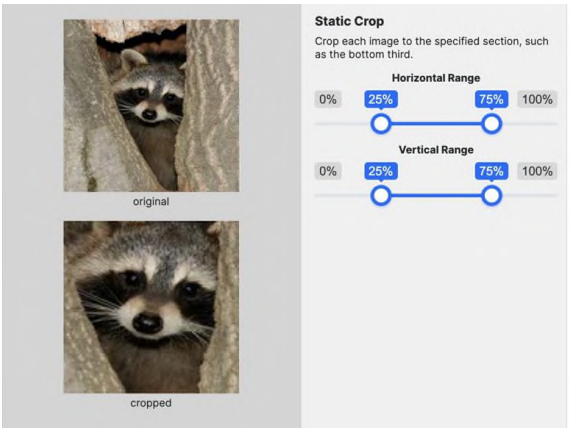

Fig 3.9: Static crop for preprocessing scale

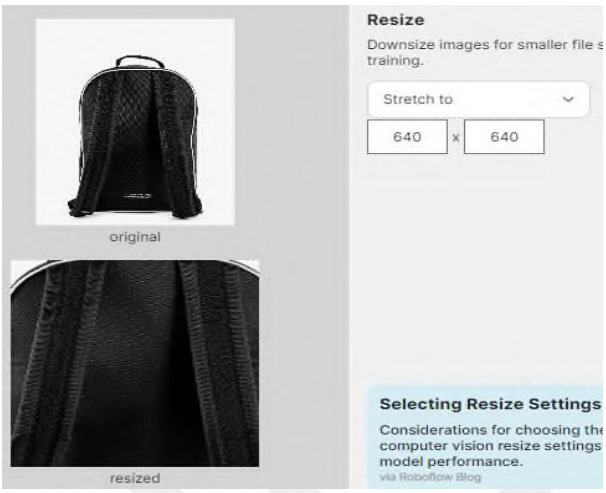

Fig 3.10: Resizing all for preprocessing scale

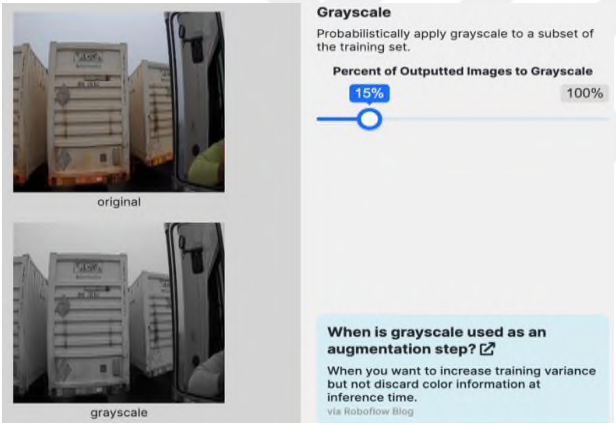

Fig 3.11: Grayscale for preprocessing scale

3.6.1. Data Augmentation

For the COCO dataset or considering the newly created dataset, data augmentation is a quite efficient technique that expands the size of the training dataset without requiring the collection of additional real-world data. The main way is by creating variants from the available data. The initial dataset was small as 1799 images. This dataset is utterly separate from COCO; has been created with the mix samples from Roboflow + samples inside the bus and sums as 1799. This dataset has

been divided into exact percentage of test train and validation as used in COCO dataset to make a valid comparison. The augmented one is 4159.

In the differential transformations; rotation, flipping, scaling, cropping, color adjustments, and noise addition are common image augmentation techniques which are also used in newly created dataset. Several versions of an image of an umbrella by flipping it horizontally, varying its brightness, and rotating it slightly so that it can be defined by in any version that has been placed. In order to improve the performance of the model, data augmentation is done. This is also a method for the model to be working better for real-life scenario and prepares better for harder real cases as well. To explain this in more detail, adding noise, cropping makes it even harder for model to recognize in all cases. When it is trained with these difficulties, the model makes it higher percentage of understanding of the model. For this dataset's final reached out samples as in the number of 4159; flip, crop, grayscale, noise and bounding box of 90^{o} rotate is applied as augmentation steps.

For the data augmentation, the problem with COCO dataset faced is one of the objects that needs to be detected was umbrella. In COCO dataset, the umbrella was all open in the whole dataset, so that it couldn't be used in this specified detection. Another issue was also the small objects that used in this thesis. The COCO dataset has all the objects in big objects that can be detected easily, however in the dataset that's been taken samples from inside the bus were small. Such as book, glasses, umbrella, laptop, cellphone. These objects are small compared to detection of small objects in COCO dataset object detection data. These two main problems lead to creation of a new dataset. In order to improve the performance of the model, data augmentation is done. This is also a method for the model to be working better for real-life scenario and prepares better for harder real cases as well. To explain this in more detail, adding noise, cropping makes it even harder for model to recognize in all cases. When it is trained with these difficulties, the model makes it higher percentage of understanding of the model. For this dataset, flip, crop, grayscale, noise and bounding box of 90^{o} rotate is applied as augmentation steps.

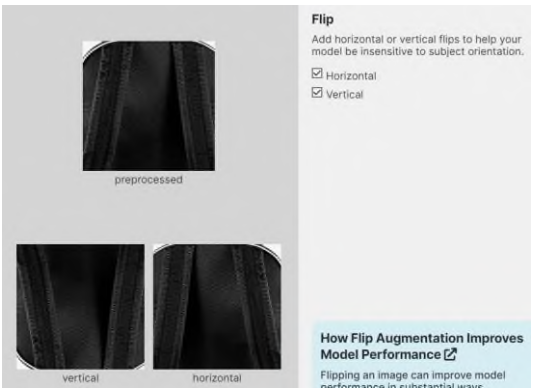

Fig 3.12: Flip in horizontal and vertical

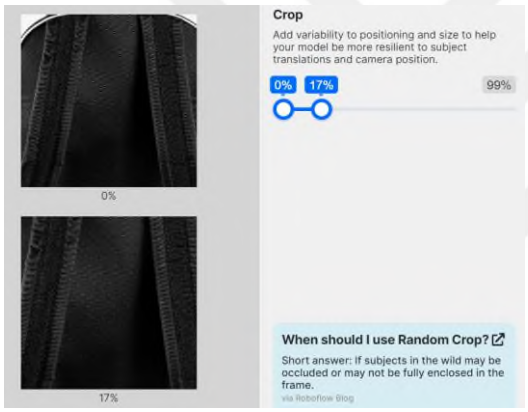

Fig 3.13: Crop is done for augmentation with 17%

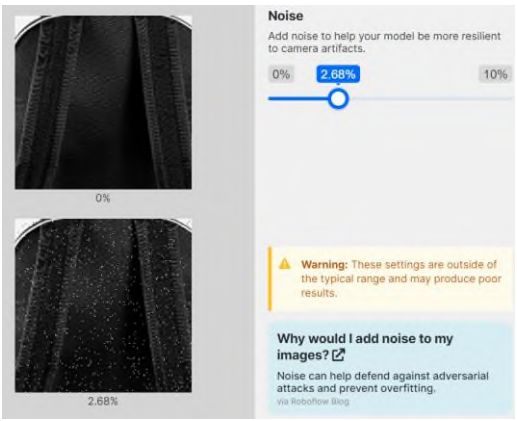

Fig 3.14: Noise is added as 2.68 on 10% for augmentation

In the data augmentation part, the parts that's been applied as;

- 25-75% Horizontal Region, 25-75% Vertical Region static crop operation,
- Flip: Horizontal
- Crop: 0% Minimum Zoom, 20% Maximum Zoom
- Grayscale: Apply to 15% of images
- Noise: Up to 1.33% of pixels
- Bounding Box: 90° Rotate: Clockwise, Counter-Clockwise

The model now can easily predict the mixed object probability for an image rather than a single class. It undoubtedly aids the model's exploration of the images' more intricate channel-wise properties.

Table 3.1: COCO dataset number of objects, image numbers regarding newly created dataset and data augmented images

Class	**Backpack**	**Closed Umbrella**	**Book**	**Glasses**	**Laptop**	**Luggage**	**Phone**
Coco dataset number of objects	5756	0	5562	12604	3707	2507	5017
Number of Images added to the dataset	602	526	684	702	710	624	649
Data Augmented Images	1867	1712	1973	1882	2019	1904	2045

3.7.Camera Configuration and Hardware Setup

In this project, HIK-Vision branded 2 fish-eye cameras will be employed. There will be two active cameras since they have a superb view for detecting items. Two will be on one side and the other two will be just behind these two. There is a location that a bus line can detect. The region that was

picked up by both cameras will be highlighted in yellow, and their combined image will be highlighted. The two cameras' locations will be shown below:

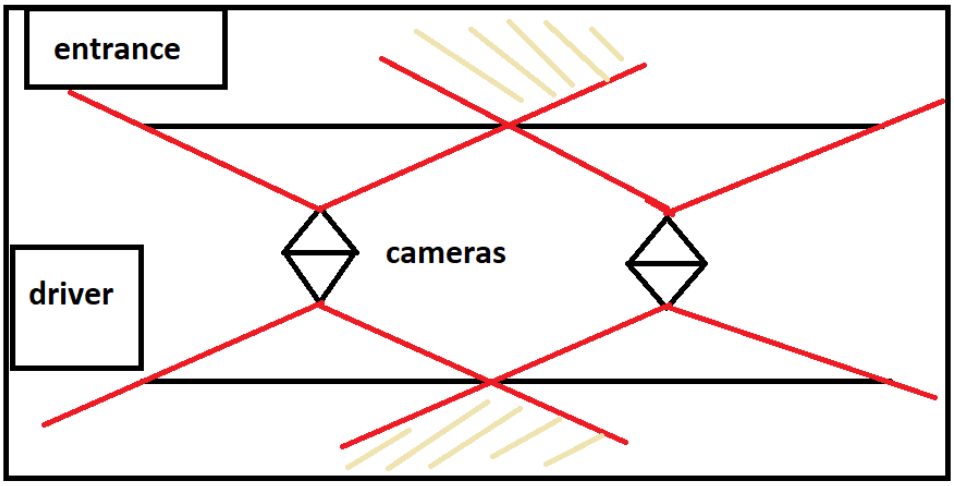

Fig 3.15: Bus inside view 2 cameras places with their vision involved

Here, the bus contains 2 cameras inside them, they have vision with red colors. The yellow paintings in two reciprocal directions are the intersection areas. In this case, in order to obtain two different images into one, image stitching method will be operated here. The objects mentioned will be trained and tested before validating them.

Data Source is the cameras will capture the required data from within the bus, including passenger counts and luggage status. Data Processing is; a computer will process the captured data using image processing algorithms to perform required checks and generate relevant alerts. Data Display is; the processed information will be displayed on a driver screen in a clear and intuitive manner, enabling the driver to take prompt actions as needed.

