ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

# ΜΕΘΟΔΟΙ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΟΡΑΣΗΣ ΓΙΑ ΤΟΝ ΕΝΤΟΠΙΣΜΟ ΑΝΤΙΚΕΙΜΕΝΩΝ ΜΕ ΧΡΟΝΙΚΑ ΑΠΟΔΟΤΙΚΕΣ ΜΕΘΟΔΟΥΣ

ΑΥΛΩΝΙΤΗ ΕΛΕΝΗ ΣΠΥΡΙΔΟΥΛΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ
Ιακωβίδης Δημήτριος
Καθηγητής

Λαμία, 2024

UNIVERSITY OF THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & BIOMEDICAL INFORMATICS

# COMPUTER VISION METHODS FOR OBJECT DETECTION USING TIME-EFFICIENT TECHNIQUES

AVLONITI ELENI SPYRIDOULA

FINAL THESIS

ADVISOR

Iakovidis Dimitris
Professor

Lamia, 2024

*«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:*

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

*3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

*4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: 10./..7../20.24

Ο – Η Δηλ. ΑΥΛΩΝΙΤΗ ΕΛΕΝΗ ΣΠΥΡΙΔΟΥΛΑ

*(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση*
*του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων*
*σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»*

# ΠΕΡΙΛΗΨΗ

Η τεχνολογία ανίχνευσης αντικειμένων βρίσκεται στο προσκήνιο της προόδου πολλαπλών τομέων, καθώς επιτρέπει την ακριβή αναγνώριση και τοποθέτηση αντικειμένων μέσα σε εικόνες. Αυτή η εργασία επικεντρώνεται στην κριτική αξιολόγηση και βελτίωση των αλγορίθμων ανίχνευσης αντικειμένων, παρουσιάζοντας την ευρεία εφαρμοσιμότητά τους σε διάφορους τομείς, από την ιατρική απεικόνιση έως την αεροπορική επιτήρηση. Μία καίρια εφαρμογή αυτών των τεχνολογιών είναι στον ιατρικό τομέα, ιδιαίτερα στην πρώιμη ανίχνευση του καρκίνου του παχέος εντέρου, όπου μπορούν να έχουν σημαντικό αντίκτυπο στα αποτελέσματα για τους ασθενείς.

Ο καρκίνος του παχέος εντέρου αποτελεί σημαντικό παγκόσμιο κίνδυνο για την υγεία, απαιτώντας αποτελεσματικές στρατηγικές έγκαιρης ανίχνευσης για τη βελτίωση της κατάστασης των ασθενών. Η κολονοσκόπηση παραμένει ζωτικό εργαλείο για την έγκαιρη ανίχνευση πολύποδων, που μπορεί να μειώσει σημαντικά τον κίνδυνο καρκίνου του παχέος εντέρου. Πρόσφατες εξελίξεις στα συστήματα υποβοηθούμενα από υπολογιστή ανίχνευση (Computer Aided Detection, CAD), που χρησιμοποιούν προηγμένους αλγορίθμους αναγνώρισης αντικειμένων, έχουν δείξει ελπιδοφόρα αποτελέσματα στην ενίσχυση της αποτελεσματικότητας των κολονοσκοπήσεων.

Αυτή η μελέτη στοχεύει στην αξιολόγηση και σύγκριση της απόδοσης τριών προηγμένων μεθόδων ανίχνευσης αντικειμένων: του αλγορίθμου YOLOv5 (You Only Look Once), του βελτιωμένου αλγορίθμου HIC-YOLOv5, και του βελτιωμένου αλγορίθμου YOLOv5s-TST. Επιπλέον, η έρευνα εισάγει μια νέα μέθοδο βελτίωσης για την περαιτέρω ενίσχυση των υπαρχόντων αλγορίθμων, με έμφαση στην αύξηση της ταχύτητας και της ακρίβειας ανίχνευσης. Χρησιμοποιώντας ένα εκτενές σύνολο δεδομένων που περιλαμβάνει εικόνες κολονοσκόπησης με διάφορα μεγέθη, τύπους και τοποθεσίες πολύποδων, καθώς και ένα σύνολο δεδομένων με εικόνες που έχουν καταγραφεί από drone σε πτήση, οι αλγόριθμοι υφίστανται αυστηρή εκπαίδευση και βελτίωση για τη βελτιστοποίηση των δυνατοτήτων ανίχνευσής τους.

Μέσω ολοκληρωμένων πειραματισμών και αναλύσεων, η έρευνα προσδιορίζει την ανώτερη προσέγγιση για την ανίχνευση πολύποδων και την ανίχνευση αντικειμένων γενικότερα, παρέχοντας πολύτιμες πληροφορίες για τη βελτίωση των συστημάτων CAD. Τα ευρήματα υποδεικνύουν σημαντικές βελτιώσεις στην απόδοση της ανίχνευσης, οι οποίες μπορούν να υποστηρίξουν πρωτοβουλίες έγκαιρης ανίχνευσης και τελικά να μειώσουν την εμφάνιση καρκίνου του παχέος εντέρου. Η έρευνα προς την κατεύθυνση αυτή μπορεί να συμβάλει στην ανάπτυξη πιο αποδοτικών και ακριβών διαγνωστικών εργαλείων, που ενδέχεται να μεταμορφώσουν τις κλινικές πρακτικές και να βελτιώσουν τα αποτελέσματα της υγείας.

## ABSTRACT

Object detection technology stands at the forefront of advancing numerous fields by enabling precise identification and localization of objects within images. This work focuses on the critical evaluation and enhancement of object detection algorithms, showcasing their broad applicability across different domains, from medical imaging to aerial surveillance. A pivotal application of these technologies is in the medical field, particularly in the early detection of colon cancer, where they can significantly impact patient outcomes.

Colon cancer poses a significant global health risk, necessitating effective early detection strategies to improve patient outcomes. Colonoscopy remains a vital tool for the early detection of polyps, which can significantly reduce the risk of colon cancer. Recent advancements in computer-aided detection (CAD) systems utilizing sophisticated object recognition algorithms have shown promise in enhancing the efficacy of colonoscopies.

This study aims to evaluate and compare the performance of three advanced object detection methodologies: the conventional YOLOv5 algorithm, the enhanced HIC-YOLOv5 algorithm, and the enhanced YOLOv5s-TST algorithm. Additionally, the research introduces a novel enhancement method to further improve existing algorithms, focusing on boosting both detection speed and accuracy. Utilizing an extensive set of datasets of both colonoscopy images that include diverse polyp sizes, types, and locations, a dataset with images captured by a drone mid-flight, the algorithms undergo rigorous training and refinement to optimize their detection capabilities.

Through comprehensive experimentation and analysis, the research identifies the superior approach for polyp detection, and object detection in general, providing valuable insights for enhancing CAD systems. The findings suggest significant improvements in detection performance, which can support early detection initiatives and ultimately reduce the incidence of colon cancer. The research towards this direction can contribute to the development of more efficient and accurate diagnostic tools, potentially transforming clinical practices and improving health outcomes.

## Contents

# CHAPTER 1 INTRODUCTION

## 1.1. Introduction

Computers have come a long way since their creation. Their evolution, thanks to human ingenuity, assists society daily in performing basic as well as more complicated tasks. A significant contributor to this evolution is the field of Artificial Intelligence (AI). Over the past two centuries, AI has undergone remarkable development, fundamentally changing the way we interact with technology [25]. It is the science that endows machines with intelligence, drawing inspiration from the workings of the human brain. Today, AI enables us to solve complex problems by training computers, and other machines, to approximate human-level intelligence [26].

One of the areas in AI is Computer Vision, which aims to assist computers in understanding information from their environment. Computer vision systems extract insights from images or videos similar to how humans perceive visual images. This technology is widely used across industries, such as healthcare and autonomous vehicles transforming how we interact with data.

AI also covers Machine Learning, which enables computers to learn from data without programming. These algorithms empower computers to identify patterns and make decisions based on data leading to progress in fields, like analytics, natural language processing and autonomous systems.

It can be said that AI has made an impact, on healthcare by improving patient care outcomes. Through the use of AI powered systems, the healthcare industry has seen advancements in diagnosing illnesses quickly and accurately. This leads to treatment plans and better results for patients.

Medical professionals can now utilize object detection algorithms to detect any abnormalities swiftly and precisely in Magnetic Resonance Imaging (MRI), Computed tomography (CT) scans, X-Rays and other medical imaging that may go unnoticed by the eye. This early detection enables diseases like cancer to be identified and treated when chances of recovery are high.

This study focuses on exploring architectures and training methods of artificial neural networks for object detection, in images and videos using efficient computational techniques. By leveraging the latest advancements in AI, particularly in computer vision and machine learning, this research endeavors to push the boundaries of what is achievable in image and video analysis, ultimately benefiting various sectors and enhancing societal well-being.

## 1.2. Medical Background

The medical field encompasses a vast array of knowledge and practices aimed at diagnosing, treating, and preventing diseases to improve patient health and well-being. One critical aspect of medical science is the understanding of abnormal growths or masses, often referred to as polyps (Figure 1). These growths can appear in various parts of the body, including the pulmonary, genitourinary, and gastrointestinal tracts, and are derived from the Greek term "polypous," meaning "morbid lump." Notably, not all polyps exhibit neoplastic behavior, but when it comes to colorectal polyps, understanding their classification is pivotal. They can manifest as inflammatory, hamartomatous, hyperplastic, or malignant entities. However, it is the neoplastic polyps that warrant heightened concern, as they possess the potential to progress into malignancy, marking a crucial stage in the evolution of colorectal cancer [30].

Early detection and intervention play pivotal roles in managing colorectal polyps. Identifying these polyps in their early stages enables medical professionals to conduct straightforward procedures for their removal, thereby halting the progression of colorectal cancer and averting potentially severe consequences, including illness and mortality. Leaving polyps unattended can exacerbate the risk for the patient, underscoring the importance of proactive management.

The genesis of colorectal polyps lies in the colon, where the body generates an excess of aberrant cells within the lining of the digestive tract. These polyps can emerge anywhere along the length of the large intestine or within the rectum. Given the significant impact of colorectal cancer globally, with alarming statistics reported by the International Agency for Research on Cancer (IARC) of the World Health Organization (WHO), it is imperative to prioritize strategies for prevention, early detection, and treatment [31].

In 2020 alone, nearly one million deaths attributed to colon cancer were documented, with an estimated two million new cases diagnosed worldwide (reported by The International Agency for Research on Cancer (IARC) of the World Health Organization (WHO)). These figures underscore the pressing need for comprehensive approaches to colorectal health, including regular screenings, lifestyle modifications, and public health initiatives aimed at raising awareness and promoting early intervention [31]. Colorectal cancer ranks as the second most prevalent cancer among women and the third most common cancer among men, further emphasizing the urgency of addressing this significant public health challenge [29].

**Figure 1 Images of Polyps [29]**

| Types | Average Size | Cancerous | Frequency in Patients |
|---|---|---|---|
| **Tubular adenomas** | 1 to 12 mm | Non-cancerous | 50% |
| **Villous adenomas** | ≥20 mm | Cancerous | >5% |
| **Tubulovillous adenomas** | ≤1 cm | Cancerous | 8–16% |
| **Serrated adenomas** | ≤10 mm | Non-cancerous | 15–20% |
| **Hyperplastic** | ≤5 mm | Non-cancerous | 20–40% |
| **Inflammatory** | ≤2 cm | Non-cancerous | 10–20% |

**Table 1 Classes of colorectal polyps [31]**

## 1.3. Neural Networks

Neural networks, an innovative category of machine learning algorithms, are inspired by the structure and function of the human nervous system and represent a significant field in the natural and technological sciences. Developed intensively over the past forty years, they have experienced substantial growth and have numerous applications.

This technology consists of layers of interconnected artificial neurons that process data and produce results based on the input data. The ability of neural networks to mimic human reasoning and thought combines mathematical computations with biological principles, offering an innovative approach to solving complex problems.

The philosophy behind neural networks differs from classical computers, as they incorporate ideas traditionally attributed to human thinking, such as learning, memory, and forgetting, making them ideal for applications that require intelligent automation and decision-making. Neural networks enable computers to process and analyze information in a way that was previously considered possible only for humans, opening new horizons in technological development.

Artificial neural networks consist of artificial neurons, which are software nodes organized into networks, and are used in algorithms to solve various computational and problem-solving situations. A traditional neural network, called Multilayer Perceptron (MLP) or Multilayer Feedforward Neural Network (MFNN) is composed of three basic layers (Figure 2) [56]:

1. Input layer: This is the first layer of the neural network where the initial reception of data occurs. The nodes here analyze and forward the data to the next stage.

2. Hidden layer: The hidden layers are the intermediate layers between the input and output layers, where the majority of computations take place. There can be many such layers in a neural network.

3. Output layer: The final layer of the neural network serves as the system's output, with the number of neurons in this layer depending on the nature of the specific problem being solved.

MLPs do not handle spatial data efficiently. Since they treat input data as flat, one-dimensional vectors, they cannot easily recognize spatial hierarchies in images or other similar structured data.

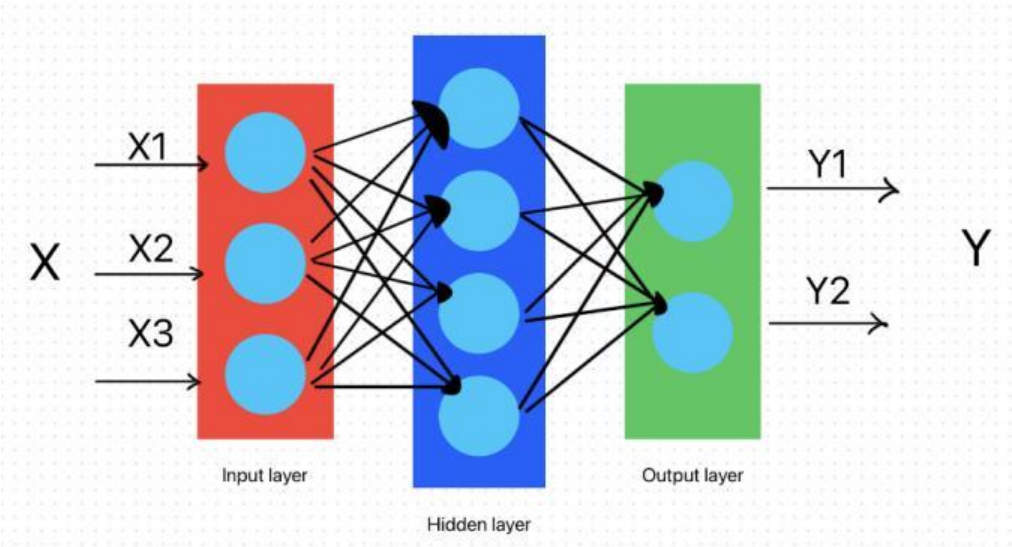



**Figure 2 Neural Network Layers**

- Convolutional Neural Network (CNN) [56]: Convolutional Neural Networks (CNNs) are similar to traditional Artificial Neural Networks (ANNs) in that they consist of neurons that optimize their parameters through learning. Each neuron receives an input and performs an operation (such as a scalar product followed by a non-linear function). From the initial input of the raw image to the final output of the class score, the entire network still expresses a single perception function. The last layer includes loss functions related to the classes, and all the usual techniques developed for traditional ANNs still apply. CNNs are composed of three types of layers: the convolutional layer, the pooling layer, and the fully connected layers (Figure 3). When these layers are stacked, a CNN architecture is created.

  The fundamental difference between CNNs and traditional ANNs is that they automatically extract features for pattern recognition within images. This makes it more efficient for image-focused tasks while simultaneously reducing the necessary parameters for model configuration.