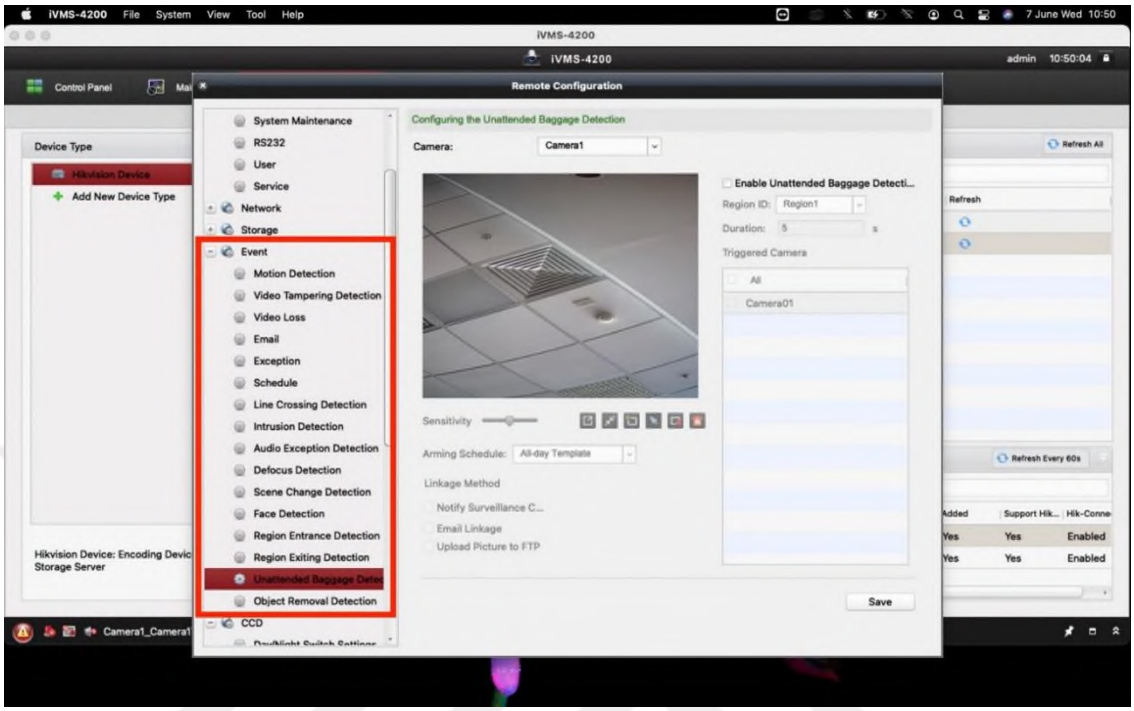

Fig 3.16: IVMS-4200 configuration page for unattended baggage detection

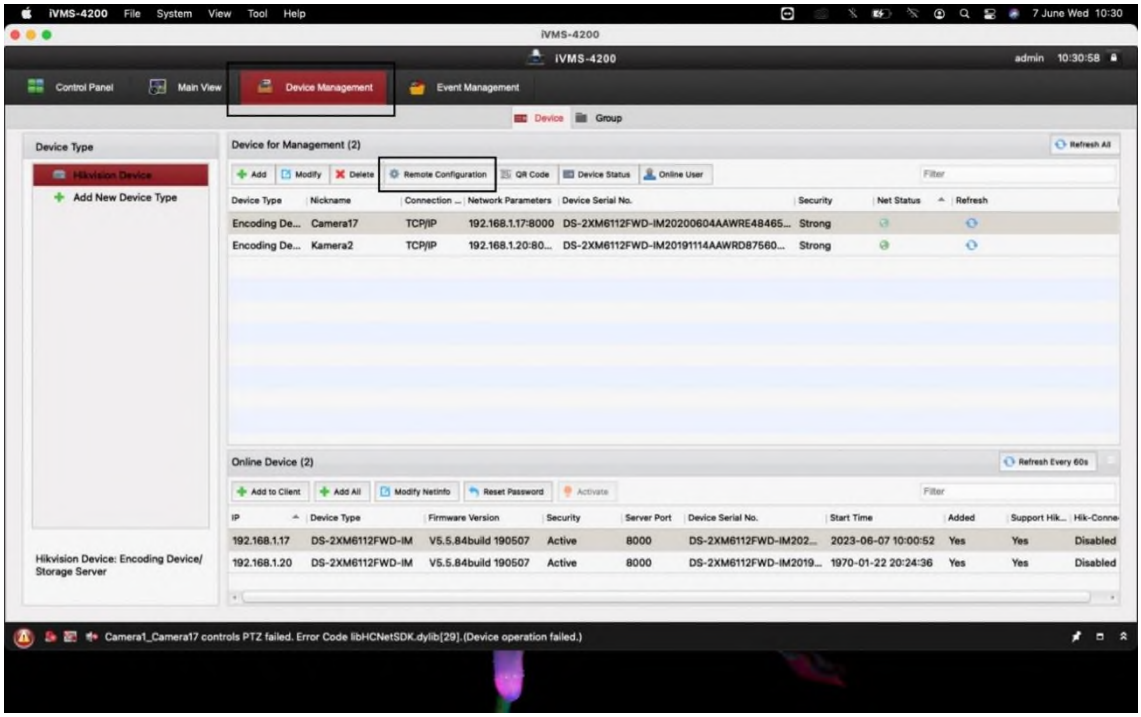

Fig 3.17: 2 camera configuration screen

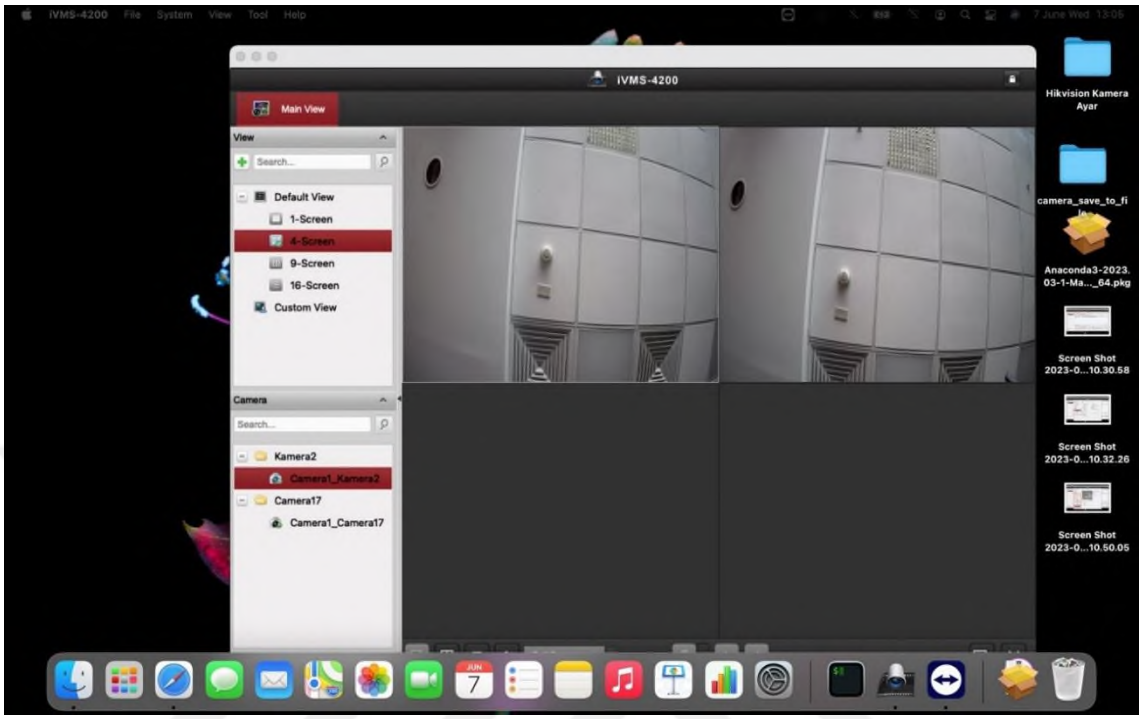

Fig 3.18: 2 cameras in display mode, clearly seen simultaneously

In figure 25-26-27; the IVMS-4200 interface is used for the camera system.

Here, system can be seen in device management with remote configuration events.

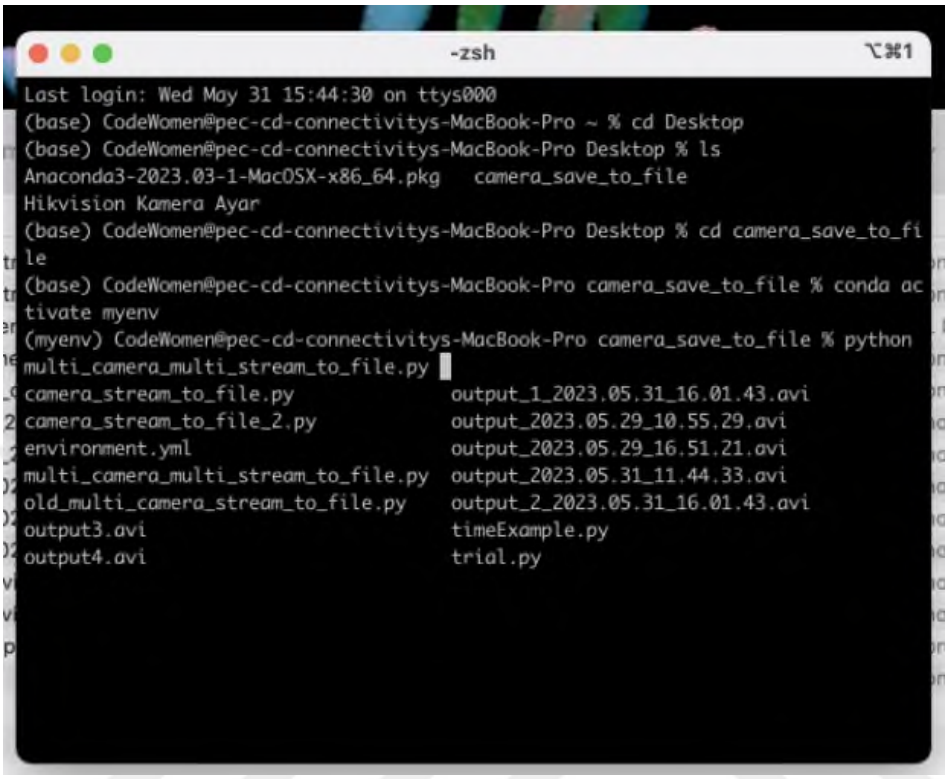

Fig 3.19: Base to myenv environment created in aim to use anaconda environment

The configuration needs to be settled with another environment creation as there are some reasons. The very first is that, anaconda environment is much simpler to use, whenever you want to delete something or add something it is quite safe. Second, downloading python and opencv was a must in this project, however it lead to downloading major ones additional; such as numpy, pip and others. Also, when two projects need to be run at the same time, the problem for different versions would create a big problem; now in this myenv, it won't be faced anymore.

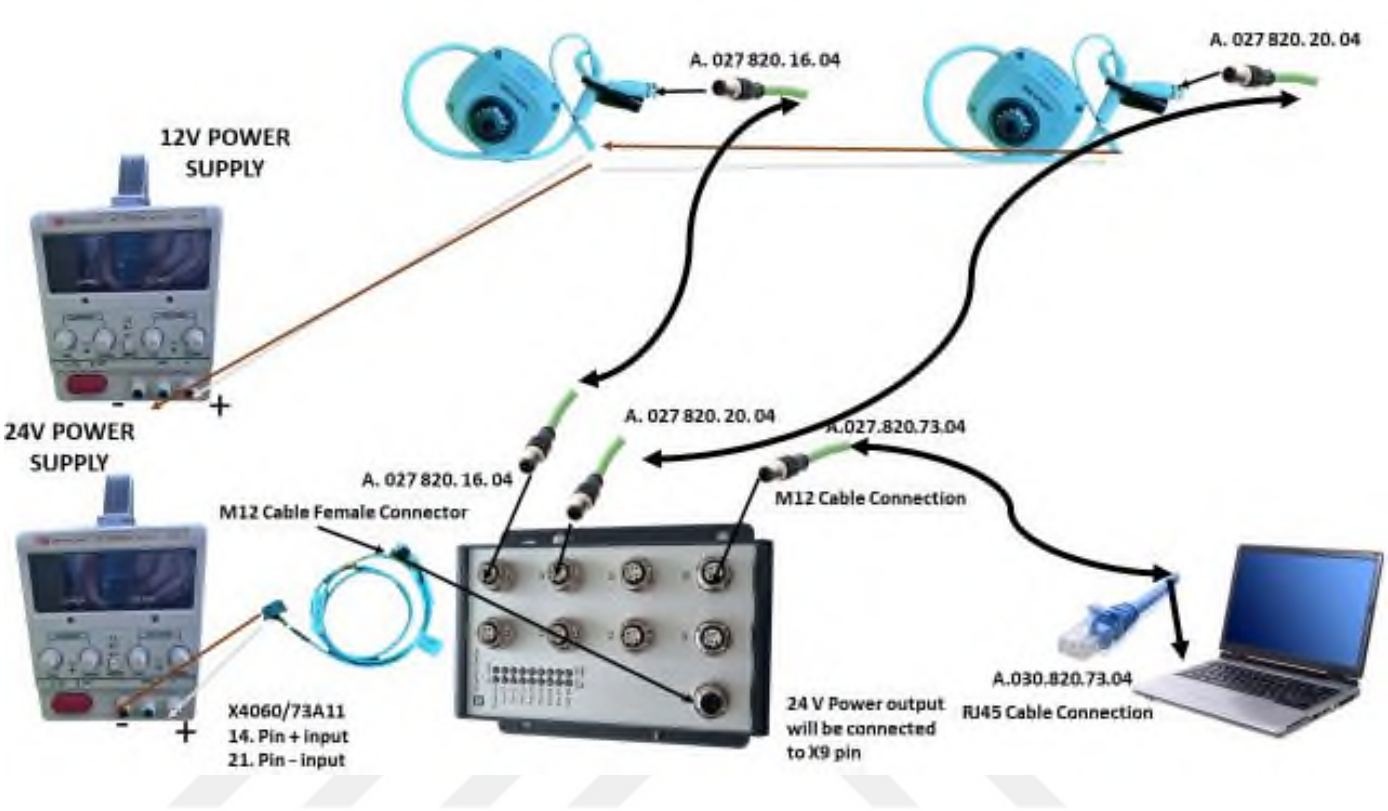

Fig 3.20: Hardware setup for this schemes as connected with power supplies and computer; all connected to Ethernet switch

In the hardware system;

2 x HIK Vision camera with following IPs:

192.168.1.17

192.168.1.20

1 x Network Switch

2 x Network Switch power cables

2 x Camera IP Video Cable

1 x Ethernet to IP cable

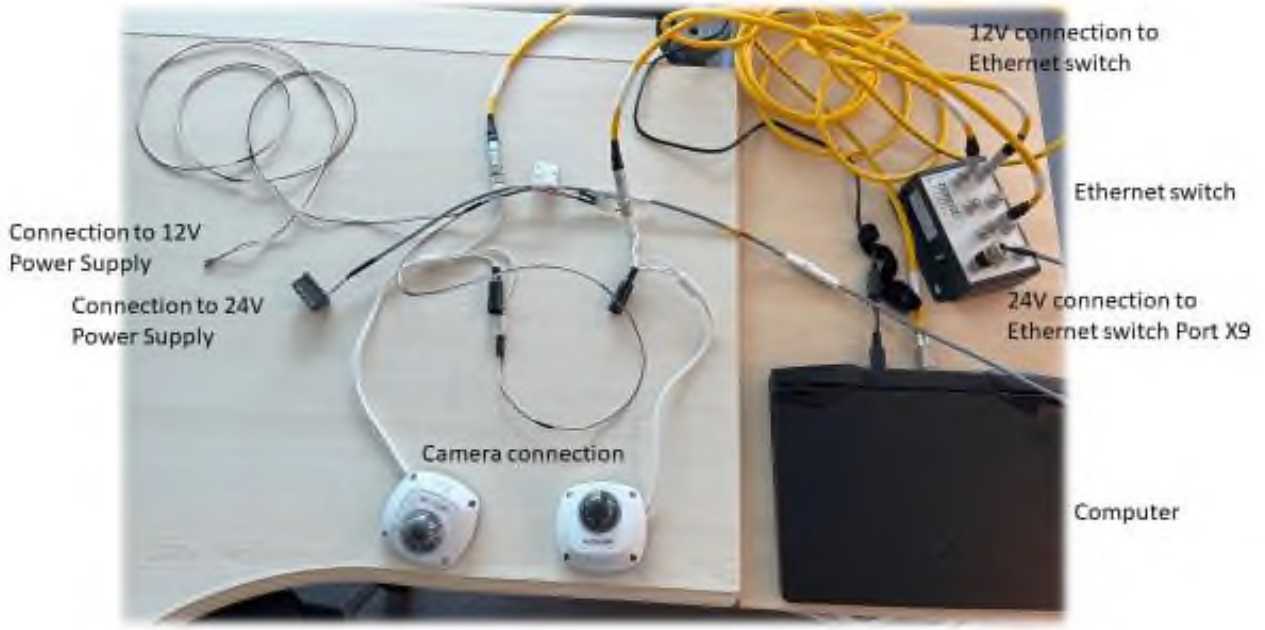

Fig 3.21: Hardware setup

This is the complete setup done in the table, is ready for setting up inside the bus. There are 2 cameras, and they need to be performed while in the record action. The python code for video streaming with two cameras:

3.8. Implementation

3.8.1. Building of the Model

There is always a backbone and head structure in every YOLO, having detailed module names and arguments inside them as well. There are multiple modules such as Conv, DWConv, DWConvTranspose2d, C3, Bottleneck, CrossConv etc. These modules are being decided whether the model will have better backbone system so that has much higher percentages for detecting the outputs.

```yaml
backbone:
  # [from, number, module, args]
  [
    [-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
    [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
    [-1, 3, C3, [128]],
    [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
    [-1, 6, C3, [256]],
    [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
    [-1, 9, C3, [512]],
    [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
    [-1, 3, C3, [1024]],
    [-1, 1, SPPF, [1024, 5]], # 9
  ]
```

Fig 3.22: The most used backbone architecture for YOLOv5 in module names

```yaml
# YOLOv5 v6.0 head
head: [
    [-1, 1, Conv, [512, 1, 1]],
    [-1, 1, nn.Upsample, [None, 2, "nearest"]],
    [[-1, 6], 1, Concat, [1]], # cat backbone P4
    [-1, 3, C3, [512, False]], # 13

    [-1, 1, Conv, [256, 1, 1]],
    [-1, 1, nn.Upsample, [None, 2, "nearest"]],
    [[-1, 4], 1, Concat, [1]], # cat backbone P3
    [-1, 3, C3, [256, False]], # 17 (P3/8-small)

    [-1, 1, Conv, [256, 3, 2]],
    [[-1, 14], 1, Concat, [1]], # cat head P4
    [-1, 3, C3, [512, False]], # 20 (P4/16-medium)

    [-1, 1, Conv, [512, 3, 2]],
    [[-1, 10], 1, Concat, [1]], # cat head P5
    [-1, 3, C3, [1024, False]], # 23 (P5/32-large)

    [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
  ]
```

Fig 3.23: Head structure of YOLOv5

New network will be created and it will be trained then inferenced. A new attention convolution layer is being create in order to effect and improve the algorithm's speed. The way that algorithm is seen with a way that has a room to be improved is that improving the backbone.

3.8.2. Attention Convolution Layer

This layer is being done with a convolution layer's improvement.

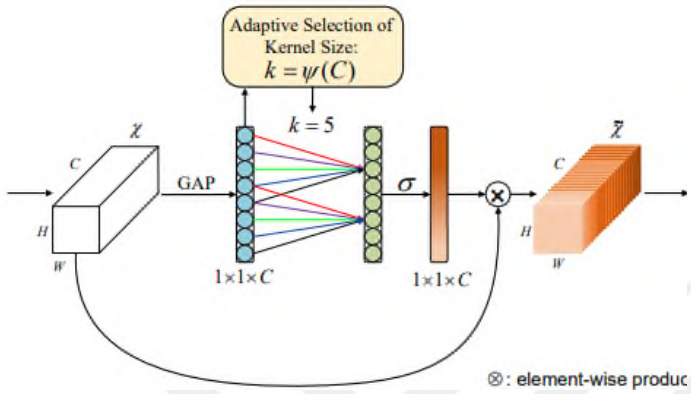

Fig 3.24: ECA module diagram

Detection Layer for YOLOv5

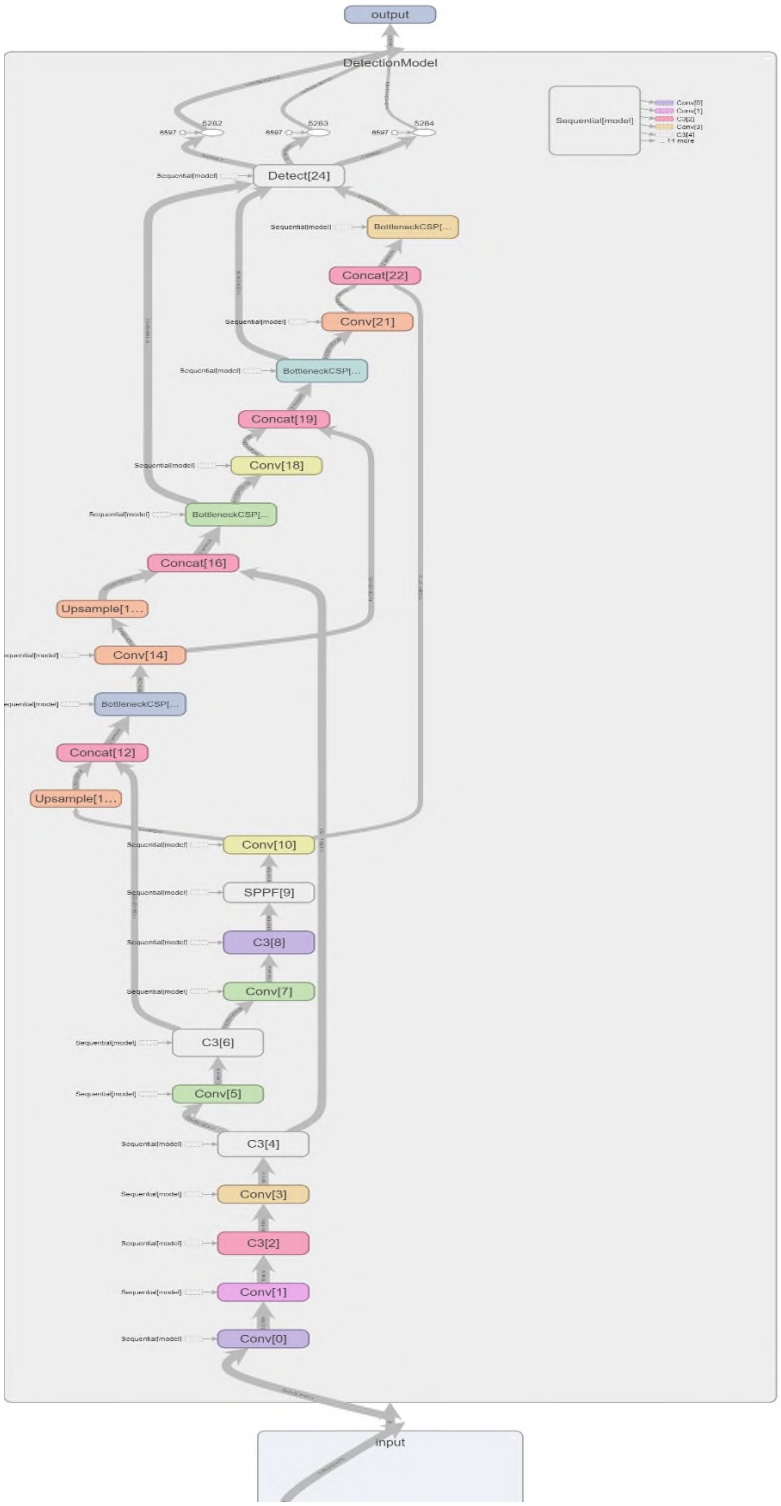

Fig 3.25: YOLOv5 Tensorboard visualization of Detection layer

Detection Layer for YOLOv8

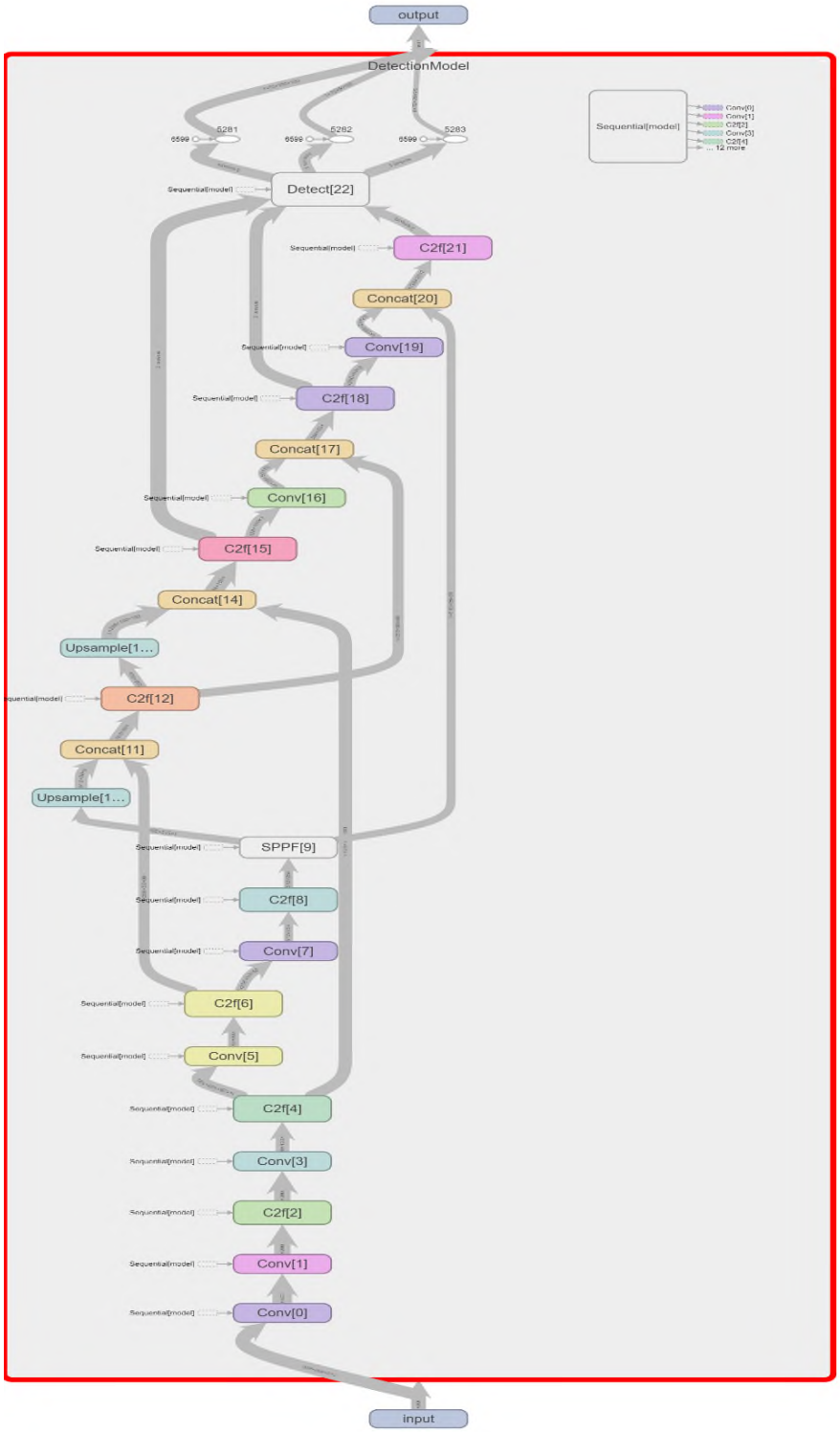

Fig 3.26: YOLOv5 Tensorboard visualization of Detection layer

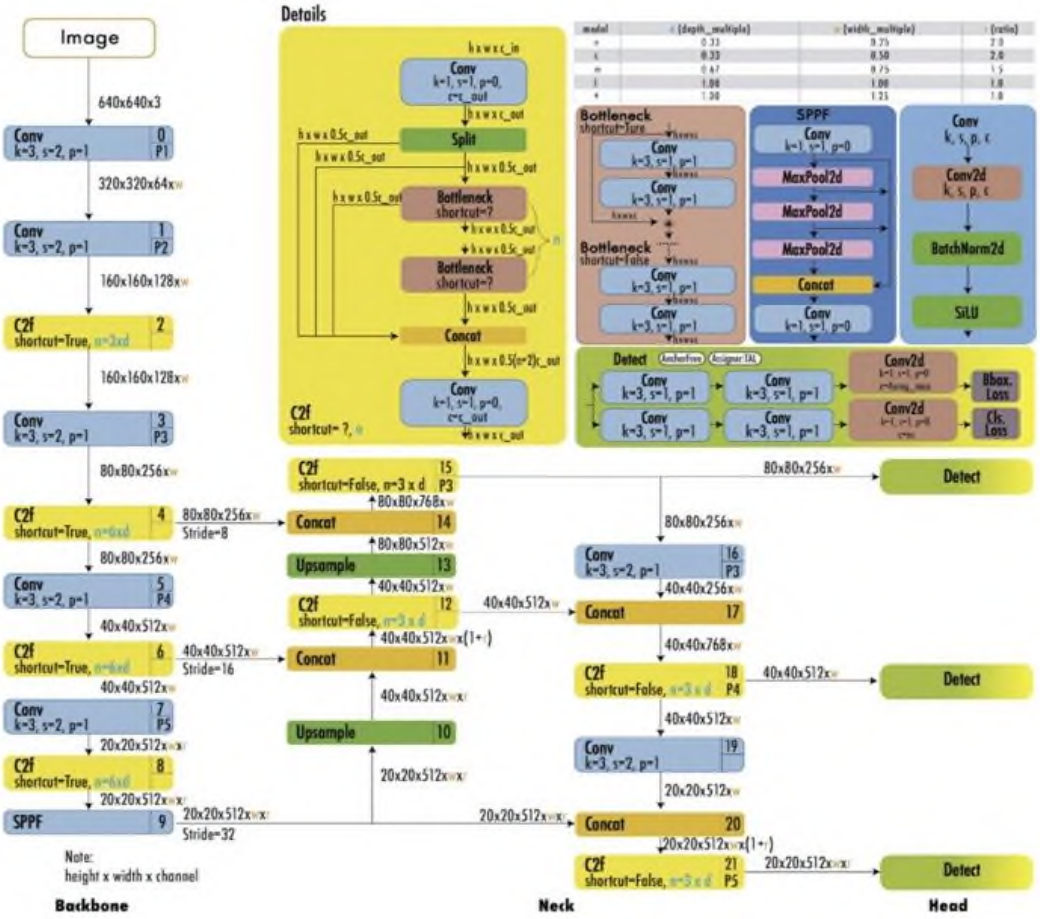

Fig 3.27: YOLOv8 network structure with backbone, neck and head

3.9. Implementation

3.9.1. Efficient Channel Attention Module

In this part, instead of using a standard convolution module, the new Attention Convolution module is introduced. There are new features introduced such as; AdaptiveAvgPool2d layer, an ECA module, and a linear transformation layer. With this newly assigned layer, there are some new techniques and improvements are imported. As the layer enables the optional batch normalization and activation of a normal convolutional layer upon initialization; it remarks a forward pass technique using activation function, batch normalization, and convolution. It applies a forward pass method for activation function and fused convolution.