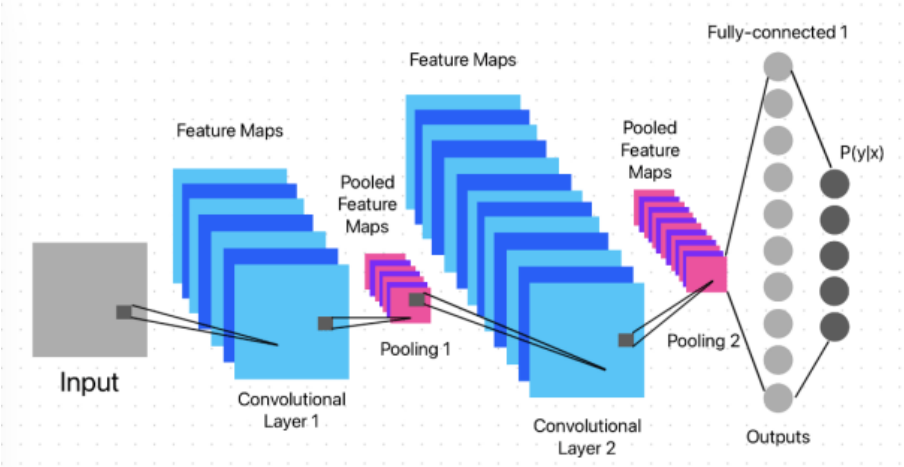



**Figure 3 Convolutional Neural Network**

This work will be focusing on the YOLO algorithm [1]. The YOLO (You Only Look Once) algorithm is a popular real-time object detection system known for its speed and accuracy. Unlike traditional methods that apply a classifier to various regions of an image, YOLO frames object detection as a single regression problem, predicting both bounding boxes and class probabilities directly from full images in one evaluation. This unified approach enables YOLO to process images in real time, making it highly efficient for applications requiring rapid detection, such as autonomous driving and video surveillance. The algorithm's ability to generalize well to new domains with minimal loss in performance further enhances its versatility and robustness in diverse computer vision tasks.

## 1.4. Study Goal

- Evaluate and Compare Object Detection Algorithms: The study aims to assess the performance of various advanced object detection methodologies: the conventional YOLOv5 algorithm, the enhanced HIC-YOLOv5 algorithm, the YOLOv5s-TST algorithm, the YOLOv9 algorithm, and an improved version of the YOLOv5 algorithm. The comparison will focus on their effectiveness in detecting objects in different settings, including medical imaging for colon cancer and more general, drone captured, images.

- Optimization of Algorithms Using Diverse Datasets: The study will utilize a comprehensive set of datasets, including colonoscopy images with diverse characteristics and images captured by drones. The goal is to rigorously train and refine the algorithms to enhance their object detection capabilities.

## 1.5. Study Contribution

The contributions of this study include:

- An Improved Object Detection Methodology: This research introduces a novel variation of a contemporary ANN-based (Artificial Neural Network) object detection method offering enhanced detection performance.

- Comprehensive Experimental Study: Through comprehensive experimentation, the research aims to identify the most effective approach for both general object detection and specific applications, such as polyp detection in colonoscopy images and pedestrian localization in drone captured images.

- Contribution to Diagnostic Tools Development: By demonstrating improvements in detection performance, this study makes a small step towards the development of more efficient and accurate image-based diagnostic tools.

## 1.6. Structure

This work is divided into 5 sections. More specifically, Chapter 1 includes the introduction which goes into detail about what this work's goal is, the definition of colorectal polyps as well as the structure of this work. Chapter 2 is the literature review where insights are provided on object detection and the extensive reference to various research and studies conducted up to date about YOLO algorithms. Apart from how each YOLO algorithm functions and is structured, their limitations are presented as well. In Chapter 3 the methodology of this work is analyzed in depth, providing information on topics that will be presented later on. Then, in Chapter 4 is where the experiments are conducted, and the results are presented. First the datasets used are listed, then the experimental methodology is analyzed and finally the results are collected. Lastly, Chapter 5 includes the conclusion and discussion of this work.

# CHAPTER 2 LITERATURE REVIEW

## 2.1. Object Detection

Finding and categorizing items in a single image is the goal of generic object detection, which also labels objects with rectangular bounding boxes to indicate their presence with confidence. One of the most important tasks in computer vision is object detection, which is searching for and recognizing items inside an image or video frame (Figure 4). It goes beyond simple image classification by not only recognizing what is present in an image but also precisely determining where each object is located. Numerous applications, such as autonomous driving, surveillance, picture retrieval, and augmented reality, depend on this feature. In this project it will be used to accurately identify and locate colon polyps.

One of the fundamental approaches to object detection is based on deep learning, particularly convolutional neural networks (CNNs). CNNs excel at learning hierarchical features from raw pixel data, making them well-suited for tasks like object detection. Models like Faster R-CNN (Region-based Convolutional Neural Network), YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector) have gained prominence due to their efficiency and accuracy in detecting objects in real-time.

There are usually several steps involved in the object detecting process. To extract features from the input image, a CNN model that has already been trained is used. Subsequent layers utilize these properties to forecast class labels and bounding boxes for every object. To improve accuracy and remove duplicate detections from the final collection of detected objects, post-processing techniques like non-maximum suppression are frequently used [32].

The goals of recent developments in object detection have been to increase robustness, accuracy, and speed. Notable performance improvements have been made possible by methods like attention mechanisms, anchor box refinement, and feature pyramid networks (FPNs). Furthermore, object detection systems may now function more successfully in difficult contexts with a variety of lighting conditions and occlusions thanks to the integration of deep learning with other technologies such as radar and LiDAR. Object detection has the potential to transform many industries and spur advancements in computer vision research as it develops further [32].

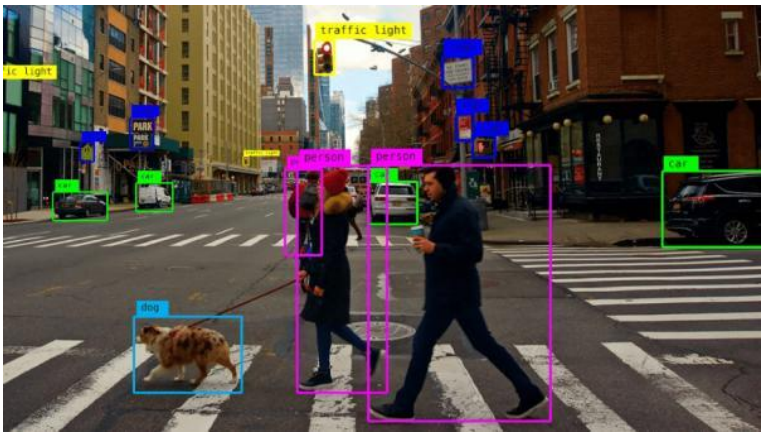



**Figure 4 Object Detection with Bounding Boxes [66]**

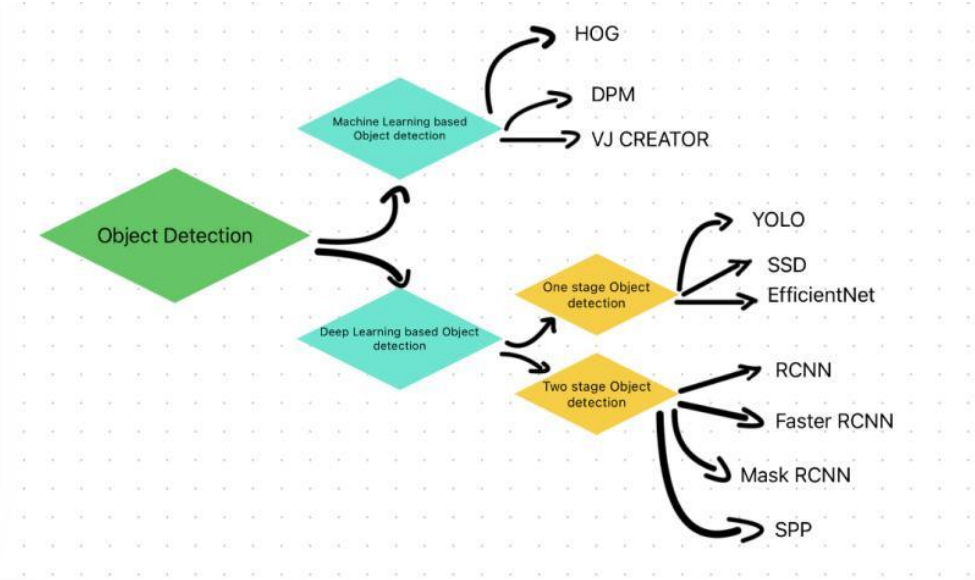



**Figure 5 Evolution of object detection algorithms [53]**

## 2.2. Detection Algorithm Categories

Finding and identifying things of interest in an image or video is the aim of the computer vision problem known as "object detection". Object Detection algorithms can be split up into two categories, Two-Stage Detectors and One-Stage Detectors. Some use a separate category for algorithms designed for drone captured images.

| Algorithm Type | Included Algorithms |
|---|---|
| **Two-Stage Detectors** | R-CNN, Fast R-CNN, Faster R-CNN, FPN, Trident Net, VFNet, CenterNet2 |
| **One-Stage Detectors** | YOLO, SSD, RetinaNet, FCOS, DETR, EfficientDet |
| **For drone captured images [33]** | RRNet, PENet, CenterNet |

**Table 2 Detection Algorithm Categories**

Two basic forms of generic object recognition method frameworks can be distinguished (Figure 6). One uses a conventional object detection pipeline, first producing region proposals, then categorizing each proposal into distinct object groups. The other uses a single framework to directly obtain the end results (categories and locations), viewing object identification as a regression or classification challenge. R-CNN (Region Based Convolutional Network), SPP-net (Spatial Pyramid Pooling), Fast R-CNN, Faster R-CNN, R-FCN, FPN, and Mask R-CNN are the primary region proposal-based algorithms; some of them are correlated with each other (e.g. SPP-net modifies RCNN with an SPP layer). The primary approaches based on regression and classification are MultiBox, AttentionNet, G-CNN (Gated Convolutional Neural Networks), YOLO (You Only Look Once), SSD (Single Shot Detector), YOLOv2, DSSD (Discrete Single Shot Detectors), and DSOD (Deeply Supervised Object Detector) [32].

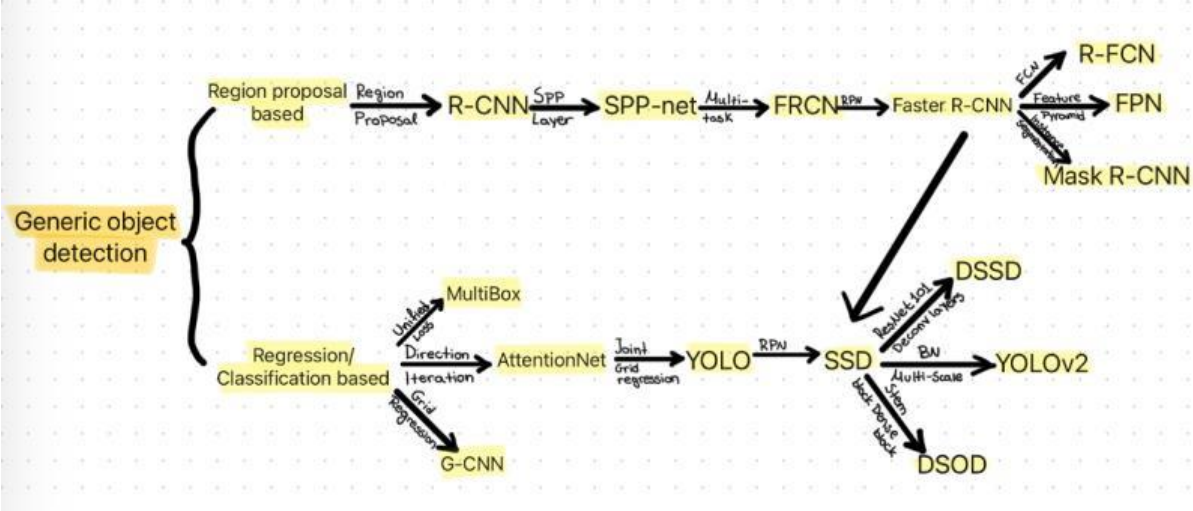



**Figure 6 Region proposal based and regression/classification-based frameworks [32]**

Two-Stage Detectors are Region Proposal Models whereas One-Stage Detectors are Regression/Classification Models.

### 2.2.1. Two-Stage Detectors

Two-Stage detectors (Figure 7) are based on candidate regions [34]. First, region proposals are generated that show the likelihood of where an object is located. Then the object classification process takes place [35]. These algorithms require two stages, in the first stage the object proposals are extracted. It refines the object proposals twice meaning that algorithms such as Fast/Faster R-CNN can obtain better object detection results compared to one-stage detectors. The only drawback is they appear to be much slower than one-stage detectors and the training process requires a lot of effort because it is a challenge to optimize each network component [37].

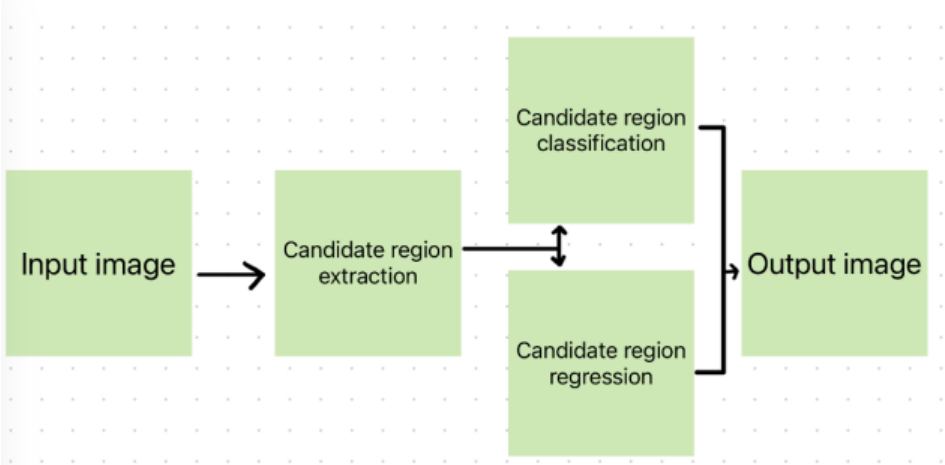



**Figure 7 Two-Stage detection algorithms [38]**

| Model | Key Features | Improvements | Issues |
|---|---|---|---|
| **R-CNN** | Uses selective search for generating object proposals | Provided a framework for object detection using CNNs | Slow due to separate stages for proposal generation and classification |
| **Fast R-CNN** | Utilizes selective search for object proposals | Improved detection accuracy and reduced computation time | Reliant on the quality of the object proposals |
| **Faster R-CNN** | Integrates object proposal generation into the network | Increased speed and accuracy compared to Fast R-CNN | Limited to fixed-sized object proposals |
| **Mask R-CNN** | Adds object mask output to the bounding box and class predictions | Improved object detection with the more precise spatial distribution of the object | Requires additional computation for the mask output |
| **Cascade R-CNN** | Employs a series of detection heads with increasing IoU thresholds | Improved accuracy by addressing the issue of missed detections | Increased computation time compared to Faster R-CNN |
| **Sparse R-CNN** | Uses predefined, learnable proposal boxes | Improved detection accuracy and reduced computation time compared to other R-CNN models | Limited to the number of proposal boxes defined |

**Table 3 R-CNN Algorithm Comparison**

#### 2.2.1.1. Example: Faster R-CNN Algorithm

This algorithm is a system that combines both RPN (Region Proposal Network and Fast R-CNN. When compared to Fast R-CNN (Figure 8), the Selective Search method is replaced with RPN. The structure of Faster R-CNN can be seen in figure 5 [38].