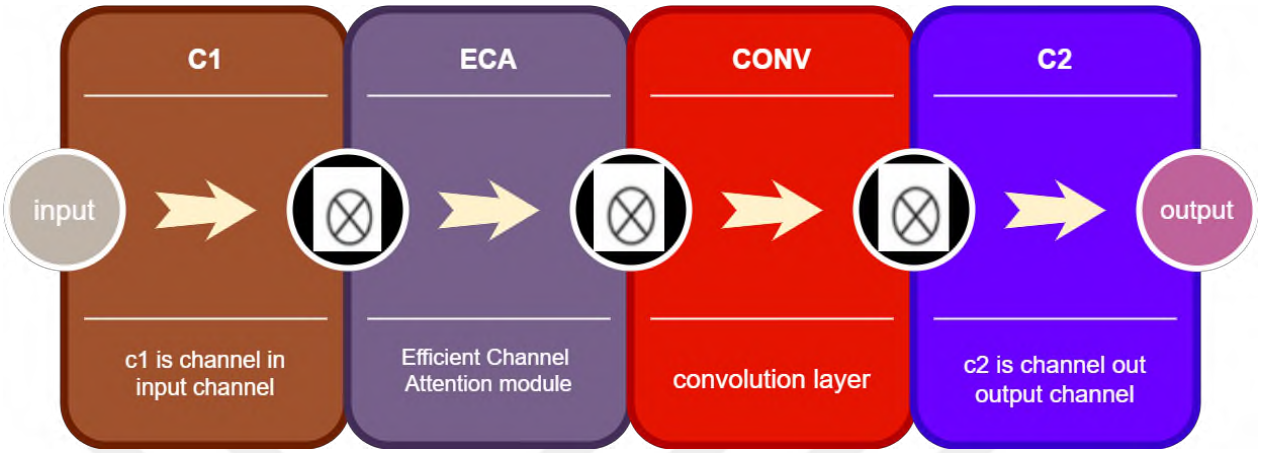

Fig 3.28: New module is being created, ECA module is integrated

ECA is integrated right after c1 and before convolution layer. To consider ECA as a forward method, it applies average pooling, convolution, sigmoid and element wise multiplication that comes out with a final output tensor.

With the __init__ method which is initialization, the module initializes different convolution layer components. Coming to these components and explaining them all in detail, avg_pool is a layer of adaptive average pooling that spatially downsamples the input tensor to a predetermined 1x1 output size. For attention convolution, att_conv is the ECA module's 1D convolutional layer. Sigmoid activation function is for channel-wise scaling in the ECA module. The input tensor's convolution operation is carried out by the primary convolutional layer. Batch normalization is represented as bn, this is a layer that is applied following the convolution process. For activation function, act as following batch normalization, the activation function is applied. A layer of linear transformation, linear is applied to ensure that the size of the input tensor x matches the size of the output tensor from the ECA module.

Pseudo code:

```
function forward(self, x):    # Input tensor x
    eca_output = self.eca(x)     # Apply ECA (Enhanced Channel Attention)
    conv_output = self.conv(eca_output) # Apply convolution operation
    bn_output = self.bn(conv_output)     # Apply batch normalization
    activation_output = self.act(bn_output)     # Apply activation function
    return activation_output
```

In the architecture, first ECA module, then convolution, then batch normalization and final part activation function. As a remark from this code, forward method is applied. The activation function, batch normalization, and convolutional layer are applied to the input tensor x. Following the avg_pool layer is used to average-pool the tensor x to a size of 1x1. The ECA module, which consists of a 1D convolution, sigmoid activation, and channel-wise scaling, is applied to the average-pooled tensor y which is flattened and reshaped using the linear layer to match the size of the input tensor x after the ECA module has been applied. For the last part, element-wise multiplication is carried out between x and y after the altered tensor y has been rebuilt to resemble x's original shape.

For the Fused Convolution Forward Pass (forward_fuse method), without using any further procedures like linear transformation or ECA, this technique just applies the convolution operation and activation function to the input tensor x. All things are being considered, this part constructs a newly convolutional layer with improved features including element-wise multiplication with the input tensor and linear transformation to match the output size for channel-wise attention.

3.9.2. ECSI

The two components are combined as High and Low Order Global Attention by the Enhancing Module. Neural networks, especially transformer models, employ both low-order and high-order

global attention techniques in order to see the relationships between various input data segments. In Low-Order Global Attention, the model combines data in a straightforward way from all around the input space. Because they are computationally efficient, low-order attention mechanism is used to record coarse-grained interactions. 3x3D convolution is followed by reshaping, fully linked layers, and sigmoid activation in the low-order global attention section. Continuing process as High-Order Global Attention, splitting the input, using 7x7, 1x1, 5x5, and 5x5 convolutions, channel shuffling, and sigmoid activation are all part of the high-order global attention portion. They interact between various input components as in more complex processes, such as the Transformer model's self-attention mechanisms. For the input part, high-order attention enables the model to discover complex patterns and connections.

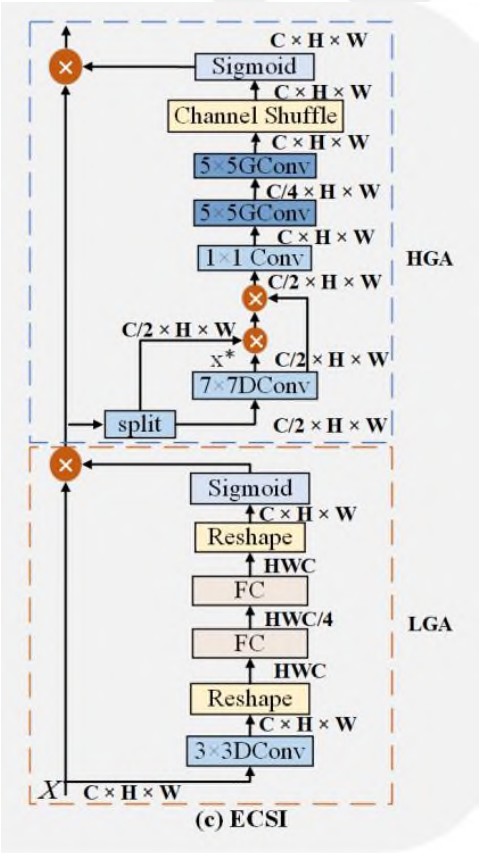

Fig 3.29: ECSI figure

Taking into account for positive sides and developed features of ECSI, they will be quite effective on object detection. Getting into detail for these two attention mechanisms; considering HGA part;

by including multi-scale data and utilizing sophisticated convolutional procedures, such as 5x5GConv and 7x7DConv, the HGA module improved the feature extraction procedure in YOLOv8+. As in Channel and Spatial Attention, adding sigmoid and channel shuffle operations suggests that the feature maps are improved by highlighting significant channels and spatial regions, which strengthens the attention mechanism of the model.

Considering LGA, by reshaping and processing feature maps using FC layers and 3D convolutions, the LGA module is focused on improving the YOLOv8+s capacity to locate items precisely.

On the feature representation part, it gives much better representation for the ECSI module it makes quite possible for the network to capture more discriminative features as background effects, which improves representation learning by improving both channel-wise and spatial interactions. The network is much effective and able to comprehend the context of objects within the input data thanks to the ECSI module's facilitation of the integration as of improved contextual information across channels and dimensions. Attempts to incorporate contextual information and decrease dimensionality are demonstrated by the repeated reshape and FC operations on reducing dimensionality. This directly improved the detection accuracy, particularly for tiny or closely spaced objects which the scores are higher than 90% on YOLOv8+. The LGA module's usage of sigmoid and reshape operations, like HGA, points to a complex attention process that is intended to improve the feature maps' detection performance.

3.9.3. LGIM

The Local Global Interactive Module is intended to extract dependencies from the input data that are both local and global in normal sense. For its system of operating, Local portion of the module concentrates on collecting local characteristics and associations while processing the input tensor locally. It is composed of a SiLU activation function and a 3x3 convolution that is followed by batch normalization. The module will be able to collect spatial connections and fine-grained features within the input thanks to this local processing.

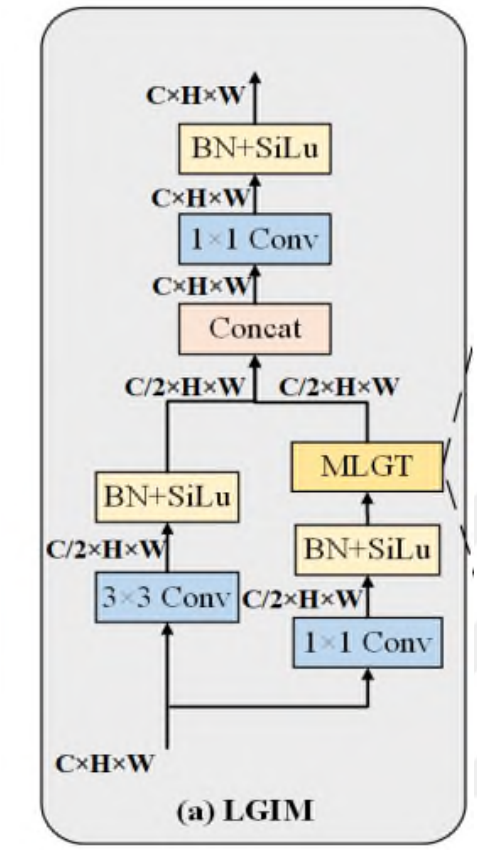

Fig 3.30: LGIM figure

The Mixing Local Global Transformer block consists of many convolutional layers that are activated with SiLUs and batch normalization. The model may discover intricate correlations and patterns throughout the whole input space thanks to this block. This block's combination of local and global data allows the module to efficiently identify both short- and long-term connection in the data. The input feature maps of dimensions C×H×W, where C is the number of channels and H×W is the spatial dimensions, are the first thing the module sees. The first feature maps flow via an activation function called SiLU after passing through a BN layer. In order to increase the model's stability and learning capacity, this phase normalizes the feature maps and adds non-linearity. A 1×1 convolutional layer is applied to improve feature representation without changing the spatial dimensions by reducing or mixing the number of channels. This lessens computing strain while preserving crucial information. The BN+SiLU and 1×1 Convolution layers' outputs are combined

together. Concatenation combines characteristics from many layers, adding more information to the representation. Two branches are then created from the concatenated feature maps. After additional processing on each branch, there are only C/2 channels left.

Considering the first branch, after BN+SiLU, one branch has a 3×3 convolutional layer. In order to precisely localize objects, this combination captures spatial relationships and refines the feature maps even more. A MLGT is incorporated in the second branch. An MLGT usually improves the feature maps by capturing long-range relationships and interactions, giving the local features a global context. The feature maps pass via a 1x1 convolution and an additional BN+SiLU layer after the MLGT. In this stage, the features that have been improved worldwide are refined and reintegrated into the network. The final output tensor is generated by this last convolutional layer, which combines local and global data as C×H×W is enriched in the final part.

4. EXPERIMENTAL STUDY

4.1. Performance Evaluation

On the results and performance evaluation part, there are some metrics which represent the accuracy of the algorithm. These metrics also help us to analyze all the values in detail and case by case; training, testing and validation parts can be analyzed in detail (Fang et al.,2024).

Precision is the number of accurate object detections, or the accuracy of the identified objects. Recall is the model's capacity to recognize every occurrence of an object in an image. The average precision measured at a threshold of 0.50 for IoU is called mAP50. IoU is necessary when exact item placement is important. To mention in a general sense, mAP is a fit for a comprehensive assessment of model performance. In the result graphs, there is another metric related with mAP value mAP50-95: The mean average sensitivity determined at different IoU thresholds between 0.50 and 0.95. It provides a thorough understanding of the model's functionality at various detection difficulty levels. These two different mAP values help to divide the IoU into two and then considering them for valuable criteria. These metrics compute the precision and recall of the predicted bounding boxes with respect to the ground truth, and are used to evaluate the accuracy of object identification algorithms. By taking into account a larger range of IoU (Qaddour,2023) thresholds and capturing both high and low overlap between predicted and ground truth bounding boxes, mAP50-95 offers a more thorough review. The exact constraints of the object identification job and the intended balance between accuracy and recall determine the measure to utilize.

Precision is quite important in circumstances when reducing false positives is the top concern. Recall is essential when it comes to identifying each and every instance of an object. F1 Score is much helpful when memory and accuracy need to be balanced. The F1 score at different thresholds is shown by the F1 Score Curve. Understanding how this curve is interpreted might help one understand how the model balances false positives and false negatives at certain levels. Curve of Precision-Recall is a crucial tool for visualizing classification problems as it illustrates the balance

between recall and accuracy at different thresholds. It takes on particular significance when handling uneven class sizes. Precision levels at various thresholds are graphically represented by the Precision Curve. This curve makes it easier to see how sensitivity varies with threshold. Recall Curve illustrates the variation in recall levels at various thresholds.

The first table represents the scores and their success rates for COCO dataset and 5 algorithms. These two tables will be used for comparison for COCO dataset and newly created dataset with taken samples from inside the bus in Roboflow.

Considering COCO dataset on YOLOv5 and YOLOv8 before implementing YOLOv8+. This comparison is for without ECSI and LGIM implementation. These are the results for all models in mAP, Precision, Recall and F1 score;

Table 4.1: Results for COCO dataset on 7 objects

Model	**mAP**	**Precision**	**Recall**	**F1 score**
YOLOv5	0.69385	0.68456	0.67463	0.68001
YOLOv8	0.81077	0.80276	0.79465	0.80767
R-CNN	0.71573	0.70813	0.72753	0.71825
Mask RCNN	0.73568	0.73693	0.75140	0.75862
Faster R-CNN	0.78796	0.76621	0.77926	0.79715
Stand-alone CNN	0.72846	0.73292	0.72947	0.73216
Stand-alone RNN	0.51496	0.51933	0.52324	0.50014
Stand-alone LSTM	0.69123	0.69395	0.69032	0.69439

One of the cases that differs in YOLOv8+ and YOLOv8 is that, the umbrella in COCO dataset is an open version, however inside the bus, the detected umbrella needs to be closed umbrella. As these cases occur in object detection for forgotten objects, the newly created dataset's results shows the success fitting for this detection.

Because stand-alone CNNs tackle simpler tasks (image categorization), they outperform YOLOv5 in terms of inference speed. They are effective for jobs that just need detecting an image's general information since they can process images in 1–10 ms. While YOLOv5 is not as successful as a standalone CNN, it is still quite effective at detecting objects. YOLOv5 (particularly the smaller version, YOLOv5s) is one of the quickest object detection models available for real-time applications that need object localization and classification; it operates at 5–10 ms per frame.

A CNN will be more efficient and quicker if the task is just categorization. However, because of its superb mix of speed and precision, YOLOv5 is a great option if you need real-time object recognition. Stand-alone CNN can handle spatial data and image hierarchies; it performs significantly better in object detection tasks across all metrics. It performs well in bounding box prediction as well as object localization.

Stand-alone RNN, due to its sequential design, which makes it ill-suited for spatial relationships in images, RNN struggles with object detection tasks. All indicators show a significant decrease in performance, especially in localization tasks.

This is the table for two layers introduction to COCO results that has been done with scales of all algorithms. Mainly, for two-staged algorithms R-CNN, Mask R-CNN, Faster R-CNN; Stand-alone CNN; Stand-alone RNN and Stand-alone LSTM; for one-staged algorithms YOLOv5, YOLOv8 and YOLOv8+. Considering the COCO dataset on YOLOv5 and YOLOv8 with implementing YOLOv8+. This comparison is for with ECSI and LGIM implementation these are the results for all models in mAP, Precision, Recall and F1 score;

Table 4.2: Results with two layers introduction for COCO dataset adding YOLOv8+ on 7 objects

Model	mAP	Precision	Recall	F1 score
YOLOv5	0.73597	0.72959	0.73927	0.72944
YOLOv8	0.79045	0.76938	0.77485	0.78133
YOLOv8+	0.87560	0.86345	0.85035	0.87496
R-CNN	0.79259	0.79627	0.79297	0.79970
Mask RCNN	0.81783	0.81737	0.81846	0.81058
Faster R-CNN	0.81796	0.81927	0.81086	0.81017
Stand-alone CNN	0.74276	0.74978	0.75927	0.76172
Stand-alone RNN	0.59376	0.60284	0.60276	0.61836
Stand-alone LSTM	0.72957	0.73867	0.74827	0.73817

Comparing the algorithms and their success rates, comparing the mAP scores with two-staged and one-staged algorithms, one-staged is more successful and high results. The greatest result is on YOLOv8+. It has reflected the best results on each compared algorithms.

Regarding inference speed, CNNs are significantly more effective for applications involving object detection. Their ability to handle spatial data in parallel makes them ideal for object detection in real-time. Because RNNs process sequences piecemeal, they are significantly slower and result in long inference times for jobs using images. Their architecture makes them unsuitable for object detection tasks, and their testing time is too sluggish for real-time applications. When it comes to applications that demand object detection accuracy and speed, CNNs are the obvious solution.

Table 4.3: Results for augmented dataset on 7 objects

Model	mAP	Precision	Recall	F1 score
YOLOv5	0.72812	0.73807	0.72050	0.73056
YOLOv8	0.78960	0.79514	0.78045	0.80702
R-CNN	0.76905	0.76298	0.76127	0.76538
Mask RCNN	0.78837	0.77837	0.77928	0.77827
Faster R-CNN	0.81834	0.81737	0.80378	0.81837
Stand-alone CNN	0.73826	0.73726	0.73989	0.73217
Stand-alone RNN	0.55836	0.56938	0.55029	0.56380
Stand-alone LSTM	0.70375	0.70167	0.70276	0.70376

CNN on its own may benefit from quicker training and more effective processing, but the complexity of the dataset determines how much performance is improved by adding more layers. For the general perspective here, adding two layers and a new dataset helped to enhance the accuracy of object detection, particularly for diverse and augmented dataset in here.

Seeing the results on Faster R-CNN, it is clear that why it is being used on this thesis, which is the improved performance in unknown environments. Faster R-CNN's two-stage detection procedure makes it already quite accurate, and enhanced datasets make it even more capable of generalizing to new, unknown data. Taking into account of better item recognition in a range of environments, the model can adapt to alterations brought about by augmentation, such as shifts in illumination, size, and orientation.

Table 4.4: Results with two layers introduced for augmented dataset adding YOLOv8+ on 7 objects

Model	mAP	Precision	Recall	F1 score
YOLOv5	0.80275	0.79385	0.79265	0.80318
YOLOv8	0.86369	0.85938	0.85029	0.85980
YOLOv8+	0.94307	0.93840	0.94309	0.92270
R-CNN	0.85027	0.86023	0.87017	0.87208
Mask RCNN	0.85796	0.86927	0.85086	0.86017
Faster R-CNN	0.90947	0.89761	0.90360	0.89932
Stand-alone CNN	0.76827	0.75917	0.76039	0.76387
Stand-alone RNN	0.62949	0.62194	0.62097	0.62036
Stand-alone LSTM	0.75709	0.75029	0.75298	0.75027

RCNN already extracts region ideas and runs them through a CNN for categorization. This is where adding two more layers (more convolutional layers, for example) could help in feature extraction, especially for bigger or more complicated datasets. On the added layers effect, RCNN had been able to capture more intricate characteristics and perform better on difficult datasets. However, because RCNN is already slow, this also resulted in a considerable increase in training time.

Mask R-CNN's capacity to extract more intricate features from the data is directly improved by adding two more layers, either to the mask prediction branch or to the feature extraction backbone. Convolutional layers, for instance, might enable Mask R-CNN to record finer details, improving instance segmentation. For its accuracy, by including more layers, the model's accuracy is quite increased, particularly for small or heavily obscured objects.

More depth helped the feature extraction layers of Faster R-CNN, enabling the model to capture more intricate patterns. Enhancing the model's ability to learn better features from the new dataset may require adding layers to the backbone (such as ResNet or other CNN architectures). Especially on this complex dataset that has been on the augmented dataset, faster R-CNN have seen improvements in bounding box localization and classification accuracy.

Stand-alone CNN has been able to learn more abstract properties from the new dataset. Nevertheless, the outcome relies on whether further depth is needed for the dataset.

Table 4.5: Results with two layers' introduction for original COCO dataset adding YOLOv8+ on 7 objects

Model	mAP	Precision	Recall	F1 score
YOLOv5	0.62264	0.61736	0.62917	0.62817
YOLOv8	0.71836	0.70374	0.70275	0.70327
YOLOv8+	0.80264	0.80391	0.80328	0.79320
R-CNN	0.74259	0.74829	0.75937	0.75928
Mask RCNN	0.71783	0.72946	0.72917	0.72198
Faster R-CNN	0.75796	0.75823	0.75978	0.76938
Stand-alone CNN	0.64276	0.64928	0.64827	0.64826
Stand-alone RNN	0.49376	0.49305	0.50389	0.50276
Stand-alone LSTM	0.62920	0.62378	0.63237	0.62958

For original COCO dataset, the greatest problem was to analyze umbrella and which it could not be able to understand it. The values decreased so critical that one object affected the results. COCO dataset has quite much number of objects so that it also took so much time in order to analyze the overall dataset for training and testing as well. The numbers are decreased compared to other parts as in all objects could not be able to recognized in a high percentage.

4.1.1. YOLOv5

Taking into consideration for YOLOv5, comparison for metrics will be examined. The below ones will be compared with YOLOv5, YOLOv8 and YOLOv8+.

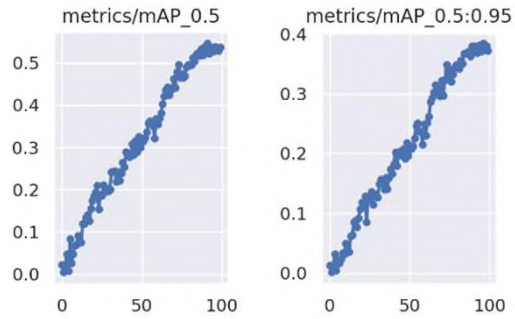

Fig 4.1: mAP scores for both 0.5 and 0.5:0.95

This is a metric comparison in which mAP stands for mean average precision, scores are seen. The x coordinate shows the number of epochs and y coordinate shows the metrics in mAP in 0.5. For YOLOv5, the mAP score for 100 epochs is 0.5865. Generally, the value reaches its peak at 100 epochs. It will be interpreted with YOLOv8 and YOLOv8+.

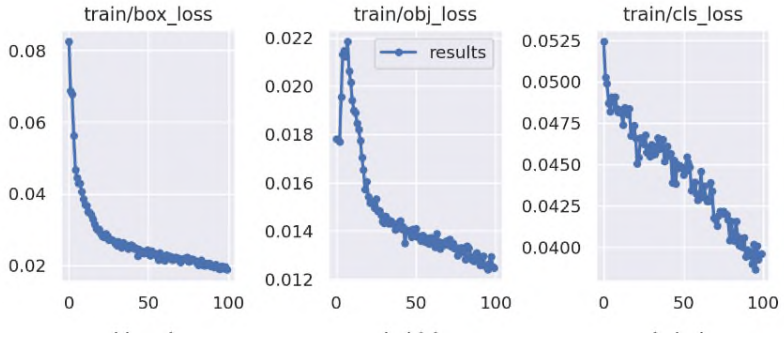

Fig 4.2: Results for YOLOv5 train loss results

The first figure represents training procedure in 100 epochs. The middle is training object loss over epochs. The left one is training box loss over epochs. These values represent the loss functions in which the model retains data. The loss function is represented in the training set. The values show in results that they fluctuate a lot which shows the process needs to be cleaned with some other additional algorithms. This is exactly the part where the algorithm needs improvement in which the lines are not directly linear but fluctuates.

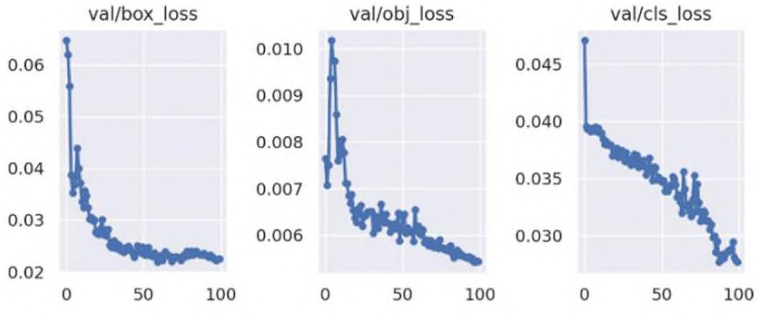

Fig 4.3: Results for YOLOv5 validation loss results

It shows the validation part in which in following; validation box loss over epochs, validation object loss over epochs and validation classification loss over epochs. It can be directly commented for validation box loss over epoch graph that, as the number of epochs rises, the validation box loss falls.

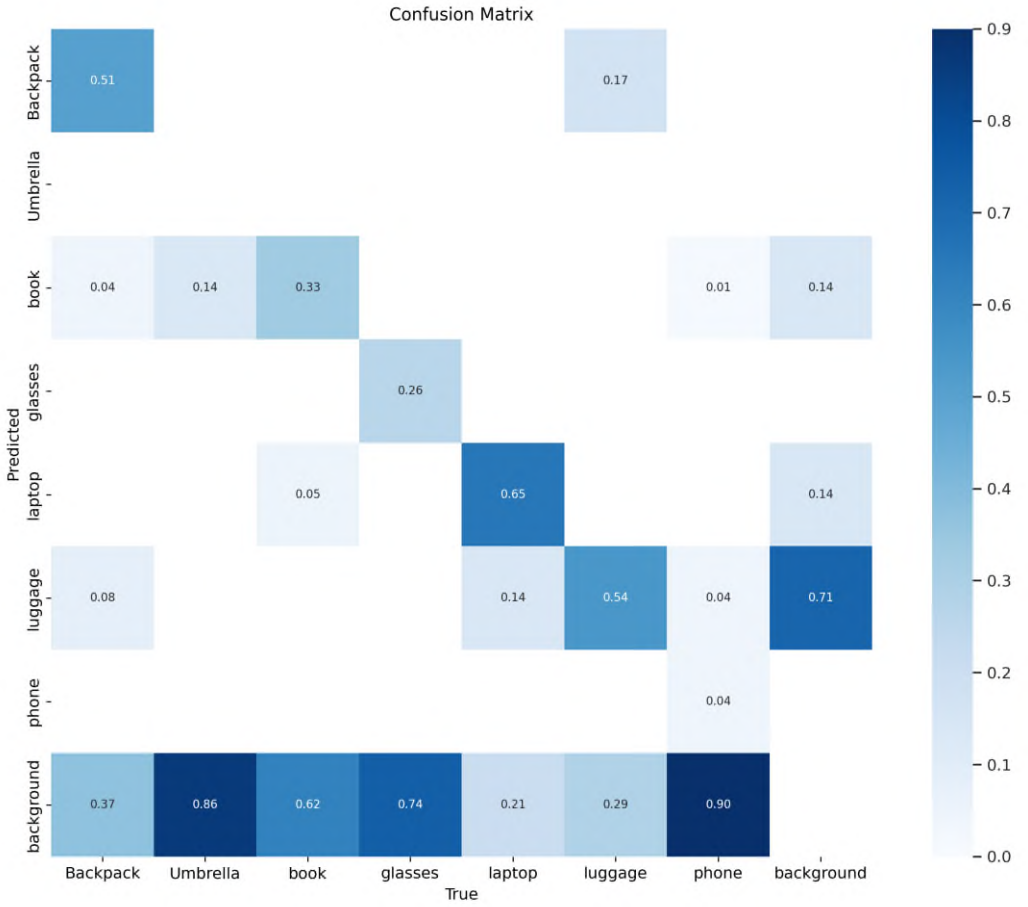

Fig 4.4: YOLOv5 confusion matrix

Confusion matrix shows briefly an overview of the results by showing the quantities of true positives, true negatives, false positives and false negatives. Instead of showing data as raw numbers, it provides it as ratios. It is actually quite simple to compare performance between classes with this format in order to see the ratios from 0.0 to 1.0 as in this range. Higher value means higher correct recognition rate for detection. Confusion Matrix works by showing the quantities of true positives, true negatives, false positives and false negatives the confusion matrix offers a thorough overview of the findings.