Faster R-CNN is made up of the following components [36]:

1. Convolution layer: Capable of extracting the image's whole feature map.
2. Region Proposal Network (RPN): the main component of the Faster R-CNN algorithm, generates a region proposal by replacing the Selective Search method in the Fast R-CNN algorithm. The CNN network can be used more quickly and effectively by using RPN to obtain a region proposal. As it creates a region proposal, RPN also creates anchors. After determining whether the anchors are background or foreground, the judgement function uses border regression to modify the anchors in order to produce an accurate region proposal.
3. ROI (Region of Interest) pooling: This technique can address the issue of varying feature map sizes entering the network at the fully linked layer. Upsampling is used to acquire the fixed size.
4. Classification layer and regression layer: The former determines the class to which an object belongs, while the latter fine-tunes the locations of regions of interest (RoIs) to get the desired object detection outcome.

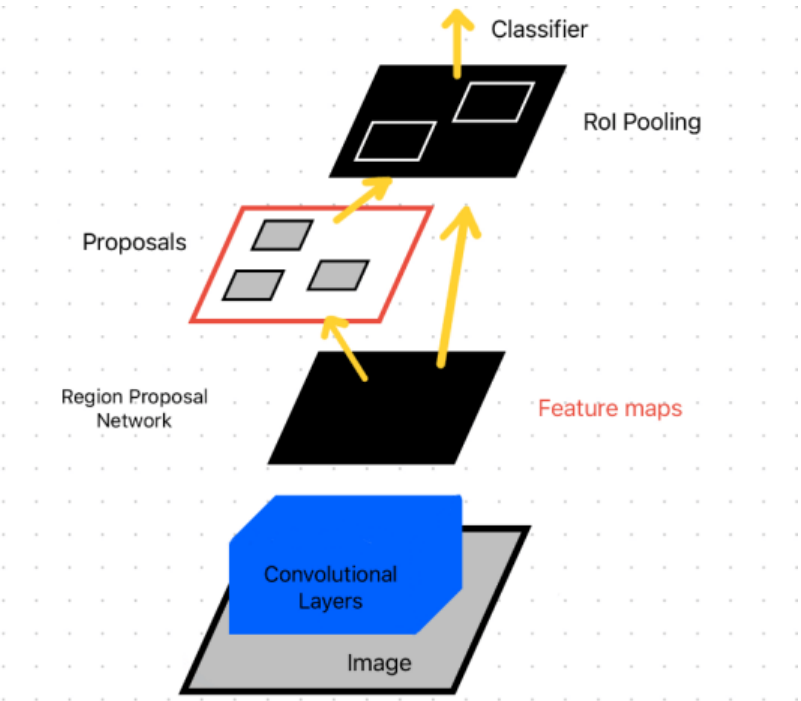



**Figure 8 Faster R-CNN network structure [36]**

It extracts features from the images by convolutional layer, and the generated feature map is input to both the region proposals network and ROI pooling layer. By combining the classification loss and localization loss in the multi-task loss, detection accuracy is improved [20]. It cannot achieve real-time detection [34].

### 2.2.2. One-Stage Detectors

One-Stage detectors (Figure 9) produce the region proposals and predict the class probabilities in one evaluation. They have faster computational speed than Two-Stage detectors but that results in detection accuracy sacrifice [35]. They have no region proposal meaning it does not separate the detection and proposal process, making the process one-stage [20]. It is alternatively known as the regression-based object detection algorithm. It indicates that the object in the picture is found and classified immediately, and the input image is no longer processed by the candidate area.

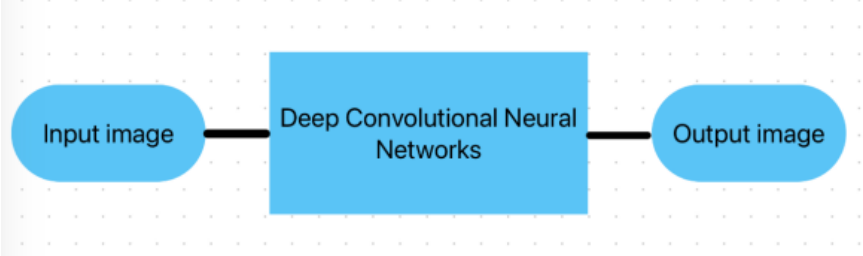



**Figure 9 One-Stage detection algorithms [38]**

### 2.2.2.1. Example: You Only Look Once Algorithm

The YOLO algorithm is a region-free one-stage detection algorithm that divides the image into grid cells and considers each cell as a proposal to detect the object. Each grid is responsible only for the targets whose centroids are within the grid [20]. The YOLO algorithm uses a newfangled residual network called Darknet, a factor that allows better feature extraction [6].

How it works: The network predicts every bounding box for every class simultaneously by utilizing characteristics from the full image. First, the input image is divided into a S x S grid by the system (for YOLOv1, S = 7) (Figure 11). When an object's center falls inside a grid cell, that grid cell is responsible for recognizing that object. Also, each cell predicts B bounding boxes as well as the confidence scores for said boxes. If no object is present in a cell, the confidence scores are equal to zero, otherwise the confidence score is equal to the intersection over union (IOU) between the predicted box and the ground truth [1].

A bounding box consists of five predictions: w (box width), h (box height), x and y (box center position), and confidence (which indicates the IOU between the predicted box and the ground truth box). The coordinates (x, y) are in relation to the grid cell's boundaries, while the coordinates (w, h) are in relation to the entire image (Figure 10) [1].

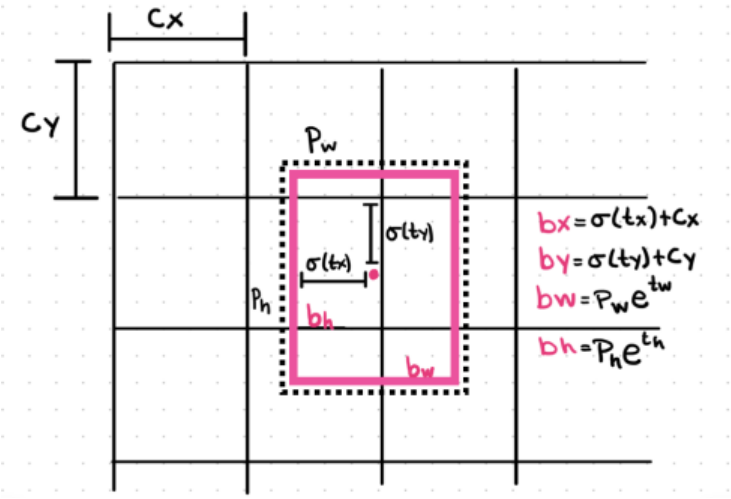



**Figure 10 Bounding boxes with dimension priors and location prediction [2]**

In addition to this, each grid cell predicts C, the conditional class probabilities (Pr(Classi |Object)), which are conditioned on the grid cell that contains an object. When it comes time to test the network, the conditional class probabilities are multiplied with the individual box confidence predictions, which provides the class-specific confidence scores for each box. Said score encode both the probability that the class appears in the box as well as how good the predicted box fits the object [1].

The network's fully connected layers forecast the output probabilities and coordinates, while the network's initial convolutional layers extract features from the image [2].

Made up of [20]:

1. Backbone: convolutional neural network that can receive images of different sizes and form the overall features of the image.

2. Neck: series of network layers that fuse the image features extracted by backbone makes the feature semantic information richer and outputs the processed features as the input of the prediction layer.

3. Head: predicts the input features. The classifier obtains the class of objects and generates the final coordinates of the bounding box.

Since the backbone pays a significant percentage of the computing cost, its design has a key impact on the inference efficiency. The backbone primarily influences the feature representation ability. Pyramid feature maps at all levels are created by combining low-level physical information with high-level semantic features via the neck. The head, which is made up of many convolutional layers, forecasts detection outcomes based on multi-level characteristics that the neck has put together. From the standpoint of the structure, it may be divided into parameter-coupled head and parameter-decoupled head, or anchor-based and anchor-free [20].

When it comes to YOLOv5, the loss is computed as the combination of three individual loss components: Classes Loss (BCE Loss), Objectness Loss (BCE Loss), and Location Loss (CloU Loss). The loss function is a crucial component that measures how well the model's predictions match the ground truth. The loss function is:

$$Loss = \lambda 1 * L_{cls} + \lambda 2 * L_{obj} + \lambda 3 * L_{loc}$$ [7].

1. $L_{clc}$ (Classification Loss): Measures how accurately the model predicts the class of each detected object. The weight λ1 adjusts the influence of the classification loss on the total loss.
2. $L_{obj}$ (Objectness Loss): Evaluates how well the model predicts whether a given region contains an object or not. It helps the model to discern between background and object regions. The weight λ2 controls the impact of the objectness loss on the overall loss.
3. $L_{loc}$ (Localization Loss): Assesses the accuracy of the predicted bounding box coordinates for each detected object. It typically uses a loss function like mean squared error or smooth L1 loss to measure the differences between the predicted and ground truth bounding box parameters (e.g., center coordinates, width, height). The weight λ3 determines the contribution of the localization loss to the total loss.

By adjusting the weights λ1, λ2, and λ3, the importance of each component in the final loss can be balanced, allowing the model to be fine-tuned for better performance in different aspects of object detection, such as improving classification accuracy, enhancing objectness confidence, or refining bounding box precision.

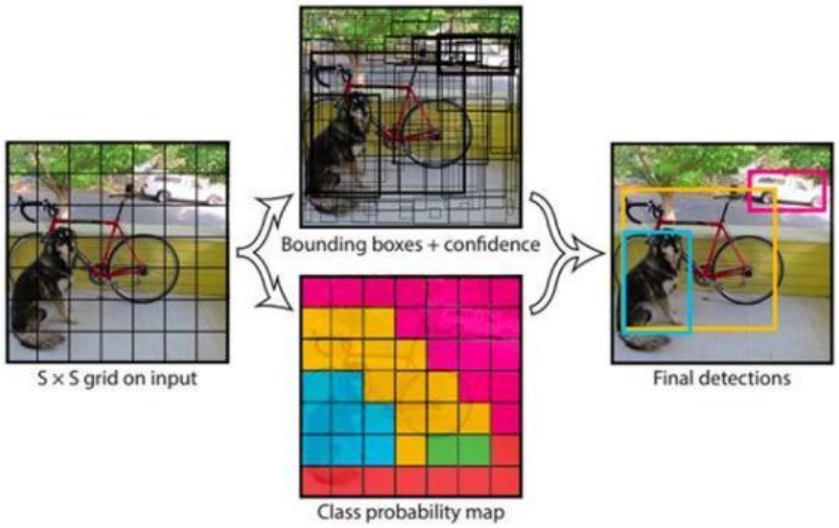



**Figure 11 YOLO Detection [1]**

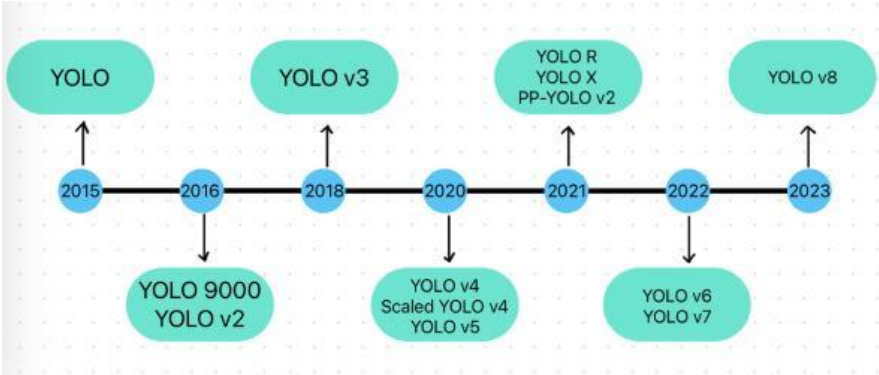



**Figure 12 The evolution of YOLO algorithms [53]**

## 2.3. Applications in Biomedical Images and Videos

### 2.3.1 General Applications

In 2024, Ragab et al. [43] performed a thorough study where they reviewed the applications of multiple YOLO algorithms with medical images, performing object detection. They found that the application of YOLO in healthcare presents significant potential, yet faces challenges like the need for extensive, diverse datasets and sensitivity to object scale in images. Despite this, continuous research offers promise, especially in medical imaging fields such as radiology, oncology, and pathology. YOLO has demonstrated efficacy in tumor detection and COVID-19 pattern recognition in lung CT scans. As healthcare digitizes, AI tools like YOLO enable the analysis of growing medical imaging data, aiding diagnosis and treatment planning. Notably, YOLO streamlines processes like breast mass detection in mammograms, offering automated classification of lesions and reducing human error.

In 2024, Huang et al. [44] proposed a new YOLO model, YOLO-MED used for medical image detection. They compared their newly created model with Faster-RCNN, RetinaNet and YOLOv5s using the Kvasir-Seg dataset, all of which YOLO-MED outperformed when it comes to detection accuracy. The only one able to surpass in real-time detection was YOLOv5s.

They also tested how well the cross-scale task-interaction module works, showing how combining information from different scales can be useful in biomedical tasks. This study is important for improving future research in biomedical multi-task learning.

In 2024, Huang et al. [61] proposed the research paper that provides an extensive evaluation of the Segment Anything Model (SAM) specifically for medical image segmentation (MIS). SAM, originally showing promise in natural image segmentation, faces new challenges in the medical domain due to the complexity of medical images such as varying modalities, detailed anatomical structures, and ambiguous boundaries. The study utilized 53 open-source datasets to create a comprehensive medical segmentation dataset named COSMOS 1050K, featuring a variety of modalities and anatomical structures. SAM's performance was analyzed across these datasets, revealing its strengths in specific scenarios and weaknesses in others. For instance, SAM performed well in tasks with clear object boundaries but struggled with amorphous lesion regions. The research highlights SAM's potential in aiding human annotation by reducing time and improving labeling quality, although its performance fluctuates with changes in the segmentation task's complexity and the modalities involved. The study also demonstrated that fine-tuning SAM on medical-specific tasks could enhance its performance, offering a valuable foundation for further research into adaptable segmentation models for medical applications.

In 2023, Aldughayfig et al. [45] uses the YOLOv5s algorithm to detect pressure ulcers, also known as bed sores, using a custom dataset from images they collected as well as images from Google Images. To enhance said dataset augmentation techniques were utilized. The trained model achieved an overall mAP50 of 0.769 and mAP50-95 of 0.398 on the validation set. This means that the model was able to accurately detect and classify pressure ulcers with a high degree of confidence. (mAP@50): It measures the accuracy of the model in detecting objects by calculating the average precision across all classes at an Intersection over Union (IoU) threshold of 0.50. A prediction is considered correct if the IoU is at least 0.50. (mAP@50-95): averages the precision across multiple IoU thresholds, starting from 0.50 to 0.95 in steps of 0.05. This metric provides a more nuanced evaluation of the model's ability to accurately detect objects and their boundaries.

In 2022, Elyan et al. [46] provided an in-depth critical technical review of the latest developments in medical image analysis, reviewing existing literature in the computer vision field that included medical image classification, shape and object recognition, and medical image segmentation. They found that before deep learning became prevalent, traditional computer vision methods were used for medical image analysis, involving feature extraction and machine learning training. Over the past decade, deep learning, and convolutional neural networks have revolutionized this field, significantly improving accuracy in applications like brain, retinal, and chest image analysis. The progress has been driven by algorithmic advancements and the availability of public datasets, enabling extensive testing and evaluation. In the literature, several datasets for medical image analysis have been presented based on recent deep learning methods. These datasets have been crucial for advancing research in complex medical image tasks. Much like Ragab et al. [43], they found that a serious limitation consists of poorly collected image data and annotations.