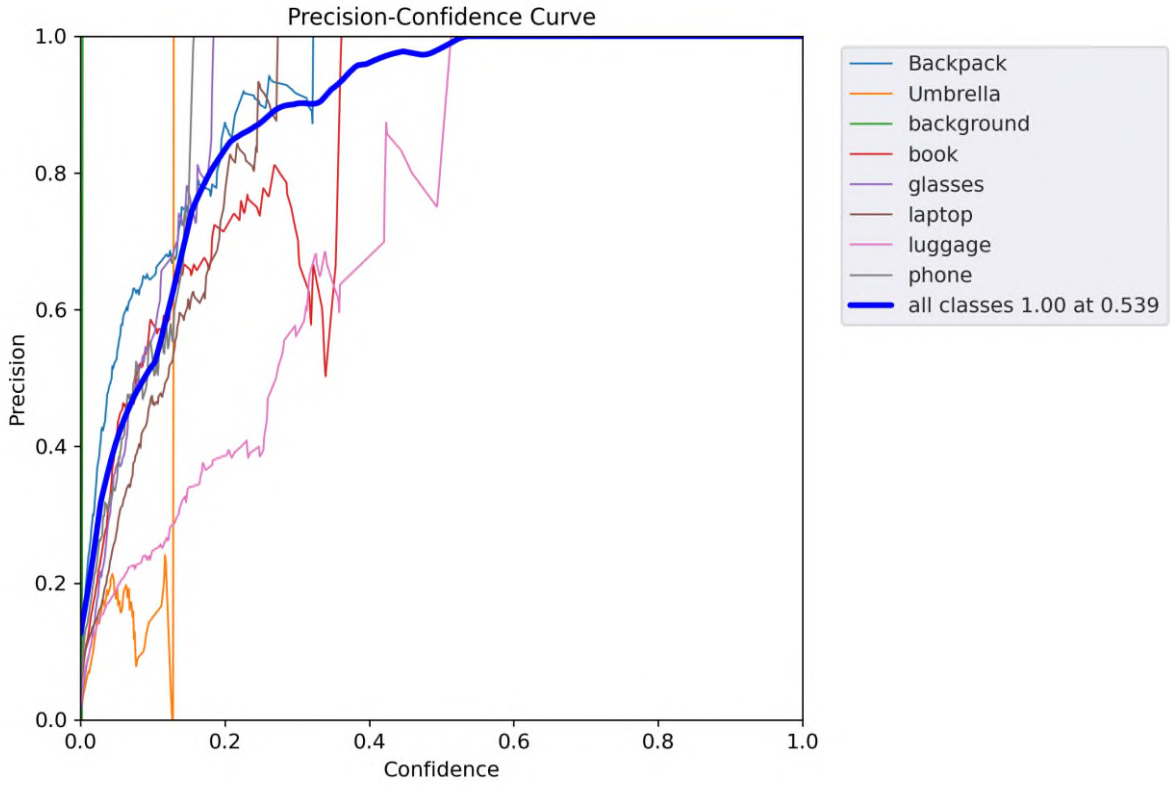

Fig 4.5: Precision-Confidence curve for YOLOv5

Precision levels at various thresholds are graphically represented by the Sensitivity Curve. This curve makes it easier to see how sensitivity varies with threshold.

4.1.2. YOLOv8 and YOLOv8+

Before the application of all the Attention Convolution Module, LGIM and ECSI; the results will be discussed for confusion matrix.

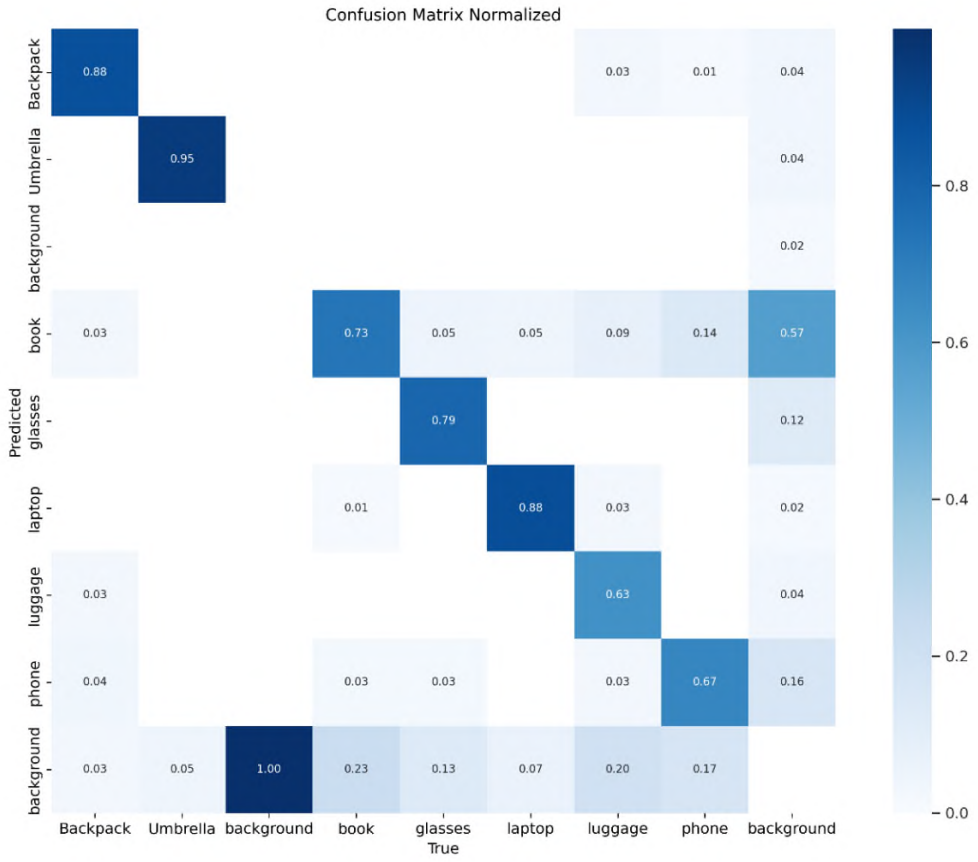

Fig 4.6: Confusion Matrix of YOLOv8 before the improvements

After applying Attention Convolution Module, LGIM and ECSI; the results are like this for all in detail, named as YOLOv8+.

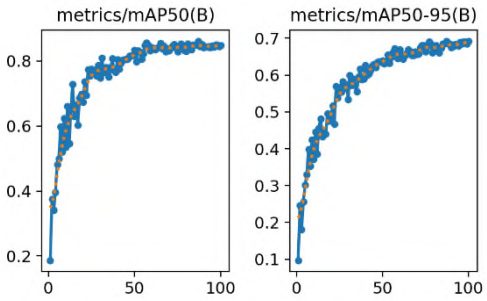

Fig 4.7: mAP scores for both 0.5 and 0.5:0.95

Taking consideration for the metrics comparison with YOLOv5 and YOLOv8+ after all implementations, this is a much better one.

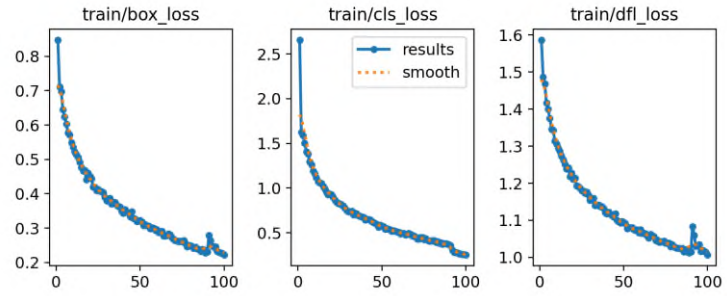

Fig 4.8: Results for YOLOv8+ train loss results

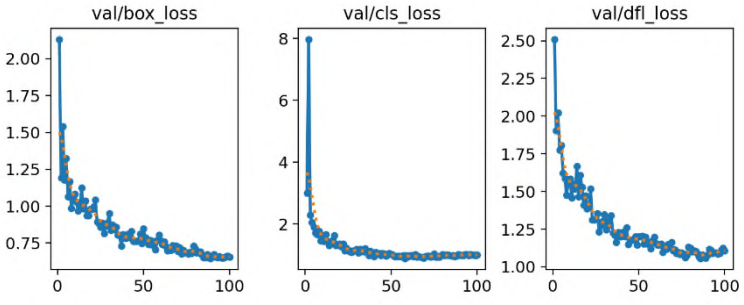

Fig 4.9: Results for YOLOv8+ validation loss results

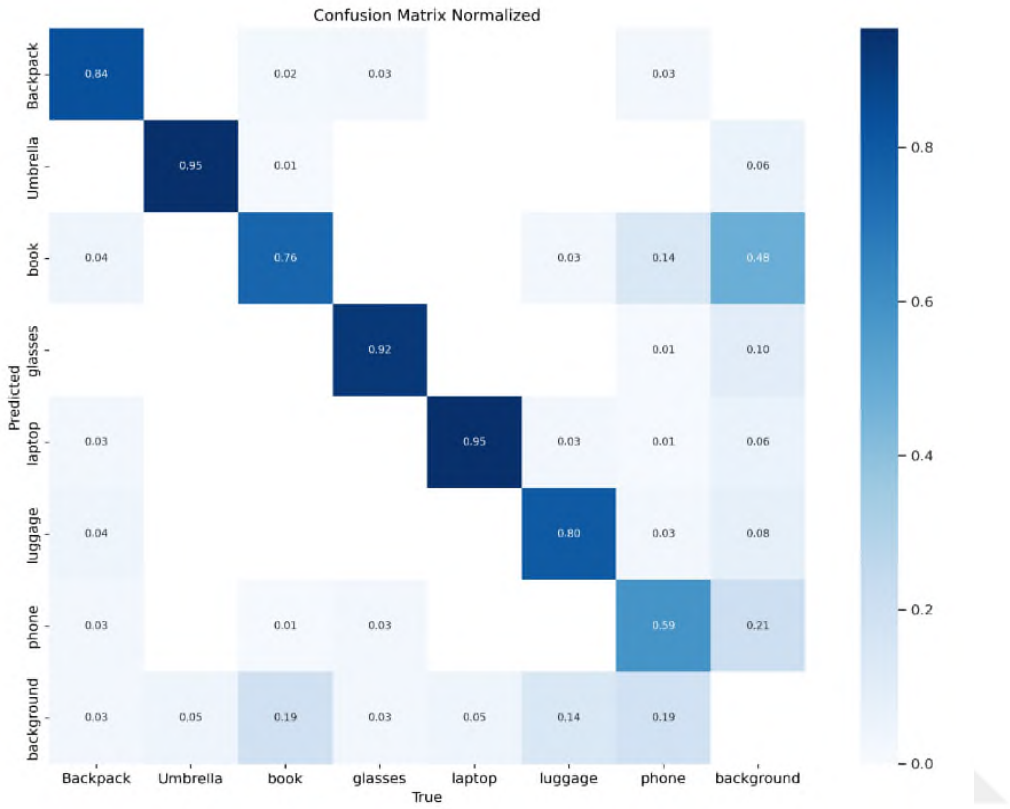

Fig 4.10: YOLOv8+ version after all implementations

After applying Attention Convolution Module, LGIM and ECSI, the results are further developed in such a successful rate. In order to analyze it with each metric, the results are much less fluctuated and resulted as a perfect line for a confusion matrix. All the graphs are nearly smoothless and direct to the values. The epoch number has been kept equal in each algorithm type, as 100.

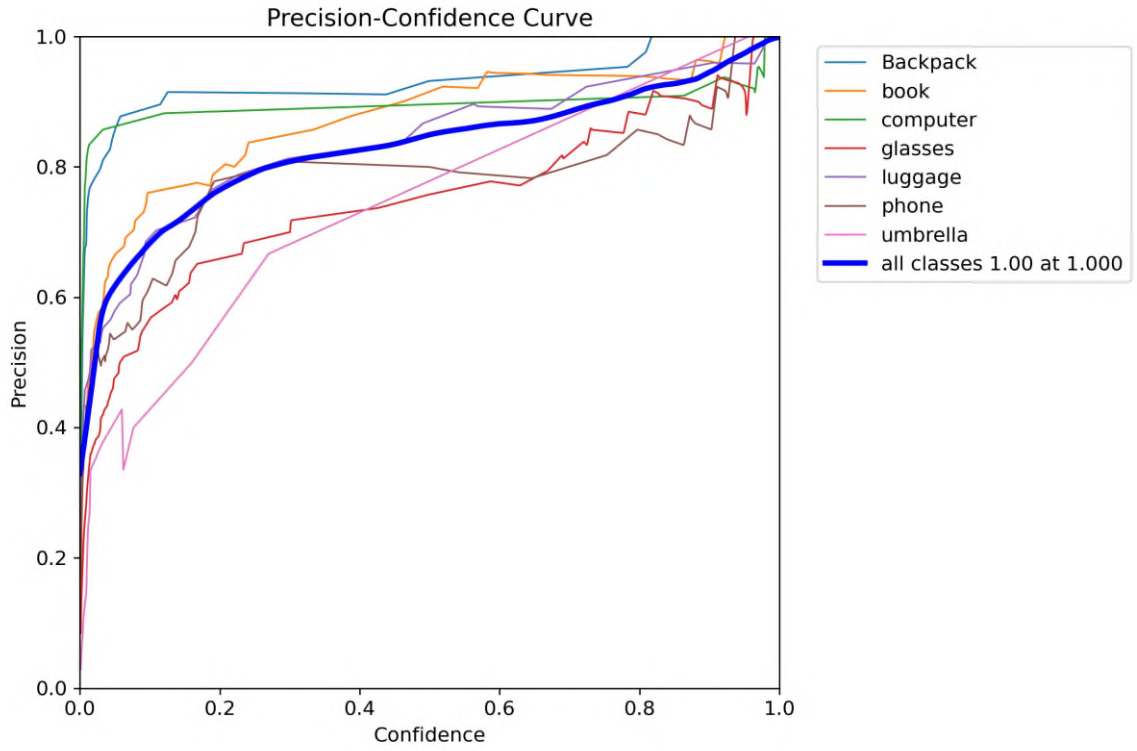

Fig 4.11: Precision-Confidence curve for YOLOv8+

Background effect

The term "background detection" is an object detection model's capacity to discern between an image's foreground items of interest and its background. A model's overall accuracy will decrease if it misinterprets background components for objects, leading to false positives. Through the extraction of pertinent information during training, models acquire the ability to distinguish between background and objects. The model can better determine which areas of the image belong in the backdrop by using high-quality features.

RPNs are used by some object identification techniques to provide possible object areas and distinguish them from the background. These areas are then categorized by the network as background or objects. By comparing each frame to a backdrop model, background subtraction techniques are used to identify moving objects in images.

The model can better comprehend what is background versus foreground by carefully annotating the data. To explicitly train the model, a class label for "background" may be present in some datasets.

Bench Marking

A methodical procedure of dataset selection, model training, assessment, and comparison with other models is involved in benchmarking YOLOv8 on object identification datasets. This procedure gives information on how well the model is performing, points out areas for development, and finally directs the choice of the best model for a given job. You can make sure that YOLOv8 is ready for the real-world scenarios and applications it will face by doing efficient benchmarking.

After the application bench marking to newly created dataset, it is now for the dataset in a developed and further scaled case. For this bench marking applied in YOLOv8+, the objects are highly identified. It is evident that a high percentage of recognition is achieved for all algorithm types. On the other hand, it is evident that the YOLOv8+ has the highest.

Runtime Costs:

Several aspects are taken into consideration when assessing the runtime cost of object detection models, such as YOLO and RCNN. These variables may be roughly divided into memory utilization, computational expenses, and pragmatic concerns like hardware requirements and deployment simplicity. Let's dissect them for RCNN and YOLO separately: YOLO Costs of Computation in inference time for Yolo's real-time performance is well-known. Compared to RCNN-based models, it processes pictures in a single network pass, which makes it incredibly fast.

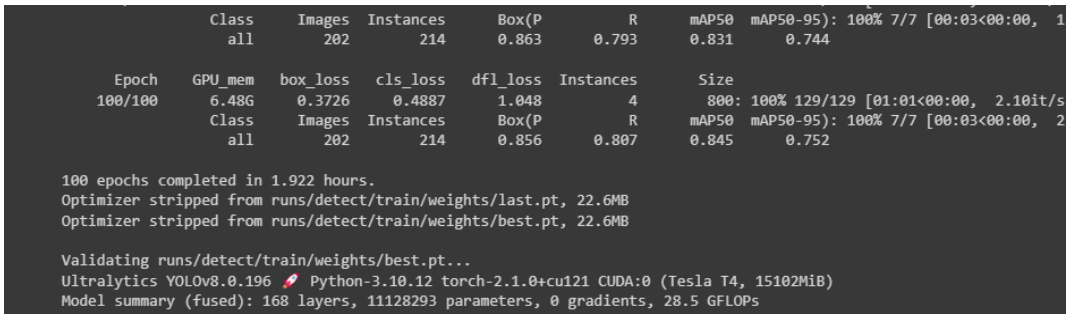

Fig 4.12: Sample for R-CNN runtime cost

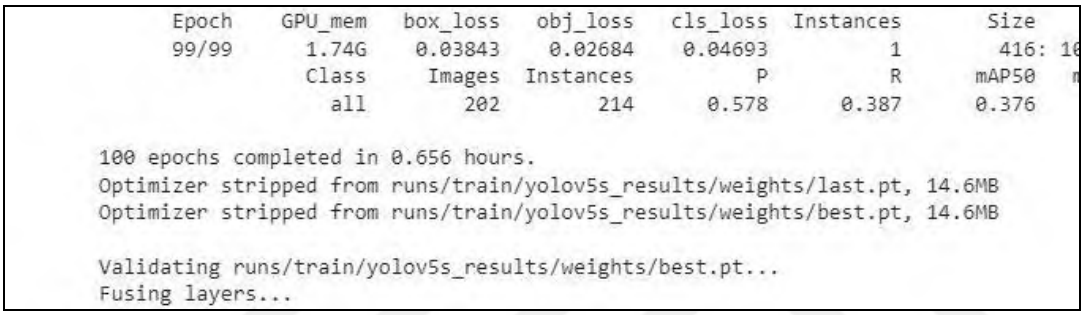

Fig 4.13: Sample for YOLOv8 runtime cost

Table 4.6: Runtime Costs comparison of COCO dataset and own dataset in training process

Model	Runtime Cost with COCO	Runtime Cost with own dataset
YOLOv5	46 mins 28.5 secs	44 mins 12.5 secs
YOLOv8	51 mins 37.5 secs	48 mins 52.4 secs
YOLOv8+	2 hour 04 mins 21.6 secs	1 hour 55 mins 19.2 secs
R-CNN	2 hour 32 mins 14.6 sec	2 hour 19 mins 4.5 secs
Mask RCNN	49 mins 45.3 secs	39 mins 21.6 secs
Faster R-CNN	42 mins 29.0 sec	39 mins 12.6 sec
Stand-alone CNN	6 hours 45 mins 53.2 sec	5 hours 56 mins 23.7 sec
Stand-alone RNN	12 hours 31 mins 24.8 sec	11 hours 16 mins 30.5 sec
Stand-alone LSTM	10 hours 14 mins 35.6 sec	10 hours 1 min 24.7 sec

YOLOv5 will probably perform better in terms of efficiency and speed when comparing training and testing timeframes for object detection among a standalone CNN, a standalone RNN, and YOLOv5. RCNN models often exhibit lengthier inference times. The original RCNN was slower since it used a two-stage method (region suggestion followed by classification). Although they have improved, RCNN and Mask R-CNN are still usually slower than YOLO.

The whole object recognition process from dataset preparation and model training to testing and deployment, is streamlined by Roboflow. To evaluate the performance of the model, use a validation dataset that is different from the training set. Analyze measures such as F1 score, recall, precision, and mAP. Here is the test process for runtime cost comparing with COCO and own dataset.

Table 4.7: Runtime Costs comparison of COCO dataset and own dataset in testing process

Model	**Runtime Cost with COCO (ms)**	**Runtime Costs with own dataset (ms)**
YOLOv5	19.48 ms	15.72 ms
YOLOv8	12.97 ms	11.22 ms
YOLOv8+	8.65 ms	6.15 ms
R-CNN	26.81 ms	23.02 ms
Mask RCNN	25.34 ms	24.43 ms
Faster R-CNN	25.43 ms	23.69 ms
Stand-alone CNN	27.53 ms	33.83 ms
Stand-alone RNN	55.42 ms	38.61 ms
Stand-alone LSTM	43.33 ms	42.03 ms

Fig 4.14: Samples from augmented dataset closed umbrella detection

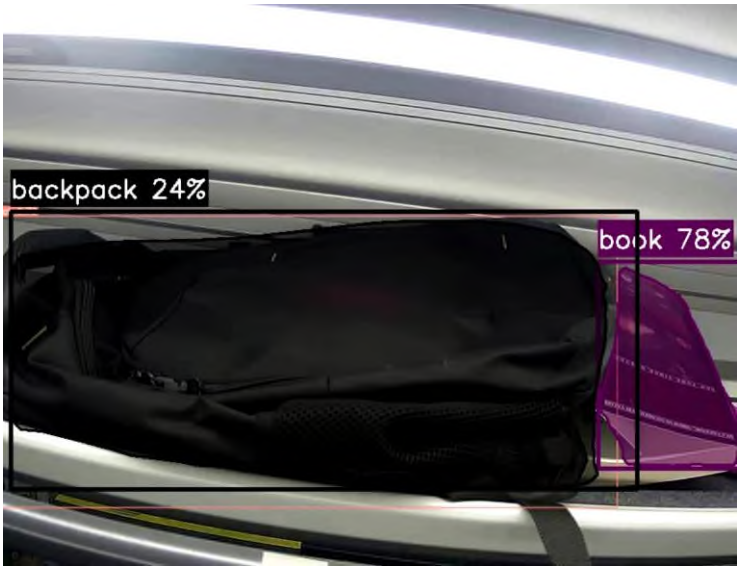

Fig 4.15: Samples from augmented dataset object detection

YOLOv5 is even compatible with CPUs and can achieve real-time inference speeds of up to 200 FPS on a GPU. In terms of inference speed, it performs better than RNNs and solo CNNs.

Despite being slower, standalone CNNs can nevertheless offer testing speeds that are quite quick (5–20 FPS on GPU). Standalone RNNs are inappropriate for real-time applications because they are too sluggish for object detection.

RCNN models often exhibit lengthier inference times. The original RCNN was slower since it used a two-stage method (region suggestion followed by classification). Although they have improved, RCNN and Mask R-CNN are still usually slower than YOLO.

5. CONCLUSIONS AND DISCUSSIONS

Recursive neural networks, a subset of recurrent neural networks, or RNNs, may handle organized or sequential input, however this is not very common. For instance, a recursive method might follow objects over time, leveraging input from earlier frames to improve detection in subsequent frames, if an object recognition job necessitates context from a series of pictures (e.g., video object detection).

Why the RNN or recursive algorithms are used here is, complex or hierarchical items can also be handled using recursive methods. The model may construct a whole item by iteratively identifying smaller components and then aggregate these findings. This technique is particularly helpful for gradually identifying items with intricate structures, such as people, buildings, or trees. Considering the stages in here to detect objects in shelves, recursive methods might be useful in managing object occlusion, a situation in which an object partially obscures another item. Better localization and recognition may be achieved by the model by iteratively analyzing different aspects of the picture or scene in order to infer the hidden components.

Recursive methods might be useful in managing object occlusion, a situation in which an object partially obscures another item. Better localization and recognition may be achieved by the model by iteratively analyzing different aspects of the picture or scene in order to infer the hidden components. Recursive algorithms are not frequently used in object detection designs such as YOLO, although they can be helpful in some situations, such addressing hierarchical structures and occlusion, sub region analysis, and recursive feature refining.

There are other algorithms used and done as; YOLOv7, Mask R-CNN. The YOLOv7 is used for object detection for the sample objects. The figure 5.1 shows the YOLOv7 in a sample taken from inside the bus, overhead luggage of the part. Figure 5.2 is also a sample from overhead luggage part as well. The most recent YOLOv8 model emphasizes engineering techniques and promotes the integrated YOLO API architecture. It is mostly based on YOLOv6 and YOLOv7.

Fig 5.1: YOLOv7 objects are detected with percentages

Mask R-CNN algorithm is the algorithm type which labels the objects. The name of it is image segmentation. Pixel recognition and their belongings are related. In Mask R-CNN the object are detected with the percentage details and the labelling is done with coloring all around the objects.

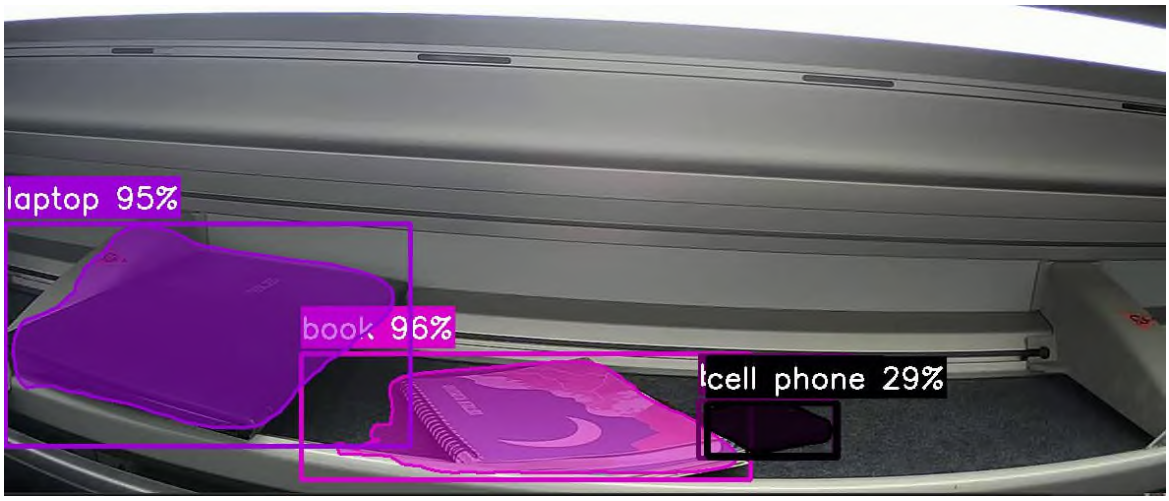

Fig 5.2: Mask RCNN sample as objects are detected with percentages

The objects recognized and colored are laptop, cell phone and book with high percentages. It is clear that all algorithm types are done with high percentage of recognition. However, we can see that the YOLOv8+ has the highest out of the others.

Fig 5.3: YOLOv8+ detecting objects with higher rate for Augmentation

This image displays the YOLOv8 model's predictions for the pertinent batches. It is easily seen that how effectively the model visually recognizes and classifies items by comparing them to the label pictures.

Fig 5.4: YOLOv8+ prediction results done correctly after the training process

Though the images are cropped or grey scaled etc. the detection rates are quite high that directly reflects the success rate of YOLOv8+

6. CONCLUSION

Coming to conclusion remarks, the new attention convolution layer with all the new architectural characteristics performs perfectly for object detection for any background since it can get closer to the improved characteristics of YOLO as mentioned in the research. Considering mainly YOLOv5 and YOLOv8, since one is anchor-based algorithm and one is anchor-free algorithm, it directly gives an outlier for the model results. Supporting them with Attention Convolution module, gives an ultimate better results for object detection. By utilizing an element-wise multiplication with the input tensor and linear transformation, adding ECA module and many data augmentation strategies, the model was able to more accurately for detecting these objects. Background creates quite much interference, so that with this newly introduces architecture, YOLOv8+ can be seen as a perfect improvement. Also further new Attention Convolution Module, two new modules are introduced as ECSI module and LGIM is together a perfect combination for detecting small and background objects. So that the algorithm is perfectly defining the background and differentiating the objects and background itself. Background interference is rather low, allowing YOLOv8 with this newly developed design YOLOv8+ to be seen as having a fairly high success rate.