In 2022, Shou et al. [58] introduced the MS Transformer model, which enhances object detection in medical imaging by integrating a mask mechanism with a hierarchical transformer. This novel approach addresses the unique challenges of medical images, such as low resolution, high noise, and small object sizes, which traditional models struggle with. By segmenting the input image into patches and reconstructing the image features to emphasize small, significant

details like lesions, the MS Transformer significantly improves detection accuracy. The model's efficacy is validated on the DeepLesion and BCDD benchmark datasets, where it demonstrates superior performance in recognition accuracy and detection efficiency compared to existing models. This advancement underscores the potential of advanced neural network architectures to revolutionize medical diagnostics by enhancing the early detection and accurate diagnosis of medical conditions through more effective image analysis.

In 2022, Zhou et al. [59] wrote a paper titled "Dense Convolutional Network and Its Application in Medical Image Analysis" where they elaborated on the application and development of Dense Convolutional Networks (DenseNet) in the field of medical image analysis. The research delves into the basic principles of DenseNet, including its architectural evolution and the introduction of various modifications aimed at enhancing its structure and efficiency. Key aspects discussed include the broadening of DenseNet's architecture, development of lightweight models, and integration of advanced connection modes and attention mechanisms. The paper emphasizes the application of DenseNet across different tasks in medical imaging such as pattern recognition, image segmentation, and object detection, illustrating how these adaptations have improved the network's performance in handling medical datasets. The study showcases DenseNet's capabilities in refining image analysis which is critical for medical diagnostics, proving its utility in enhancing accuracy and efficiency in clinical settings.

In 2022, Arabahmadi et al. [60] provided a comprehensive review of the application of deep learning technologies in the diagnosis of brain tumors through medical imaging, particularly MRI. It highlights the significant potential of Convolutional Neural Networks (CNNs) in enhancing image recognition tasks within the medical field, emphasizing their role in identifying and classifying brain tumors with greater accuracy than traditional methods. The survey discusses various CNN architectures and their evolution, detailing how these models have been tailored for the segmentation and classification of brain MRI images. The paper underscores the critical challenges in traditional brain tumor diagnosis methods and how deep learning offers substantial improvements by enabling the detailed and accurate segmentation of MRI images. It also points to future directions in the field, including the need for further research on overcoming the existing limitations of deep learning models, such as the requirement for large datasets and high computational power. Overall, the study showcases the transformative impact of deep learning in medical imaging for brain tumor detection, which not only enhances diagnostic accuracy but also contributes to the broader field of smart healthcare technologies.

In 2021, Yang and Yu [57] dove into the advancements and applications of convolutional neural networks (CNNs) in the medical field, particularly focusing on object detection and semantic segmentation to enhance diagnostic processes. It outlines the development of CNNs from their theoretical underpinnings to practical implementations in digital medicine, highlighting models such as VGG, GoogLeNet, and ResNet that have improved depth and efficiency for medical imaging tasks. The paper evaluates the performance of these models in detecting medical anomalies and segmenting images at a pixel level for detailed tissue analysis, showcasing their potential to significantly reduce the workload on medical professionals and increase diagnostic accuracy. It also acknowledges the challenges in training these models, such as the need for extensive datasets and high computational demands, pointing towards ongoing needs for optimization in clinical settings.

### 2.3.2 Polyp Detection

In 2023, Ali et al. [47] decided to take on a pressing issue when it comes to polyp detection and medical image detection in general: lack of trustworthy datasets that can be used to train a network. To do so they curated a dataset from six unique centres incorporating more than 300 patients. The dataset includes both single frame and sequence data with 3762 annotated polyp labels with precise delineation of polyp boundaries verified by six senior gastroenterologists. The dataset is named PolypGen.

In 2023, Lalinia and Sahafi [50] utilized the YOLOv8 algorithm to detect colorectal polyps in colonoscopy images. First, they created a custom dataset by collecting images from multiple publicly available sources and proceeded with extensive evaluations. They found that YOLOv8m achieved 95.6% precision, 91.7% recall, and 92.4% F1-score outperforming other state-of-the-art models. The datasets used were: Kvasir-SEG, CVC-ClinicDB, CVC-ColonDB, ETIS, and EndoScene.

In 2022, Pacal et al. [49] improved the performance of both YOLOv3 and YOLOv4 Cross Stage Partial Network (CSPNet) for real-time and high-performance automatic polyp detection. Then, they used advanced augmentation techniques and transfer learning in order to improve the polyp detection process. To further improve the performance, they substituted d the Sigmoid-weighted Linear Unit (SiLU) activation functions instead of the Leaky ReLU and Mish activation functions, and Complete Intersection over Union (CIoU) as the loss function. They tested their network on both the SUN and PICCOLO datasets. Both of the proposed YOLOv3 and YOLOv4 models outperform the original YOLOv3 and YOLOv4 models.

In 2022, Yu et al. [65] introduced a novel deep learning module called the Instance Tracking Head (ITH) that integrates polyp detection and tracking in real-time during colonoscopies. This method, embedded within the YOLOv4 architecture, allows for simultaneous detection and tracking by sharing low-level feature extraction processes between the detection head and the ITH. By combining these tasks, the model avoids the inefficiencies of traditional tracking-by-detection systems. The study employed a dense similarity metric learning approach to improve tracking performance, achieving high detection accuracy (mAP 91.70%) and multiple object tracking accuracy (MOTA 92.50%) at a frame rate of 66 FPS. This model was validated on various datasets, including CVC-ClinicDB, CVC-VideoClinicDB, and ETIS-LARIB, demonstrating its capability to aid clinicians in real-time polyp detection and tracking during colonoscopies, potentially enhancing colorectal cancer screening and diagnosis.

In 2021, Li et al. [48] created an endoscopic dataset, collected from various sources, and annotated with the help of experienced gastroenterologists.

In 2021, Pacal and Karaboga [63] created an enhanced polyp detection system using a modified YOLOv4 model optimized for real-time application. Addressing the limitations of previous systems in sensitivity and specificity, this study integrates Cross Stage Partial Networks (CSPNet), Mish activation function, and Distance Intersection over Union (DIoU) loss into the YOLOv4 framework to improve detection accuracy and speed. The research demonstrates significant advancements over existing models by applying various innovative strategies such as ensemble learning and using NVIDIA TensorRT for increased processing efficiency. The system was tested using public datasets and showed superior precision, recall, and F1-scores compared to earlier models, highlighting its effectiveness in detecting polyps in real-time with high accuracy. This work contributes notably to the field of medical imaging and computer-

aided diagnosis by enhancing the detection capabilities of endoscopic systems in colorectal cancer screening.

In 2021, Nogueira- Rodríguez et al. [64] presented a deep learning model designed for real-time detection of polyps in colonoscopy images, aimed at aiding colorectal cancer diagnosis. The model is based on a pre-trained YOLOv3 architecture, fine-tuned with a dataset of 28,576 images annotated with 941 polyps. To enhance accuracy and reduce false positives, a post-processing step using an object-tracking algorithm was incorporated. The model achieved an F1 score of 0.88, with better performance on sessile and pedunculated polyps compared to flat polyps. In video-based evaluations, it demonstrated 72.61% sensitivity and 83.04% specificity using a 50-frame segment criterion, with improved sensitivity (around 90%) when the criteria were less stringent. The object-tracking algorithm notably improved specificity without significantly affecting computational performance, suggesting its effective integration into computer-aided diagnosis (CAD) systems for colorectal cancer screening.

In 2021, Ou et al. [62] discussed the development of a reduced-complexity model for detecting polyps during colonoscopies. This model is based on YOLOv5, but it is significantly lighter—only 2.8 MB compared to YOLOv3-spp's 119.7 MB—owing to a 50% reduction in convolutional kernels and the removal of a component designed to detect large objects. Despite its reduced size, the model maintains a precision close to that of YOLOv3-spp, and is better suited for low-performance medical diagnostic devices due to its simpler requirements and real-time processing capabilities. The study validates the model's effectiveness in clinical settings and highlights the importance of MS COCO pre-training and Mosaic data augmentation in improving detection precision, particularly when the training dataset is small. This advancement offers a promising tool for enhancing early colon cancer diagnosis by integrating efficient polyp detection into routine colonoscopy procedures.

## 2.4. YOLO Algorithms

- YOLOv1 [1]: Processes images at 45 frames per second in real-time and has a lower probability of detecting false positives on background (less than half compared to Fast R-CNN) but presents more localization errors. It can achieve more than twice the mean average precision compared to other real-time detection systems. It is made up of a single convolutional network that can simultaneously predict multiple bounding boxes and class probabilities for said boxes. When it comes to making predictions, the way YOLO works is it first sees the entire image during the training process, as well as when it comes time for testing, this way it implicitly encodes contextual information about the classes we are training for and their appearance. The only drawback is that even though YOLO can quickly identify objects, it struggles to precisely localize some, especially those in small size.

  The network is made up of 24 convolutional layers followed by 2 fully connected layers as well as 1 × 1 reduction layers followed by 3 × 3 convolutional layers, with the final output being a 7 × 7 × 30 tensor of predictions (Figure 13).

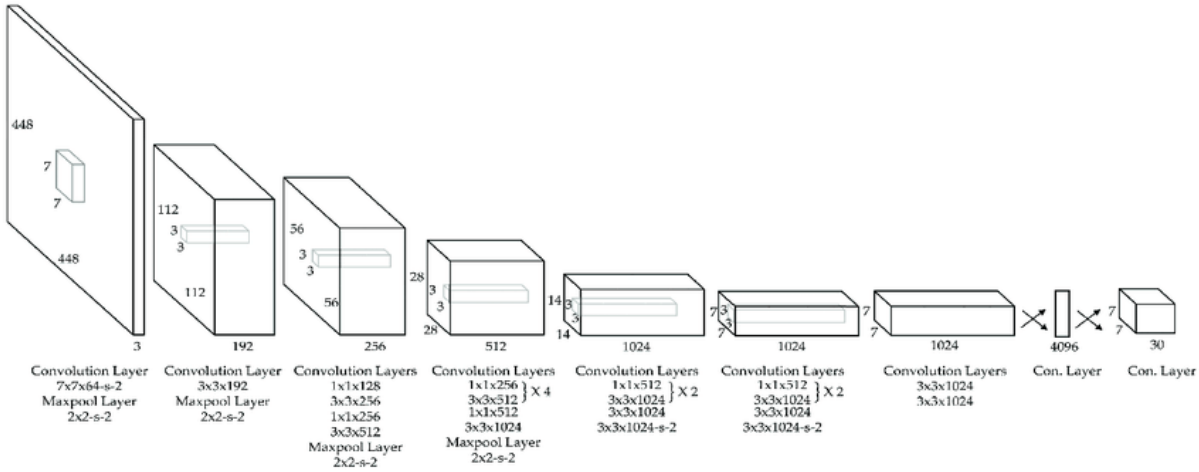



**Figure 13 The detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers [1].**

To perform detection, 4 convolutional layers and 2 fully connected layers are added with randomly initialized weights. Detection most of the time requires fine-grained visual information, so because if this the input resolution of the network is increased from 224 × 224 to 448 × 448. For each grid cell, YOLO predicts several bounding boxes. We only want one bounding box predictor to be responsible for each object during training. We assign only one predictor to oversee making predictions about an object by determining which forecast now has the highest IOU with the ground truth. Specialization amongst the bounding box predictors results from this. With practice, each predictor improves general recall by becoming more accurate at identifying specific object sizes, aspect ratios, or classes.

- YOLOv2 / YOLO9000 [2][3]: Can detect over 9000 object categories, is state-of-the-art on standard detection tasks like PASCAL, VOC, and COCO. The YOLOv2 model can run at varying sizes, offering a tradeoff between speed and accuracy. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet (Residual Network) and SSD while still running significantly faster.

  The method used in this algorithm is the hierarchical view of object classification that allows the user to combine distinct datasets together. It also proposes a joint training algorithm that lets the user train object detectors on both classification and detection data. The base YOLO detection system is improved to produce the real-time, state-of-the-art YOLOv2 detector. Its focus is to improve recall and localization while still being able to maintain classification accuracy. YOLOv2 aims for a faster and more precise detector compared to YOLOv1. Rather than scaling up the network, it is simplified so the representation is easier to learn.

  To make the algorithm better the following additions are made: Batch Normalization (it helps regularize the model, 2% mAP improvement), High Resolution Classifier (full 448 × 448 for 10 epochs on ImageNet → the network has time to adjust its filters to work better on higher resolution input, 4% mAP improvement), Convolutional with Anchor Boxes, Dimension Clusters, Direct localization prediction, Fine-Grained Features, and Multi-Scale Training.

  Most frameworks use VGG-16 (Convolutional Neural Network that is 16 layers deep) as the base feature extractor. YOLOv2 uses a custom network that is based on the GoogleNet

architecture which is faster than VGG-16 by only using 8.52 billion operations for a forward pass compared to the 30.69 billion used by VGG-16. However, its accuracy is a bit worse than VGG-16, YOLO's model gets 88.0% on ImageNet compared to 90.0% for VGG-16.

To make the algorithm better the following modifications are made: Darknet-19 is used (19 convolutional layers and 5 max pooling layers, it only requires 5.58 billion operations to process an image and achieves 72.9% top-1 accuracy and 91.2% top-5 accuracy on ImageNet), Training for classification, and Training for detection.

To make the algorithm stronger: A mechanism for jointly training on classification and detection data is proposed that uses images labelled for detection to learn detection-specific information such as bounding box coordinates and how to classify common objects. It uses images than include only class labels as to expand the number of categories it can detect. We combine images from the detection and classification datasets during training. Upon detecting an image that has been labeled for detection, our network may backpropagate using the whole loss function of YOLOv2. We only backpropagate loss from the architecture's classification-specific elements when it detects a classified picture.

- YOLOv3 [6]: Runs in 22 ms at 28.2 mAP (320 × 320), meaning it has the same accuracy as SSD but performs three times faster. Similarly, to YOLO9000, it can predict bounding boxes using dimension clusters and anchor boxes. Also, it predicts an objectness score for each one of the bounding boxes with the use of logistic regression, meaning that it should be equal to 1 if the bounding box prior overlaps a ground truth object by more than any previous bounding box. If the previous bounding box is not the best but does overlap a ground truth object by more than some threshold, the prediction is ignored. A SoftMax is not used as it has been found to be unnecessary, instead independent logistic classifiers are used and during training binary cross-entropy loss is used for class predictions.

  It can predict boxes at 3 different scales. A feature map is collected from the network and is then merged with the up sampled features using concatenation, allowing us to get meaningful semantic information from the up sampled features. K-Means clustering is still used to determine the bounding box priors, 9 clusters are chosen along with 3 scales arbitrarily, then the clusters are divided evenly across all scales (On the COCO dataset the 9 clusters were: (10×13), (16×30), (33×23), (30×61), (62×45), (59×119), (116×90), (156×198), (373×326)). Influenced by YOLOv2, a hybrid approach between its network and Darknet-19 is used that has successive 3 x 3 and 1 x 1 convolutional layers but appears to have some shortcut connections and a larger size. It is called Darknet-53 because of its 53 convolutional layers, making it more powerful than Darknet-19 but still more efficient than ResNet-101 and ResNet-152 (Figure 14).

  It was noticed that as the IOU threshold rises, performance drastically decreases, suggesting that YOLOv3 has difficulty getting the boxes precisely positioned with the object. YOLO used to have trouble handling small objects. Now, we observe a reversal of that pattern. With the new multi-scale predictions, we see YOLOv3 has relatively high APS performance. However, when it comes to medium and larger size items, its performance is significantly lower.

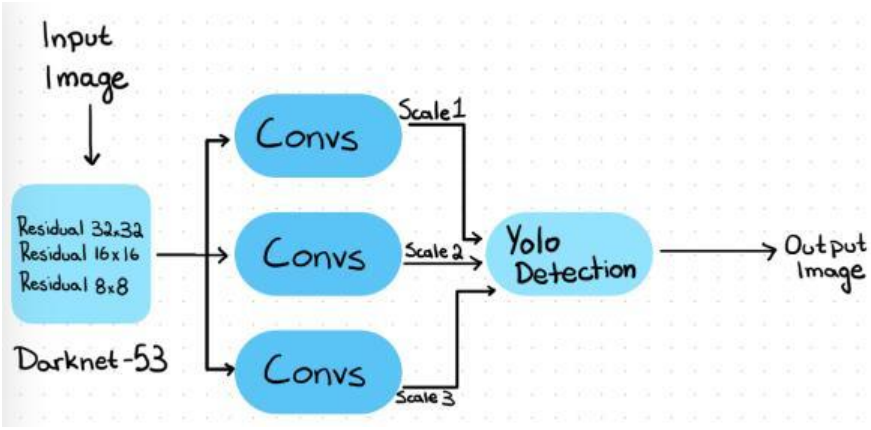



**Figure 14 Architecture of YOLOv3 [53]**

- YOLOv4 [5]: It utilizes features such as: Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT), Mish-activation, Mosaic data augmentation, DropBlock regularization, and CIoU loss.

    - Weighted-Residual-Connections (WRC): Improves traditional residual connections in deep neural networks. Normally, in these networks, the input to a layer is added to its output, helping the network learn better and train deeper. WRC adds a learnable weight to this connection, so the network can adjust how much it relies on this added input during training. This adjustment can help the network train more effectively and perform better.

    - Cross-Stage-Partial-connections (CSP): Designed to make the network easier to train and faster to run. It works by splitting the data from an early layer into two parts and then combining them again later in the network. This approach reduces the amount of computation and memory needed, making the training process more efficient without sacrificing accuracy, and can even improve it.

    - Cross mini-Batch Normalization (CmBN): A variation of batch normalization that normalizes the data across multiple mini-batches instead of just one. This is particularly useful when training with small batch sizes, as it provides more stable and accurate estimations of the batch statistics, which can lead to improved training stability and model performance.

    - Self-adversarial-training (SAT): A technique where the model learns to generate adversarial examples during training to improve its robustness. The model creates perturbed versions of the training data that are designed to fool itself, and then it learns to correctly classify these adversarial examples. This helps the model become more resilient to variations and potential adversarial attacks in the real world.

    - Mish-activation: A novel activation function defined as $Mish(x) = x * \tan h\,(softplus(x))$, where $softplus(x) = \ln(1 + e^{x})$. Mish is smooth and non-monotonic, which can help the model capture complex patterns and gradients more effectively compared to traditional activation functions like ReLU. It has been shown to improve training dynamics and achieve better performance on various tasks.

- Mosaic Data Augmentation: A technique that combines four different images into one during training. This augmentation strategy helps the model learn from more diverse and varied data within a single batch, improving its generalization ability. By seeing a mixture of different images, the model can better handle variations and complexities in real-world data.

- DropBlock Regularization: A regularization technique similar to dropout but designed for convolutional layers. Instead of randomly dropping individual neurons, DropBlock drops contiguous regions of the feature maps. This forces the network to learn more robust features, as it cannot rely on any specific local region of the feature maps. This helps in preventing overfitting and improves the generalization of the model.

- CIoU Loss (Complete Intersection over Union Loss): A loss function for object detection that looks at three things: how much the predicted box overlaps with the actual box, how far apart their centers are, and how similar their shapes are. This makes it better than traditional methods because it gives a more complete picture of how well the predicted box matches the actual box. As a result, it helps the model find objects more accurately and train more effectively

The detector is made up of two parts, a backbone which has been pre-trained on ImageNet, as well as a head than can be used to predict classes and bounding boxes. The backbone can either be VGG, ResNet, ResNeXt (Residual Networks with Next-generation Aggregated Transformations), or DenseNet (for GPU), or SqueezeNet, MobileNet, or ShuffleNet (for CPU). The head can categorize the object detector into one-stage (dense prediction) or two-stage (sparse prediction). YOLOv4 uses a one-stage object detector (YOLOv3 detection head), CSP-Darknet53 as the backbone and PANet as the neck (Figure 15).

YOLOv4 utilizes “bag of freebies” (techniques that improve the accuracy of the model during training without increasing the cost of inference, such as data augmentation) and “bag of specials” (Figure 15).

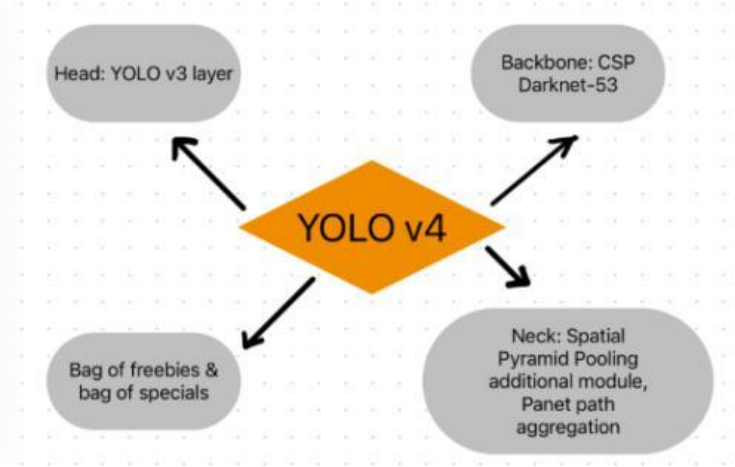



**Figure 15 YOLOv4 attributes [53]**

- YOLOv5 [7]: Its architecture is made up of three main components --> 1) Backbone: it uses CSP-Darknet53 that was previously used in YOLOv4, 2) Neck: acts as a connection

between the backbone and the head, YOLOv5 SPPF and New CSP-PAN are chosen, and 3) Head: Uses the YOLOv3 head that generates the final output. The alterations made compared to previous networks are that the Focus structure is replaced with a 6 × 6 Conv2d structure with a result of boosted efficiency and the SPP structure in the neck is replaced with SPPF with a result of more than double the processing speed.

When it comes to data augmentation, YOLOv5 utilizes Mosaic Augmentation, a technique that combines four training images into one, encouraging detection models to better handle object scales and translations. It uses Copy-Paste Augmentation, a method that copies random patches from an image and pastes them onto another image that was randomly chosen, this way it generates a new training sample. It also uses Random Affine Transformations, which includes random rotation, scaling, translation, and shearing of images. It uses MixUp Augmentation, meaning it creates composite images by taking a linear combination of two images and their associated labels. It uses Albumentations, an image library for image augmentations that supports a variety of augmentation techniques, and HSV Augmentation that creates random changes to the Hue, Saturation, and Value of the images. Finally, it uses Random Horizontal Flip, an augmentation method that can randomly flip images horizontally.

When it comes time to training, YOLOv5 uses a plethora of training strategies to enhance the model's performance including Multiscale Training, AutoAnchor, Warmup and Cosine LR Scheduler, Exponential Moving Average (EMA), Mixed Precision Training, and Hyperparameter Evolution.

- Multiscale Training: It involves varying the input image sizes during training. Instead of using a fixed image size, the model is trained on images of different scales, which helps it learn to detect objects of various sizes and improves its robustness to scale variations. This technique enhances the model's generalization capability, allowing it to perform better on images of different resolutions during inference.

- AutoAnchor: A method used to automatically select the best anchor boxes for a given dataset. In object detection, anchor boxes are predefined bounding boxes that serve as references for predicting object locations. AutoAnchor analyzes the dataset to determine the most suitable anchor shapes and sizes, optimizing the model's ability to detect objects of various dimensions more accurately. This automation reduces the need for manual tuning of anchor box parameters.

- Warmup and Cosine LR Scheduler:
    1. Warmup: A technique where the learning rate starts small and gradually increases during the initial phase of training. This helps stabilize the training process, preventing large updates that can cause instability or poor convergence early on.
    2. Cosine LR Scheduler: It adjusts the learning rate following a cosine function. It starts with a higher learning rate that gradually decreases following a cosine curve. This scheduling allows for a smooth transition in learning rates, helping the model converge more efficiently and potentially achieving better performance.
- Exponential Moving Average (EMA): EMA is a technique used to average the weights of the model over time during training. Instead of using the current weights

for evaluation, the model uses an average of its weights from recent training iterations. This averaging helps to smooth out fluctuations and can lead to better generalization and stability, often resulting in improved performance on the validation and test sets.

- Mixed Precision Training: It involves using both 16-bit (half-precision) and 32-bit (single-precision) floating-point numbers during training. This approach reduces the memory footprint and increases computational efficiency without significantly compromising model accuracy. It allows training on larger batches or more complex models on the same hardware, speeding up the training process.

- Hyperparameter Evolution: It refers to the process of using evolutionary algorithms to optimize hyperparameters. Instead of manually tuning hyperparameters such as learning rate, batch size, and momentum, an evolutionary algorithm explores the hyperparameter space by iteratively selecting and modifying the most promising hyperparameter sets. This automated search can discover better configurations that enhance model performance.

These techniques collectively contribute to improving the efficiency, stability, and performance of deep learning models, particularly in demanding tasks like object detection.

- YOLOv6 [17]:
  Backbone: While RepVGG backbones can scarcely be expanded to create bigger models due to the rapid expansion of the parameters and computational costs, we discover that they are more feature representation powerful in small networks at a similar inference speed. We consider RepBlock to be the foundation of our small networks in this regard. We update the CSPStackRep Block, a more efficient CSP block, for big models. Neck: utilizes PAN topology, following its usage in YOLOv4 and YOLOv5. The neck is enhanced with RepBlocks or CSPStackRep Blocks to have RepPAN. Head: The decoupled head is simplified to make it more efficient (Efficient Decoupled Head).

  For label assignment TAL is more effective and training friendly. The loss functions include classification loss, box regression loss and object loss where each one is systematically experimented with all the available techniques and finally VariFocal Loss is selected as the classification loss and SIoU/GIoU Loss as the regression loss. DFL is what makes the distillation of box regression possible. Two-scaled re-parameterizable backbones and necks are proposed to accommodate differently sized models.

  EfficientRep is created, an efficient re-parameterizable backbone. RepBlock serves as the primary backbone component for tiny models throughout the training phase. Additionally, during the training phase, each RepBlock is transformed into stacks of three × three convolutional layers (referred known as RepConv) using ReLU activation functions. A hybrid-channel strategy is used to build an efficient decoupled head. The number of middle 3 × 3 convolutional layers is reduced to only one. The width multiplier for the neck and backbone is used to jointly scale the width of the head. To get a shorter inference latency, these improvements further minimize computing costs.

The label assignment mechanism in the initial iteration of YOLOv6 was SimOTA. Though, it was discovered that adding SimOTA will cause the training process to underperform.

- YOLOv7 [18]: The main focus is the optimization of the training process along with architecture optimization. It is focused on a few enhanced modules and optimization techniques that could strengthen the training expense while maintaining the same level of object detection accuracy. The suggested optimization techniques and modules are referred to as "trainable bag-of-freebies". Several trainable bag-of-freebies methods are designed in order for real-time detection accuracy to be improved, while still keeping the inference cost low. Two new problems were discovered with regard to the development of object detection techniques, we discovered two new problems: the way in which the original module is replaced with a re-parameterized one, and the way in which the dynamic label assignment strategy handles assignment to various output layers. Methods are proposed to address these difficulties: 1)" extended" and "compound scaling" methods for real-time object detectors that can effectively utilize parameters are proposed, 2) The suggested approach has a quicker inference speed and better detection accuracy, and it can efficiently cut the parameters and computation of a cutting-edge real-time object detector by around 40% and 50%, respectively.

  During the inference stage, several computational modules are combined into one using model re-parametrization approaches. As an ensemble technique, the model re-parameterization technique falls into two categories: the module-level ensemble and the model-level ensemble. To get the final inference model, model-level reparameterization is often done in two ways. One method is to train several identical models with various sets of training data, and then average the weights of the various models that have been trained. The alternative involves calculating a weighted average of the model weights at various iteration counts. Recently, module-level re-parameterization has gained increased attention as a study topic. During training, this kind of approach divides a module into many identical or distinct module branches; during inference, the various branched modules are integrated into a single, fully equivalent module. Not every suggested re-parameterized module, though, can be completely adapted to various architectures. We have created a new re-parameterization module and associated application strategies for a range of architectures with this in mind.

  Extended-ELAN (E-ELAN) is proposed, based on ELAN. It uses expand, shuffle, merge cardinality with the goal to obtain the ability to continuously enhance the learning ability of the network, without destroying the original gradient path. The architecture of the transition layer remains unaltered, while E-ELAN solely modifies the architecture of the computing block. The plan is to increase the channel and cardinality of computing blocks by using group convolution. In a computational layer, we shall apply the same group parameter and channel multiplier to every computational block. Next, each computational block's feature map will be divided into g groups based on the specified group parameter g, and the groups will be concatenated together. E-ELAN can also guide different groups of computational blocks to learn diverse features.

  RepConv has performed exceptionally well on the VGG, but its accuracy will be much lower when we immediately apply it to ResNet, DenseNet, and other designs. We explore the best way to integrate re-parameterized convolution with multiple networks using gradient flow propagation pathways. In accordance, we also created a planned, re-

parameterized convolution. In reality, RepConv integrates identity connection, 3 × 3 convolution, and 1 × 1 convolution into a single convolutional layer. The identity connection in RepConv eliminates the residual in ResNet and the concatenation in DenseNet, providing greater diversity of gradients for various feature maps, according to the analysis of the combination and corresponding performance of RepConv and other designs. Because of this, RepConv is utilized without identity connection (RepConvN) to construct the architecture of proposed re-parameterized convolution.

- YOLOv8 [19]: The modular and scalable architecture of YOLOv8 is one obvious advancement. The head, neck, and backbone make up the three primary components of the model. The backbone includes CSPDarknet53 and EfficientDet and is in charge of extracting features from the input image. The neck, which joins the head and backbone, is essential for the fusion of features. Bounding boxes, object classes, and confidence ratings are all predicted by the head. PANet (Path Aggregation Network), a feature pyramid network that enables information flow across various scales, is introduced by YOLOv8. The model can handle objects with different scales more effectively thanks to PANet. The defining element of the YOLO series—the YOLO head—remains in YOLOv8. Based on the features that the neck architecture and backbone network have retrieved; this component makes predictions. For every anchor box connected to a grid cell, the YOLO head forecasts bounding box coordinates, objectness scores, and class probabilities. Anchor boxes are used by the architecture to forecast items of various sizes and shapes with efficiency.

  YOLOv8 enhances model performance and convergence speed by utilizing recent developments in training methodologies. To improve the model's generalization skills, linear interpolations of the images are produced using MixUp, a data augmentation technique. More stable convergence is also achieved by YOLOv8's use of a cosine annealing scheduler for learning rate modifications during training.

- YOLOv9 [24]: Wang et al, 2024 based it on YOLOv7. The concept of programmable gradient information (PGI) is proposed, it is able to provide complete input information for the targeted task with the goal to calculate objective function. This way, the reliable gradient information can be obtained to update the network weights. Also, it utilizes a new lightweight architecture, the Generalized Efficient Layer Aggregation Network (G-ELAN) that is based on gradient path planning. In order to achieve greater parameter usage than the state-of-the-art techniques created based on depth-wise convolution, GELAN solely uses ordinary convolution operations. PGI uses an auxiliary reversible branch to generate consistent gradients, allowing the deep features to retain important properties necessary for carrying out the intended task. Semantic loss that could result from integrating multi-path characteristics in a standard deep supervision process can be prevented by designing an auxiliary reversible branch. Stated differently, we are optimizing training outcomes by programming gradient information propagation at several semantic levels. By allowing for the free selection of a loss function that is appropriate for the intended job, PGI also solves the issues that mask modeling presents.

  PGI resolves the issue that deep supervision is limited to very deep neural network topologies, making it possible to really implement new lightweight architectures in everyday situations. In order to accomplish a higher parameter consumption than the depth-

wise convolution design based on the most sophisticated technology, the GELAN that we built just employs traditional convolution, while demonstrating excellent advantages of being light, fast, and accurate. The object detection performance of the YOLOv9 on the MS COCO dataset, when combining the suggested PGI and GELAN, significantly outperforms the current real-time object detectors in every way.