Because the attention architecture is perfectly created with ECA module and linear transformation, efficiency may be increased without requiring a lot more processing power. The suggested model combines with ECSI and LGIM in image training for object detection. The suggested designs are quite faster in both training and inference as well.

In ECSI, global bounding in the input data is caught by low-order global attention for the consequences of applying these methods, and fine-grained interactions are later captured by HGA. The split and reduction procedures (such as C/2) point to a method for keeping rich feature representations while controlling computing complexity. By combining the two approaches, this model more effectively captures the dependencies at different level of detail, which enhances performance on tasks that require understanding of spatial or channel-wise correlations in the data.

By strengthening feature extraction, focusing methods, and overall detection accuracy, adding the HGA and LGA modules, as shown in the diagram, might greatly improve the YOLO design. With these improvements, the YOLO model is more resilient and effective in a range of scenarios. These

improvements focus on important areas of object detection, including accuracy, efficiency, generalization, and scalability.

For the overall perspective for LGIM module on YOLOv8+, by skillfully fusing local and global information, the LGIM improves the feature representation of CNNs, improving performance on a range of computer vision tasks like object identification, segmentation, and image classification. Through the integration of sophisticated feature extraction algorithms that integrate both local and global information, the LGIM greatly improves the YOLO architecture. Because of these advancements, YOLO is now more effective for object identification tasks due to its increased precision, enhanced localization, and more efficient computing.

Considering the addition for data augmentation and its effects, the addition made the algorithm quite faster and effective as seen on the runtime cost as well. YOLOv8+ is a newly created algorithm that has new developments, which has not been implemented beforehand with new dataset creation. In two-staged algorithm, R-CNN and Faster R-CNN are quite important to compare, so that the scores determine the results. This thesis compares the results for COCO dataset with usage of it and creation of new one as in considering these new developments. These results will be further analyzed with results for also augmentation in the future work.

Considering the highly successful results on Faster R-CNN, it is because of its strong two-stage architecture, ability to recognize objects at different sizes using feature pyramids, and ability to efficiently narrow down regions of interest through the usage of the Region Proposal Network, Faster R-CNN works well on enhanced datasets. By adding variability, data augmentation strengthens the model's resilience and improves its ability to generalize to real-world situations. This makes Faster R-CNN an extremely powerful tool for object recognition in augmented datasets. That is the main reason to compare YOLO algorithms with one of the strongest two-staged algorithm which is Faster R-CNN.

CNN models perform better on a variety of computer vision tasks, including object identification, segmentation, and classification, when ECSI modules are used to capture richer feature representations, this is represented as Enhanced Model Generalization. This helps CNN models generalize much better to unknown data.

By combining local and global processing components, the LGIM effectively captures relationships within the input data at different dimensions and levels of abstraction. This might be useful in tasks such as image classification, where accurate forecasts rely on an understanding of both local characteristics and global context. The module facilitates interactive feature fusion by merging local and global data, enabling the network to concurrently utilize local specifics and global context. The network's capacity for discrimination and accurate prediction is strengthened by this interactive fusion. LGIM enhances the richness and diversity of feature maps and enhances object detection performance by incorporating both local and global data.

Mentioning one of the hardest problem on object detection which is context and localization, the ubiquitous problem of object detection in complicated settings is addressed by the combination of sophisticated local convolutions and global transformers, which provide exact object localization and contextual comprehension as a better development on YOLOv8+ Because 1×1 convolutions and channel reduction are used to control computational complexity, the improved YOLOv8+ model is used in real-time processes. By boosting generalization over a range of object sizes and aspect ratios, the enhanced feature maps enhance the model's accuracy and resilience in a variety of settings.

For the overall perspective for LGIM module on YOLOv8+, by skillfully fusing local and global information, the LGIM improves the feature representation of CNNs, improving performance on a range of computer vision tasks like object identification, segmentation, and image classification. Through the integration of sophisticated feature extraction algorithms that integrate both local and global information, the LGIM greatly improves the YOLO architecture. Because of these advancements, YOLO is now more effective for object identification tasks due to its increased precision, enhanced localization, and more efficient computing.

REFERENCES

[1] F. S. P. Akbar, S. Y. P. Ginting, G. C. Wu, S. Achmad and R. Sutoyo, "Object Detection on Bottles Using the YOLO Algorithm," 2022 4th International Conference on Cybernetics and Intelligent System (ICORIS), Prapat, Indonesia, 2022, pp. 1-5, doi: 10.1109/ICORIS56080.2022.10031554.

[2] Cai, Xiaowei & Luo, Fuyi & Qi, Wei & Liu, Hong. (2022). A Semi-Supervised Object Detection Algorithm Based on Teacher-Student Models with Strong-Weak Heads. Electronics. 11. 3849. 10.3390/electronics11233849.

[3] Chen, C., Liu, MY., Tuzel, O., Xiao, J. (2017). R-CNN for Small Object Detection. In: Lai, SH., Lepetit, V., Nishino, K., Sato, Y. (eds) Computer Vision – ACCV 2016. ACCV 2016. Lecture Notes in Computer Science(), vol 10115. Springer, Cham. https://doi.org/10.1007/978-3-319-54193-8_14

[4] T. D. D and K. V, "Deep Learning based Object Detection using Mask RCNN," 2021 6th International Conference on Communication and Electronics Systems (ICCES), Coimbatre, India, 2021, pp. 1684-1690, doi: 10.1109/ICCES51350.2021.9489152.

[5] H.Fang, B.Han, S.Zhang, S.Zhou, C.Hu, W.Ye; Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024, pp. 1257-1266 “Data Augmentation for Object Detection via Controllable Diffusion Models”

[6] Guo, MH., Xu, TX., Liu, JJ. et al. Attention mechanisms in computer vision: A survey. Comp. Visual Media 8, 331–368 (2022). https://doi.org/10.1007/s41095-022-0271-y

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” CoRR, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[8] J. H. Hosang, R. Benenson, and B. Schiele, “Learning non-maximum suppression,” CoRR, vol. abs/1705.02950, 2017. [Online]. Available: http://arxiv.org/abs/1705. 02950

[9] S. Jain, S. Dash, R. Deorari and Kavita, "Object Detection Using Coco Dataset," 2022 International Conference on Cyber Resilience (ICCR), Dubai, United Arab Emirates, 2022, pp. 1-4, doi: 10.1109/ICCR56254.2022.9995808

[10] Kang, C.H., Kim, S.Y. Real-time object detection and segmentation technology: an analysis of the YOLO algorithm. JMST Adv. 5, 69–76 (2023). https://doi.org/10.1007/s42791-023-00049-7

[11] T. Reddy Konala, A. Nammi and D. Sree Tella, "Analysis of Live Video Object Detection using YOLOv5 and YOLOv7," 2023 4th International Conference for Emerging Technology (INCET), Belgaum, India, 2023, pp. 1-6, doi: 10.1109/INCET57972.2023.10169926.

[12] Li, Yandong & Huang, Di & Qin, Danfeng & Wang, Liqiang & Gong, Boqing. (2020). Improving Object Detection with Selective Self-supervised Self-training. 10.1007/978-3-030-58526-6_35.

[13] X. Li, F. Xu, F. Liu, Y. Tong, X. Lyu and J. Zhou, "Semantic Segmentation of Remote Sensing Images by Interactive Representation Refinement and Geometric Prior-Guided Inference," in IEEE Transactions on Geoscience and Remote Sensing, vol. 62, pp. 1-18, 2024, Art no. 5400318, doi: 10.1109/TGRS.2023.3339291.

[14] Liu, Pengfei & Wang, Qing & Zhang, Huan & Mi, Jing & Liu, Youchen. (2023). A Lightweight Object Detection Algorithm for Remote Sensing Images Based on Attention Mechanism and YOLOv5s. Remote Sensing. 15. 2429. 10.3390/rs15092429.

[15] Merugu, S., Tiwari, A. & Sharma, S.K. Spatial–Spectral Image Classification with Edge Preserving Method. J Indian Soc Remote Sens 49, 703–711 (2021). https://doi.org/10.1007/s12524-020-01265-7

[16] F. N. Ortataş and M. Kaya, "Performance Evaluation of YOLOv5, YOLOv7, and YOLOv8 Models in Traffic Sign Detection," 2023 8th International Conference on Computer Science and

Engineering (UBMK), Burdur, Turkiye, 2023, pp. 151-156, doi: 10.1109/UBMK59864.2023.10286611.

[17] Passa, R., Nurmaini, S., & Rini, D. (2023). YOLOv8 Based on Data Augmentation for MRI Brain Tumor Detection. Scientific Journal of Informatics, 10(3), 363 - 370. doi:https://doi.org/10.15294/sji.v10i3.45361

[18] K. Patel, V. Patel, V. Prajapati, D. Chauhan, A. Haji and S. Degadwala, "Safety Helmet Detection Using YOLO V8," 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN), Salem, India, 2023, pp. 22-26, doi: 10.1109/ICPCSN58827.2023.00012.

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.

[20] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” CoRR, vol.abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[21] J.Redmon and A. Farhadi, “Yolov3: An incremental improvement,” CoRR, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767n

[22] Sary, Indri & Andromeda, Safrian & Armin, Edmund. (2023). Performance Comparison of YOLOv5 and YOLOv8 Architectures in Human Detection using Aerial Images. Ultima Computing : Jurnal Sistem Komputer. 8-13. 10.31937/sk.v15i1.3204.

[23] Shan, C., Liu, H. & Yu, Y. Research on improved algorithm for helmet detection based on YOLOv5. Sci Rep 13, 18056 (2023). https://doi.org/10.1038/s41598-023-45383-x

[24] Pravendra Singh, Pratik Mazumder, Vinay P. Namboodiri,” Context extraction module for deep convolutional neural networks”, Pattern Recognition, 2022, 108284, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2021.108284.

[25] B. Su, H. Zhang, J. Li and Z. Zhou, "Toward Generalized Few-Shot Open-Set Object Detection," in IEEE Transactions on Image Processing, vol. 33, pp. 1389-1402, 2024, doi: 10.1109/TIP.2024.3364495.

[26] Talaat, F.M., ZainEldin, H. An improved fire detection approach based on YOLO-v8 for smart cities. Neural Comput & Applic 35, 20939–20954 (2023). https://doi.org/10.1007/s00521-023-08809-1

[27] Tian, J., Jin, Q., Wang, Y. et al. Performance analysis of deep learning-based object detection algorithms on COCO benchmark: a comparative study. J. Eng. Appl. Sci. 71, 76 (2024). https://doi.org/10.1186/s44147-024-00411-z

[28] Kang Tong, Yiquan Wu, “Rethinking PASCAL-VOC and MS-COCO dataset for small object detection”, Journal of Visual Communication and Image Representation, Volume 93, https://doi.org/10.1016/j.jvcir.2023.103830.

[29] Kang Tong, Yiquan Wu, “Rethinking PASCAL-VOC and MS-COCO dataset for small object detection”, Journal of Visual Communication and Image Representation, Volume 93, 2023, https://doi.org/10.1016/j.jvcir.2023.103830.

[30] Qaddour, Jihad. (2023). Object Detection Performance: A Comparative Study. 10.21203/rs.3.rs-3181849/v1.

[31] Wang, Chengji & Luo, Zhiming & Lian, Sheng & Li, Shaozi. (2018). Anchor Free Network for Multi-Scale Face Detection. 1554-1559. 10.1109/ICPR.2018.8545814.

[32] Wang, Pei & Tang, Yin & Luo, Fan & Wang, Lihong & Li, Chengsong & Niu, Qi & Li, Hui. (2022). Weed25: A deep learning dataset for weed identification. Frontiers in Plant Science. 13. 1053329. 10.3389/fpls.2022.1053329.

[33] Q. Wang, et al., "ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020 pp. 11531-11539. doi: 10.1109/CVPR42600.2020.01155

[34] Sanghyun Woo, Jongchan Park, Joon-Young Lee, In So Kweon, “CBAM: Convolutional Block Attention Module” Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 3-19 https://doi.org/10.48550/arXiv.1807.06521

[35] Zang, Y., Zhou, K., Huang, C. et al. Semi-Supervised and Long-Tailed Object Detection with CascadeMatch. Int J Comput Vis 131, 987–1001 (2023). https://doi.org/10.1007/s11263-022-01738-x

[36] Y. Zhang, J. Miao and C. Liu, "Detection of bolts and nuts of automobile sheet metal parts based on YOLOV7," 2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2023, pp. 1648-1651, doi: 10.1109/EEBDA56825.2023.10090809

[37] R. Zhao, K. Wang, Y. Xiao, F. Gao and Z. Gao, "Leveraging Monte Carlo Dropout for Uncertainty Quantification in Real-Time Object Detection of Autonomous Vehicles," in IEEE Access, vol. 12, pp. 33384-33399, 2024, doi: 10.1109/ACCESS.2024.3355199.

[38] Zhao, R., Wang, K., Xiao, Y., Gao, F. and Gao, Z. 2024. Leveraging Monte Carlo Dropout for Uncertainty Quantification in Real-Time Object Detection of Autonomous Vehicles. IEEE Access, vol. 12.

[39] Zheng, Zhaohui & Wang, Ping & Liu, Wei & Li, Jinze & Ye, Rongguang & Ren, Dongwei. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. Proceedings of the AAAI Conference on Artificial Intelligence. 34. 12993-13000. 10.1609/aaai.v34i07.6999.

[40] Yan Zhou, A YOLO-NL object detector for real-time detection, Expert Systems with Applications, Volume 238, Part E, 2024, https://doi.org/10.1016/j.eswa.2023.122256

[41] Terven, Juan & Cordova-Esparza, Diana-Margarita & Ramirez-Pedraza, Alfonzo & Chávez Urbiola, Edgar. (2023). Loss Functions and Metrics in Deep Learning. A Review.

[42] Yuan Chen, Huilan Luo, VisioSignNet: A Dual-Interactive Neural Network for enhanced traffic sign detection, Expert Systems with Applications, 2024, https://doi.org/10.1016/j.eswa.2024.124688.

YOLOv4: https://www.mathworks.com/help/vision/ug/getting-started-with-yolo-v4.html

YOLOv8: https://yolov8.org/what-is-yolov8/