The Res2Net module merges all converted partitions before passing them backwards and joins various input partitions in a hierarchical fashion with the following partition. In order to obtain all of the original information, CBNet reintroduces the original input data using a composite backbone. It then uses a variety of composition techniques to generate several layers of multilayer reversible information. Although these network designs often have very good parameter usage, the additional composite layers result in sluggish inference times. Combining CBNet with the highly effective real-time object detector YOLOv7, DynamicDet achieves an excellent trade-off between speed, number of parameters, and accuracy.

Deep Neural Networks present problems when it comes to convergence, where it is either slow or it shows poor results. The primary source of this issue is because, shortly after being transmitted, the initial gradient—which came from a very deep network—lost a great deal of information that was necessary to accomplish the aim. We feedforward deep networks with various topologies and beginning weights to validate this inference; the resulting graphic visualizes and illustrates them. It is evident that PlainNet has lost a great deal of crucial data needed for deep layer object detection. Regarding the proportion of significant data that ResNet, CSPNet, and GELAN can retain, there is a positive correlation between it with the accuracy that can be attained during training.

PGI suggests an auxiliary reversible branch to update network parameters and produce dependable gradients. The loss function can offer direction and prevent the chance of discovering misleading correlations from incomplete feedforward features that are less important to the target by giving information that maps from data to targets. We suggest using reversible architecture to maintain entire information, however doing so will result in significant inference costs when adding the main branch. The findings indicate that a 20% increase in the number of connections from deep to shallow layers will result in a longer inference time. The inference time even goes over twice as long when we keep adding input data to the network's high-resolution processing layer (yellow box).

Different feature pyramids can be employed for different purposes in object identification; for instance, when combined, they can detect objects of various sizes. The system will therefore consider the locations of items of different sizes to be the background once it has connected to the deep supervision branch and is guiding the shallow features to learn the features needed for small object detection. But the action above will result in a significant loss of information for the deep feature pyramids needed to anticipate the target item. Regarding this matter, we think that in order for a subsequent main branch to learn predictions for different targets, it needs to get knowledge about all target objects for each feature pyramid.

The generalized efficient layer aggregation network (GELAN) is created, which considers light weight, inference speed, and accuracy, by merging two neural network architectures: CSPNet and ELAN, which are constructed with gradient route planning. We expanded

ELAN's functionality, which was limited to convolutional layer stacking, to a new architecture that supports the usage of any kind of computational block.

- PP-YOLO [11]: It is an improved YOLOv3 model based on PaddlePaddle (PP-YOLO). The backbone uses the most common ResNet and for data augmentation the most basic MixUp. The manually set parameters following YOLOv3 are used. PP-YOLO improves the mAP on COCO from 43.5% to 45.2% at a speed faster than YOLOv4.

  The original backbone of YOLOv3, DarkNet-53 is replaced by ResNet-50-vd. A few convolutional layers in ResNet50-vd are replaced by deformable convolutional layers with the effectiveness of Deformable Convolutional Networks (DCN) being verified in many detection models.

- PP-YOLOv2 [10]: Its goal is to improve the effectiveness of YOLOv3 while still being able to maintain the inference speed. The baseline model is PP-YOLO which is an improved version of YOLOv3. FPN is used to create bottom-up paths, such as BiFPN, PAN, RFP and more. To compile the top-down data, we adhere to the PAN design.
  To keep the backbone unchanged, mish activation function is applied in the detection neck instead of the backbone. Also, the size of the input is increased meaning that the area of objects is enlarged. This way the information of images on a small scale can be more easily preserved, meaning that the performance will be increased. In addition to this, a soft label format is applied:

$$Loss = -t * \log\big(\sigma(p)\big) - (1-t) * \log(1-\sigma(p)).$$

  By replacing the loss function, the IoU aware branch shows better performance than previously.

  - t: The ground truth label for a given instance. It can take on a value of 1 or 0, indicating whether the instance belongs to the positive class or negative class, respectively.
  - p: The raw prediction output from the model for the given instance, often referred to as the logits. It can take on any real value.
  - σ(p): The sigmoid function applied to the raw prediction p. The sigmoid function maps the raw prediction to a probability between 0 and 1. It is defined as:

$$\sigma(p) = \frac{1}{1 + e^{-p}}$$

  - log: The natural logarithm, which is used to compute the log-probability.

- PP-YOLOE [9]: Like all YOLO algorithms, this is a one-stage object detector with excellent speed and accuracy. Motivated by YOLOX, the previous work of PP-YOLOv2 is enhanced. PP-YOLOE avoids the use of operators such as deformable convolution and Matrix NMS making it well supported on various hardware. After running tests, it was proven that PP-YOLOE can outperform YOLOv5 and YOLOX when it comes to both

speed and accuracy trade-off. Several FPN variations have been put forth recently to improve pyramid representation.

Its architecture consists of a backbone of ResNet50-vd with deformable convolution, a neck of PAN with SPP layer and DropBlock, and a head of the lightweight IoU. A ReLu activation function is applied in the backbone while a mish activation function is used in the neck. Similarly, to YOLOv3, it only assigns one anchor box for each ground truth object. It also uses IoU loss and IoU aware loss in addition to classification loss, regression loss, and objectness loss to boost PP-YOLOE's performance.

PP-YOLOE is Anchor-free and for the backbone and neck (originates from TreeBlock) is proposed which is a combination of residual and dense connections. It starts by making the basic TreeBlock simpler [9]. Then, as RMNet (Real Mode Network) [8] shows an approximate representation of the following actions, we substitute the elementwise add operation for the concatenation operation. This means that during the inference stage, the RepResBlack can be re-parameterized to a basic residual block that is used by ResNet-34 in the style of RepVGG.

The neck and backbone are constructed using the suggested RepResBlock. The backbone, called CSPrepResNet, is similar to ResNet in that it consists of one stem with three convolution layers and four stages layered by our RepResBlock after that. While creating the backbone, the ESE (Effective Squeeze and Extraction) layer is also utilized to enforce channel attention in each CSPRepResStage. Following PP-YOLOv2, the neck is constructed using the suggested RepResBlock and CSPRepResStage. A width multiplier α and depth multiplier β are used to scale the backbone and neck jointly, like seen in YOLOv5.

- YOLOX [14]: The chosen baseline is YOLOv3 with Darknet-53 as the backbone and an SPP layer (YOLOv3-SPP). An addition is made to the training strategies, adding EMA weights updating, cosine lr schedule, IoU loss and IoU-aware branch. YOLO's head is coupled and is now replaced by a decoupled one (essential to the end-to-end version of YOLO) which greatly improves the converging speed. Mosaic and MixUp are added to the augmentation strategies to boost the performance of YOLOX.

- YOLOX-PAI [14]: To make the use of several SOTA computer vision techniques easier, EasyCV is created, an all-in-one computer vision toolkit. EasyCV includes YOLOX-PAI, an enhanced version of YOLOX. As a straightforward yet effective object detection tool, YOLOX-PAI is presented in EasyCV (which includes the docker image, the model training, model evaluation, and model deployment processes).

  YOLOX-PAI uses a RepVGG-based backbone (like YOLOv6). To enhance YOLOX's performance in the neck of YOLOX-PAI, two techniques are employed: 1) Adaptively Spatial Feature Fusion (ASFF) (Liu et al., 2019) for feature augmentation 2) GSConv (Li et al., 2022), a lightweight convolution block to lower the compute cost, and its variance (denoted as ASFF_Sim). The head is enhanced with the attention mechanism as (Feng et al., 2021) to align the task of object detection and classification.

PAI-Blade is used, it is a robust inference optimization framework for model acceleration that automatically searches for the best method to optimize the input model. EasyCV allows the user to decide whether they want to export the model with the preprocess/postprocess procedure flexibly. Afterwards, a predictor api is given to perform efficient end2end object detection with only a couple of lines.

- POLY-YOLO [13]: Hurtík et al, 2020 based it on YOLOv3 and removed two of its weaknesses, large amounts of rewritten labels and inefficient distribution on anchors. By employing stairstep upsampling and a hypercolumn approach to aggregate data from a light SE-Darknet-53 backbone and provide a single scale output with high resolution, Poly-YOLO lessens the above problems.

  The first flaw of YOLOv3 is the low accuracy in big box identification, which is brought on by improper treatment of anchors in output layers. Because of the coarse resolution, the second one involves labels being rewritten by one another. Two of the main problems with YOLO are now resolved by Poly-YOLO thanks to a new feature decoder with a single output tensor that goes head-to-head with greater resolution: rewriting of labels and erroneous anchor distribution. A new feature that uses a bounding polygon representation to achieve instance segmentation is created. The maximum number of polygon vertices can be accurately changed to meet requirements.

  The issues with label rewriting and the uneven distribution of anchors across output scales impair YOLO's performance.

  High values of sk, or a scale multiplicator that represents the ratio of output resolution with respect to input resolution r, can suppress the first problem. The best scenario would be r = rsk, or sk = 1, which denotes identical input and output resolutions. There are two possible approaches to solving the second problem. The first method is to specify the three output scales' receptive fields together with the two thresholds that will divide them. Following that, centroids triplets—which are used as anchors—will be computed by k-means based on these thresholds. As a result, the problem-driven (receptive field) anchors would replace the data-driven ones. A single output layer is proposed with a high s1 scale ration (s1 = ¼) connected with all the anchors to solve both previously mentioned problems. The use of squeeze-and-excitation (SE) blocks in the backbone is the final suggested change made to YOLO's architecture. Less convolutional filters were used throughout the feature extraction stage as Yolo's primary benefit is speed. That is, it is configured to be 75% of the initial value. Additionally, the head and neck have 37.1M parameters collectively, which is substantially less than YOLOv3's (61.5M).

- UTD-YOLOv5 [15]: An algorithm for real-time underwater detection, based on Attention Improved YOLOv5. Using a multi-stage cascade architecture of the framework, CSP2 and the self-attention mechanism are introduced to accomplish the extraction of high-order characteristics of coral reefs and CONTS. Furthermore, methods like iterative refining to locate CONTS.

  The visual channel attention mechanism (SE) and the random anchor box similarity calculation mechanism are in the input, backbone, neck, and head regions. Additionally, the original Backbone is swapped with CSP's two-stage cascade module (CSP2), which is

capable of extracting features of greater score. In addition to this, a multi-stage fusion optimization scheme is added to improve the network's performance. Anchor boxes at 12 scales are designed, and WBF is used to fuse the filtered prediction boxes.
UTD-YOLOv5 is split into two stages: the framework backbone Attention Improved YOLOv5 and multi-stage fusion optimization.

Attention Improved YOLOv5: Higher-order feature extraction is achieved by introducing CSP2, SE, and other modules into the Yolov5 network structure, which covers the input, backbone, neck, and head modules of the mainstream framework. Additionally, a visual channel and spatial channel attention mechanism are incorporated in accordance with CBAM (Figure 18) and carry out residual fusion using feature vectors that are weighted. Eventually, the CONTS category prediction and location regression in coral reefs are accomplished. YOLOv5 is innovated in the following perspectives: a) Low illumination: to perform multi-scale segmentation and identify differences in similar features, create an improved classifier in the preprocessing and Backbone, b) High turbidity: to filter the input, c) The small target (CONTS) and the high environmental similarity: to design structures that facilitate multi-scale similarity computation.

The backbone utilized is a two-stage cascaded CSP network (CSP2) that can extract higher-order features and fuse them from multiple scales. Furthermore, the convolution operation in the module connection is eliminated by the two-stage cascaded CSP2 network, which is directly coupled to a deep branch feature map. Without making the model larger, this process can gather additional data. In UTD-YOLOv5, this structure is commonly utilized. For the neck, the visual channel attention mechanism module SE is introduced. It is responsible for assigning weights to each channel through the three stages of Squeeze, Excitation and Scale. To obtain a larger receptive field, the last layer operation of YOLOv5's FPN network is changed to dilated convolution. For the head, random anchor box loss is utilized.

The WBF algorithm is provided to fuse and filter the produced multiple proposal boxes to satisfy the goals of real-time detection and speed up the UTD-YOLOv5 computation. In addition, an iterative refining plan is created to guarantee the best possible computational outcomes. The maximum value of iteration I = 3k.

- CST-YOLO [16]: Kang et al, 2023 based it on the YOLOv7 algorithm and it also leverages the Swin Transformer. In order to improve the receptive field of the model and extract the target feature information more effectively, a convolutional or CNN-Swin Transformer (CST) module based on Swin Transformer is constructed in the backbone and integrated into the YOLOv7 feature extraction module. Weighted ELAN (W-ELAN) is created, an adaptive feature fusion module that can constrain the incorrect feature map and dynamically fuse the feature map data with the effective feature map. To extract feature information from the input feature map with various receptive fields, we utilize a Multiscale Channel Split (MCS) module. To improve feature fusion performance and get superior target recognition through the fusion of several feature data sets, a module called Concatenation Convolutional Layers (CatConv) is developed.

  The CNN-Swin Transformer module uses two parallel 1 × 1 convolutions with the goal to adjust the number of channels of the input feature map to gather two outputs, one of which is used as the input to the Swin Transformer that obtains the global feature information of

the input feature map and then splices both outputs. The model's capacity to represent the input feature map and, consequently, its detection performance is significantly improved by the CST module's ability to extract features from the input feature map with varying receptivity. The Swin Transformer employs a regular window and a shift window, respectively, in its two levels of self-attention processes. In the previous layer, the input feature map is first split up into windows, and each window is then given its own self-attention mechanism without any information shared between them.

The weighted feature fusion in EfficientDet inspired the development of a Weighted ELAN (Figure 16) which is a new variant different from E-ELAN that can facilitate dynamic feature fusion.

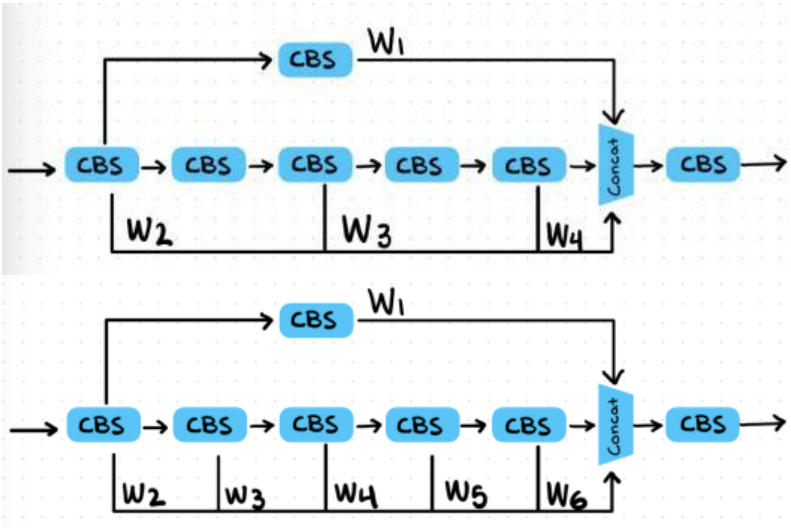



**Figure 16 The architectures of two variants of the Weighted ELAN module, W-ELAN 1 and W-ELAN 2 [16].**

CBS (Figure 16) refers to a specific type of convolutional block used in the architecture of the neural network. CBS stands for Convolutional Block with Batch Normalization and SiLU Activation. Here's a breakdown of each component:

- Convolutional (C): It involves applying convolutional filters to the input data to extract features.
- Batch Normalization (B): This technique normalizes the output of the convolutional layer to have a mean of zero and a standard deviation of one. Batch normalization helps in stabilizing and accelerating the training process by reducing internal covariate shifts.
- SiLU Activation (S): SiLU (Sigmoid Linear Unit), also known as Swish, is an activation function defined as

$$SiLU(x) = x * \sigma(x)$$

where $\sigma(x)$ is the sigmoid function. The SiLU activation function has been found to perform better than traditional activation functions like ReLU in certain deep learning models, particularly for improving convergence and performance.

To improve the model's capacity to fuse features, the MPConv structure of Yolov7's feature fusion section is altered by first swapping out the max pooling (MaxPool) function, which chooses the batch's maximum pixel value and creates CBSConcat in the backbone. Subsequently, we multiply the pertinent features from the down sampling section by two as the input to a Concatenate Convolutional Layers module in the neck in order to extract more feature information via convolution. To guarantee the uni-size of the feature map, we simultaneously modify its size using a convolution whose step is two prior to feature fusion. Furthermore, the detecting head in YOLOv7 is RepConv + IDetect. RepConv is essentially RepVGG without the identity link, and YOLOv6 and YOLOv6 v3.0 at training use the latter in their re-parameterizable backbone.

MPConv typically involves Mixed precision which refers to using a combination of 16-bit (half-precision) and 32-bit (single-precision) floating-point numbers during the computation. In MPConv, this means that some parts of the convolutional operation are performed using 16-bit precision, which requires less memory and computation compared to the standard 32-bit precision.

CBSConcat is a method where the outputs of multiple convolutional branches or paths are concatenated along a specific dimension (usually the channel dimension). This allows different features extracted by different convolutional paths to be combined into a single feature representation. A CBSConcat block integrates both concatenation and bilinear sampling techniques. It typically includes multiple convolutional branches that extract features at different scales or resolutions. These features are then concatenated and refined using bilinear sampling to produce a final feature representation that is robust to scale variations and spatial misalignments.

- YOLOv4-Tiny [21]: Lightweight version of YOLOv4 that is suitable for real-time detection on an embedded platform. When compared to YOLOv4, detection accuracy is slightly reduced but the model size is 1/10 of the original and detection speed is improved. Consists of 1) Backbone (CSPDarknet-Tiny a backbone network of feature extraction), 2) FPN, 3) YOLO head. Makes mistakes of undetected and false detection when detecting blocked/clustered items.

- LES-YOLO [21]: Based on YOLOv4-Tiny. Lightweight Enhance ShuffleNet (LESNet) was used instead of CSPDarknet-Tiny as the backbone. Squeeze-and-Excitation (SE) was added after contact of a different scale feature map. Only a 26 × 26 lightweight detection head was retained for pinecone prediction. The number of channels was compressed to 0.75 times (0.75×).

  Results: Compared with YOLOv4-Tiny, which is the fastest of the mainstream algorithms, speed was increased by 48.35% on the basis of improving the accuracy by 4.06%. The proposed model is smaller than 3 M, which is more suitable for deployment on low-power computing devices. Reduced the mistakes made by YOLOv4-Tiny and YOLOv5s when trying to detect blocked or clustered items. Parameters were only 1.17% and 0.86% compared to YOLOv5x and YOLOv4with close detection accuracy respectively.

Reduced the mistakes made by YOLOv4-Tiny and YOLOv5s(lightweight models) when detecting blocked/clustered items while maintaining similar performance when compared to heavy algorithms. After data augmentation AP increased by 3.02%.

With LESNet backbone:

- Parameters and FLOPs decreased by 52.13% and 55.52%, and FPS on the GPU reached 115.
- The SE mechanism achieved 0.84% AP improvement with negligible parameters and speed costs.
- The Recall and Precision increased by 2.08% and 2.48%, by using Focal loss and α-CIoU.
- Model size was reduced to 2905 KB without affecting the detection effect after channel compression (0.75×).
- Compared with YOLOv4-Tiny, the AP reached 95.33% (increase of 3.56%).
- Speed reached 135 FPS, 1.48 times of YOLOv4-Tiny.

- YOLOv5s-Ghost [22]: Wu et al, 2021 proposed the following method: Replace BottleneckCSP with Ghost Bottleneck which increases detection speed without reducing accuracy.
  Ghost Module steps: 1) convolution 2) Ghost generation 3) Feature map stitching.

  First, conventional convolution is used to get the eigenfeature map. Then the inherent feature map of each channel is processed by the Φ operation to generate the Ghost feature map (Φ is similar to 3×3 convolution). Finally, the intrinsic feature map obtained in the first step and the Ghost feature map obtained in the second step are connected to obtain the final result Output. Ghost Bottleneck mainly consists of two stacks of Ghost modules.

  Results: After 5000 iterations, the result of Yolo v5s-Ghost is slightly lower than the original result mAP, and the GIoU loss is declining steadily. The improved version only loses less than 3% of the mAP value compared to the original Yolo v5s. The Fps value has been increased from the original 28.57 to 47.62. The operation speed of the improved network model has been significantly improved. It can measure the distance between vehicles. The detected and actual distance results present only about 5% error.

- YOLO-FIRI [23]: Based on YOLOv5. Used to detect for small and weak objects in infrared images. Reliable and efficient for object detection in infrared images. Extends the thickness of the shallow CPS module, incorporating an improved SK attention module in the residual blocks to boost the feature extract ability and adds the detection layer to detect smaller objects. Three parts: 1) Backbone network with a lightweight feature extraction network, 2) Neck network to realize cross-stage feature fusion 3) Multiscale detection head. After multiscale fusion, the scales are: 128 × 128, 64 × 64, 32 × 32, 18 × 18.

  SK Attention Module: stands for Selective Kernel Attention, which is a mechanism designed to enhance the capability of convolutional neural networks (CNNs) to selectively attend to different spatial locations or feature channels based on their importance for a given task. This attention mechanism is particularly useful in

improving the performance of CNNs in tasks such as image classification, object detection, and semantic segmentation.

- MSFT-YOLO [20]: Guo et al, 2022 proposed to improve the scability of the model and the ability to capture contextual information. Hybrid model that integrates the CSPDarknet and Transformer modules. Enlarges the receptive field can predict multiple scale defects by expending the receptive field of convolution. Uses a combination of Transformer encoder block, multi-level feature fusion, dataset expansion, and some training techniques. Network structure is built based on YOLOv5l. Because the mAP of YOLOv5l is ahead of YOLOv5s and YOLOv5m 2.5% and 1.8%, and the FPS reaches 52.5 which has potential for real-time detection.

  Contains three parts:

  1. Backbone network: instead, the original convolution layers of YOLOv5, the self-developed TRANS structure is used which extends the reception domain of convolution by assembling it into CSPDarknet.
  2. Feature enhancement.
  3. Prediction.

  TRANS provides multi-level features with global information for detection and enhances the ability of MSFT-YOLO to recognize the background features of an image.

  Results: MSFT-YOLO can offers an improved time-performance by 7% compared to YOLOv5 (baseline model).

## 2.5. Summary

In summary, YOLO (You Only Look Once) object detection models have seen a remarkable evolution that has resulted in real-time, effective, and precise detection capabilities. The single-stage detection approach was first established by YOLOv1, and since then, improvements such as YOLOv2, YOLOv3, and YOLOv4 have consistently improved the architecture by introducing cutting-edge techniques to increase accuracy, speed, and versatility.

The real-time detection paradigm was invented by YOLOv1, which provided superior processing speeds and a reduced false positive rate over earlier techniques. Even with its shortcomings in terms of localization, YOLOv1 set the stage for later developments in object detection.

Significant gains in accuracy and speed were achieved with YOLOv2 and YOLOv3. While YOLOv3 incorporated multi-scale predictions and sophisticated feature fusion approaches, YOLOv2 introduced hierarchical object categorization and collaborative training algorithms. Thanks to these improvements, YOLO was able to achieve better results on common datasets such as COCO and PASCAL VOC, proving that it was a cutting-edge detection method.

YOLOv4 and YOLOv5 continued the trend of innovation, incorporating novel features such as Weighted-Residual-Connections, Cross-Stage-Partial-connections, and advanced data augmentation techniques like Mosaic Augmentation and MixUp. These models achieved

unprecedented levels of accuracy and efficiency, further solidifying YOLO's position as a leader in object detection.

The most recent versions, YOLOv6, YOLOv7, YOLOv8, and YOLOv9, have prioritized architectural improvements, scalability, and optimization. These models address issues with convergence speed, parameter usage, and inference efficiency by introducing novel ideas such programmable gradient information (PGI), Generalized Efficient Layer Aggregation Network (G-ELAN), and DynamicDet.

When it comes to the modified YOLO versions, the landscape of object detection algorithms has witnessed significant advancements with the introduction of various models such as PP-YOLO, PP-YOLOv2, PP-YOLOE, YOLOX, YOLOX-PAI, CST-YOLO, POLY-YOLO, UTD-YOLOv5, YOLOv4-Tiny, LES-YOLO, YOLOv5s-Ghost, YOLO-FIRI, MSFT-YOLO, and YOLO*C. Each of these models addresses different challenges and aims to improve upon existing architectures in terms of speed, accuracy, model size, and efficiency.

PP-YOLO and its variants, for instance, focus on improving accuracy while maintaining inference speed, leveraging techniques like deformable convolutions, mixup, soft label formats, and IoU-aware losses. Similarly, YOLOX and YOLOX-PAI aim to strike a balance between speed and accuracy by introducing anchor-free architectures and leveraging efficient backbones like RepVGG.

CST-YOLO and POLY-YOLO explore the fusion of convolutional and transformer architectures to improve feature extraction and capture contextual information effectively. UTD-YOLOv5, on the other hand, specializes in underwater object detection, utilizing attention mechanisms and multi-stage fusion optimization to extract high-order features.

Furthermore, lightweight models like YOLOv4-Tiny and LES-YOLO focus on real-time detection on embedded platforms, achieving significant reductions in model size and computational complexity while maintaining competitive accuracy. YOLOv5s-Ghost introduces novel modules like Ghost Bottleneck to increase detection speed without compromising accuracy.

Moreover, models like YOLO-FIRI cater to specific domains such as infrared object detection, utilizing specialized architectures and attention mechanisms to enhance detection performance in challenging environments.

Lastly, MSFT-YOLO and YOLO*C incorporate innovative features like hybrid CSPDarknet-Transformer architectures and dataset expansion techniques to improve scalability and capture contextual information effectively, achieving impressive results in terms of both accuracy and speed.

In conclusion, the development of YOLO object identification models is a noteworthy example of computer vision innovation and progress. Every version of YOLO, starting with YOLOv1 and continuing through YOLOv9, has expanded the realm of real-time object recognition capabilities, and opened new possibilities for a variety of industries, including augmented reality, autonomous driving, and surveillance. Overall, these advancements in object detection algorithms underscore the importance of continual research and innovation in the field, catering to diverse application scenarios and driving progress towards more robust, efficient, and accurate detection systems.

| Model | Architecture | Backbone | Neck | Head |
|---|---|---|---|---|
| **YOLOv1 [1]** | Fully convolution | Darknet pretrained on ImageNet | - | YOLOv1 head |
| **YOLOv2 [2][3]** | Fully convolution | Darknet-19 | - | YOLOv2 head |
| **YOLOv3 [6]** | Feature pyramid network (FPN) | Darknet-53 | FPN | YOLOv3 head |
| **YOLOv4 [5]** | CSP (cross-stage partial) architecture | CSPDarknet-53 | SPP and PANet | YOLOv3 head |
| **YOLOv5 [7]** | CSP (cross-stage partial) architecture | CSPDarknet-53 | PANet | YOLOv3 head |
| **YOLOv6 [17]** | RepPAN | CSPStackRepblock | Rep-PANet | Decoupled head |
| **YOLOv7 [18]** | Compound scaling method | E-ELAN RepConv 3-Stacked ELAN<br><br>CSPDarknet | FPN and PANet | Multiple heads |
| **YOLOv8 [19]** | Feature pyramid network | Modified CSPDarknet-53 | PANet and/or FPN | YOLOv8 head |
| **YOLOv9 [24]** | Advanced scaling method | Advanced CSPDarknet | Enhanced FPN and PANet | YOLOv9 head |
| **PP-YOLO [11]** | Improved YOLOv3 | ResNet50 | PANet | YOLOv3 head |
| **PP-YOLO2 [10]** | Improved PP-YOLO | ResNet101 | PANet | YOLOv3 head |
| **PP-YOLOE [9]** | Enhanced PP-YOLO2 | CSPResNet | PANet and FPN | YOLOv4 head |
| **YOLOX [14]** | YOLO with anchor-free mechanism | CSPDarknet | PANet | YOLOX head |
| **YOLOX-PAI [14]** | Optimized YOLOX | CSPDarknet | PANet | YOLOX head |
| **CST-YOLO [16]** | Contextualized Scene Text detection | TextDetectorBackbone | TextDetectorNeck | TextDetectorHead |
| **POLY-YOLO [13]** | Polygon-based YOLO | Darknet-53 | FPN and PANet | YOLOv3 head |
| **UTD-YOLO [15]** | Unified Text Detection YOLO | UnifiedBackbone | UnifiedNeck | UnifiedHead |
| **YOLOv4-Tiny [21]** | Light-weight YOLOv4 | Tiny CSPDarknet | Tiny PANet | YOLOv4-Tiny head |

| | | | | |
|---|---|---|---|---|
| **LES-YOLO [21]** | Light-weight Efficient YOLO | EfficientNet | Lightweight PANet | YOLOv3 head |
| **YOLOv5s-Ghost [22]** | YOLOv5 small with GhostNet | GhostNet | PANet | YOLOv5s head |
| **YOLO-FIRI [23]** | Fine-tuned YOLO for IR Images | InfraResNet | IR PANet | YOLOv3 head |
| **MSFT-YOLO [20]** | Microsoft optimized YOLO | CSPResNeXt | FPN and PANet | MSFT YOLO head |

**Table 4 Summary of YOLO comparison**

| Model | Input Size | Notable Features | Summary | Year |
|---|---|---|---|---|
| **YOLOv1 [1]** | 448×448 | First version, single neural network | Introduced real-time object detection with a single convolutional network. Balanced speed and accuracy but had limitations in localization accuracy and recall. | June 2016 |
| **YOLOv2 [2][3]** | 416×416 | Batch normalization, high-resolution classifier | Achieved better accuracy with batch normalization and high-resolution input. Introduced anchor boxes and improved recall and localization, making it more robust for diverse datasets. | July 2017 |
| **YOLOv3 [6]** | 608×608 | Multi-scale predictions, darknet-53 backbone | Enhanced detection with multi-scale predictions, allowing for better handling of small objects. Improved architecture with Darknet-53 but at the cost of reduced speed compared to previous versions. | April 2018 |
| **YOLOv4 [5]** | 608×608 | CSPDarknet53, Mish activation, mosaic data augmentation | Significant improvements in speed and accuracy with CSPDarknet53 and new training techniques like Mosaic augmentation. Maintained real-time performance with better accuracy on larger datasets. | April 2020 |
| **YOLOv5 [7]** | 640×640 | Auto learning bounding box anchors, new augmentation techniques | Dramatic increase in speed and efficiency, with a smaller model size. Enhanced usability with better training techniques and simplified deployment. Excellent balance of speed and accuracy. | June 2020 |
| **YOLOv6 [17]** | 640×640 | EfficientRep backbone, PAN-FPN neck, improved training strategies | Introduced a more efficient backbone and improved training strategies. Maintained high speed while achieving good accuracy, suitable for real-time applications. | September 2022 |
| **YOLOv7 [18]** | 640×640 | Extended efficient layer aggregation networks, compound model scaling | Focused on optimizing layer aggregation and scaling models effectively. Balanced performance with improvements in both speed and accuracy over previous versions. | July 2022 |

| | | | | |
|---|---|---|---|---|
| **YOLOv8 [19]** | 640×640 | Enhanced architecture with better feature extraction | State-of-the-art accuracy with further improved feature extraction techniques. Maintained high speed, making it suitable for various real-time applications. | January 2023 |
| **YOLOv9 [24]** | 640×640 | Advanced attention mechanisms, CSP optimization | Leveraged advanced attention mechanisms and further CSP optimizations for improved accuracy and speed. | February 2024 |
| **PP-YOLO [11]** | 416×416 | PANet, ResNet50 backbone | Improved upon YOLOv3 with PANet and ResNet50 backbone for better feature fusion and accuracy. | July 2020 |
| **PP-YOLO2 [10]** | 512×512 | PANet++, ResNet101 backbone | Enhanced PP-YOLO with PANet++ and ResNet101 backbone for further improvements in detection accuracy and robustness. | April 2021 |
| **PP-YOLOE [9]** | 640×640 | Enhanced PANet, CSPResNet backbone | Further improvements with enhanced PANet and CSPResNet backbone, providing better accuracy and efficiency. | March 2022 |
| **YOLOX [14]** | 640×640 | Anchor-free mechanism, Dynamic label assignment | Introduced anchor-free mechanism and dynamic label assignment for simplified training and improved performance. | July 2021 |
| **YOLOX-PAI [14]** | 640×640 | Optimized data augmentations, Efficient training strategies | Optimized version of YOLOX with improved data augmentations and efficient training strategies for better accuracy and speed. | August 2022 |
| **CST-YOLO [16]** | 608×608 | Contextual text detection, Efficient feature fusion | Specialized for contextual text detection with efficient feature fusion techniques. | June 2023 |
| **POLY-YOLO [13]** | 608×608 | Polygon-based detection, Improved localization | Introduced polygon-based detection for better handling of irregular shapes and improved localization accuracy. | May 2020 |
| **UTD-YOLO [15]** | 608×608 | Unified text detection, Specialized neck architecture | Focused on unified text detection with a specialized neck architecture for better text recognition in various scenarios. | November 2023 |
| **YOLOv4-Tiny [21]** | 416×416 | Lightweight model, Reduced parameters | Lightweight version of YOLOv4 with reduced parameters, suitable for real-time applications on resource-constrained devices. | April 2020 |
| **LES-YOLO [21]** | 416×416 | EfficientNet backbone, Lightweight PANet | Lightweight and efficient model with EfficientNet backbone and lightweight PANet for high-speed detection. | February 2023 |
| **YOLOv5s-Ghost [22]** | 416×416 | GhostNet integration, Reduced computation | Integrated GhostNet to reduce computation while maintaining accuracy, resulting in a smaller and faster model. | June 2020 |

| | | | | |
|---|---|---|---|---|
| **YOLO-FIRI [23]** | 512×512 | Infrared image optimization, Specialized loss function | Optimized for infrared image detection with a specialized loss function for better performance in IR image scenarios. | October 2021 |
| **MSFT-YOLO [20]** | 608×608 | Optimized for cloud deployment, CSPResNeXt backbone | Microsoft-optimized version for cloud deployment with CSPResNeXt backbone for improved accuracy and efficiency. | May 2022 |

**Table 5 Summary of YOLO comparison**

## 2.6. Limitations

Bounding box predictions are subject to high spatial limits due to YOLO, as each grid cell can only contain one class and forecast two boxes. The quantity of surrounding items that the model can predict is restricted by this spatial limitation. The model has trouble handling tiny objects that show up in groups, such as birds in a flock individually. Also, it finds it difficult to generalize to objects in novel or unusual aspect ratios or configurations since it learns to estimate bounding boxes from data. Finally, the loss function tends to treat errors the same way in small bounding boxes versus large bounding boxes. A small error in a small bounding box has a much greater effect on IOU compared to a small error in a large bounding box. A comparison of YOLO algorithms' error analysis with Fast R-CNN's reveals that they produce a considerable amount of localization errors. Moreover, the YOLO algorithms possess a lower recall than methods based on region proposals.

# CHAPTER 3 METHODOLOGY

The methodology that is employed in this research is aimed analyzing various YOLO algorithms and refining one to enhance its effectiveness, efficiency, and innovation in object detection. YOLO algorithms are renowned for their robust capabilities in fast and precise real-time object detection. However, each iteration has its distinct advantages and limitations. This research is focused on evaluating these different iterations to understand their performance nuances comprehensively. This allowed us to pinpoint specific areas that needed improvement. By implementing tailored enhancements, the goal is to advance an optimized version of the algorithm that not only boosts detection accuracy and speed but also integrates innovative features that pave the way for sophisticated applications in artificial intelligence.

## 3.1. YOLOv5

Yolov5 has the advantage of fast convergence, high precision, and strong customization. It presents real-time processing capabilities and low hardware computing requirements, meaning that it can easily be transplanted to mobile devices.

It uses three types of output feature maps to detect objects with different sizes and uses eight down-sampling output feature maps to detect small objects. In addition to this, it calculates suitable anchors according to each dataset. This makes it easier for the model to learn the converge and predict objects with different scales.

Finally, it is made up of Backbone: SPP layer, Neck: PANet, Head: YOLO detection. It has a total of three prediction heads which can detect three sizes:

- 80 × 80: large objects
- 40 × 40: medium objects
- 20 × 20: small objects

Yolov5 presents a good balance between speed, accuracy, and efficiency, something that makes it a good choice for any object detection task. But it cannot detect small objects with ease.

The neck of Yolov5 (it connects the backbone and the head) presents PANet, which incorporates bottom-up path augmentation on the basis of FPN. This gives it the ability to detect small objects by fusing features of low and high layers to obtain high-resolution and good semantic features. But, Yolov5 has 1 × 1 convolution to reduce the number of channels at the beginning of the neck meaning that calculation efficiency improves. This also means that the channel information is now reduced, which leads to the poor performance of PANet (Figure 17).

The loss function for the Yolov5 algorithm is made up of three loss values: class probability score, confidence level score, and bounding box regression score. The sum of all three is equal to the overall loss. The following are functions for confidence loss, classification probability loss, and the ones previously mentioned.

- $\text{Loss}_{\text{obj}} = \text{Loss}_{\text{class}} + \text{Loss}_{\text{box}} + \text{Loss}_{\text{conf}}$
- $L_{class} = -\sum_{i=0}^{S*S} I_{ij}^{obj} \sum_{c \in classes} [\widehat{P_i^j} \, log P_i^j + (1 - \widehat{P_i^j}) \log (1 - \widehat{P_i^j})]$
- $L_{box} = 1 - CIoU$
- $L_{conf} = -\sum_{i=0}^{S*S} I_{ij}^{obj} \left[\widehat{M_i^j} log M_i^j + \left(1 - \widehat{M_i^j}\right) \log\left(1 - M_i^j\right)\right] - \lambda_{noobj} \sum_{i=0}^{S*S} \times \sum_{j=0}^{N} I_{ij}^{noobj} \left[\widehat{M_i^j} log M_i^j + \left(1 - \widehat{M_i^j}\right) \log\left(1 - M_i^j\right)\right]$

| | |
|---|---|
| $\boldsymbol{Loss_{class}}$ | Classification loss |
| $\boldsymbol{Loss_{box}}$ | Bounding box regression loss |
| $\boldsymbol{Loss_{conf}}$ | Confidence loss |
| **S × S** | The grid size of the image |
| $\boldsymbol{I_{ij}^{obj}}$ | An indicator function that is 1 if the object is present in cell i,ji, ji,j, and 0 otherwise. |
| $\boldsymbol{\widehat{P_i^j}}$ | The ground truth probability that an object of class c is in cell i,ji, ji,j. |
| $\boldsymbol{P_i^j}$ | The predicted probability that an object of class c is in cell i,ji, ji,j |
| $\boldsymbol{\widehat{M_i^j}}$ | The ground truth confidence score indicating the presence of an object in cell i,ji, ji,j. |
| $\boldsymbol{M_i^j}$ | The predicted confidence score for cell i,ji, ji,j. |
| $\boldsymbol{I_{ij}^{noobj}}$ | An indicator function that is 1 if no object is present in cell i,ji, ji,j, and 0 otherwise. |
| $\boldsymbol{\lambda_{noobj}}$ | A weighting factor that down-weights the loss from cells where no object is present to avoid overwhelming the loss from cells with objects. |

**Table 6 Loss function variables**

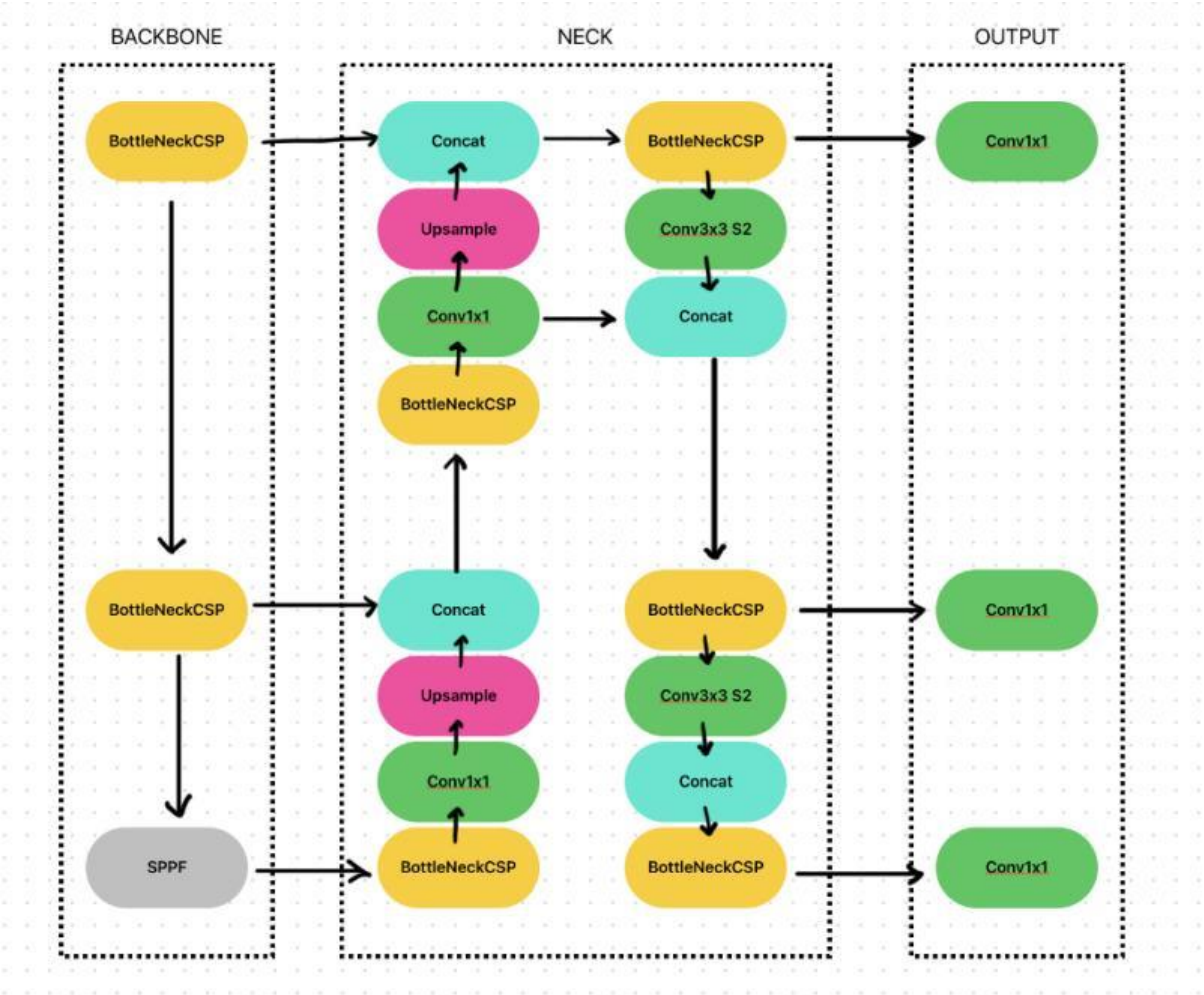



**Figure 17 YOLOv5 Architecture [40]**

## 3.2. HIC-YOLOv5

The algorithm HIC-YOLOv5 (Figure 19) was developed by Tang et al. [27] and presented great improvement compared to YOLOv5 when tested with the VisDrone-2019 dataset. More specifically, it showed an increase in mAP compared to the original YOLOv5 algorithm. This was achieved by adding an extra prediction head for small object detection (SODH), by adding an involution block between the backbone and neck (CFFI), and by adding an attention mechanism at the end of the backbone to decrease computational cost (CBAM (Figure 18)).

The combination of these three innovations—SODH, CFFI, and CBAM—enables HIC-YOLOv5 to outperform the original YOLOv5 algorithm, particularly in scenarios involving small object detection. The specialized prediction head for small objects ensures that fine details are not overlooked, while the involution block enhances feature extraction and fusion capabilities. Additionally, the attention mechanism optimizes computational efficiency by concentrating the model's resources on the most critical features.

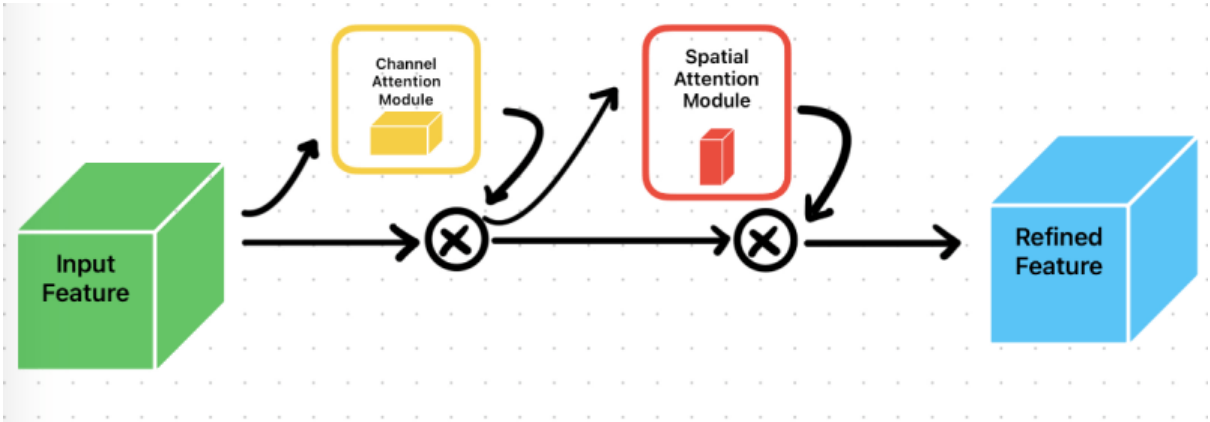



**Figure 18 CBAM Structure**

HIC-Yolov5 proves to be good at small object detection as well as recognizing each item as separate instead of one big cluster. Due to this, it was chosen to detect polyps which tend to be small anomalies in the gastrointestinal (GI) tract.

1) Loss Function: The loss function is made up of three parts, objectness, bounding box probability, and class probability. It can be calculated using the following function:

$$Loss = \alpha Loss_{obj} + \beta Loss_{box} + \gamma Loss_{cls}$$

| | |
|---|---|
| $Loss_{class}$ | Classification loss |
| $Loss_{box}$ | Bounding box regression loss |
| $Loss_{conf}$ | Confidence loss |

**Table 7 Types of loss functions**

2) Data Augmentation: Data augmentation is used to enhance the robustness of our model. In the previously mentioned model, it included Mosaic, Random affine, HSV augmentation, Mix Up, and Cutout. In HIC-Yolov5, an extra center cropping is added to the already existing data augmentation techniques.

3) Prediction Head: HIC-Yolov5 uses an additional prediction head compared to Yolov5, the Small Object Detection Head (SODH). This detection head is specifically meant to locate small objects in an image.

4) Channel Feature Fusion with Involution (CFFI): An involution block is added between the neck and the backbone, then the channel information is improved, something that results in the reduction of information loss during the FPN process. This in return results in the enhanced performance of FPN, which benefits the detection of small-sized objects.

5) Attention Mechanism: An attention mechanism is added in the end of the backbone to decrease computational cost.

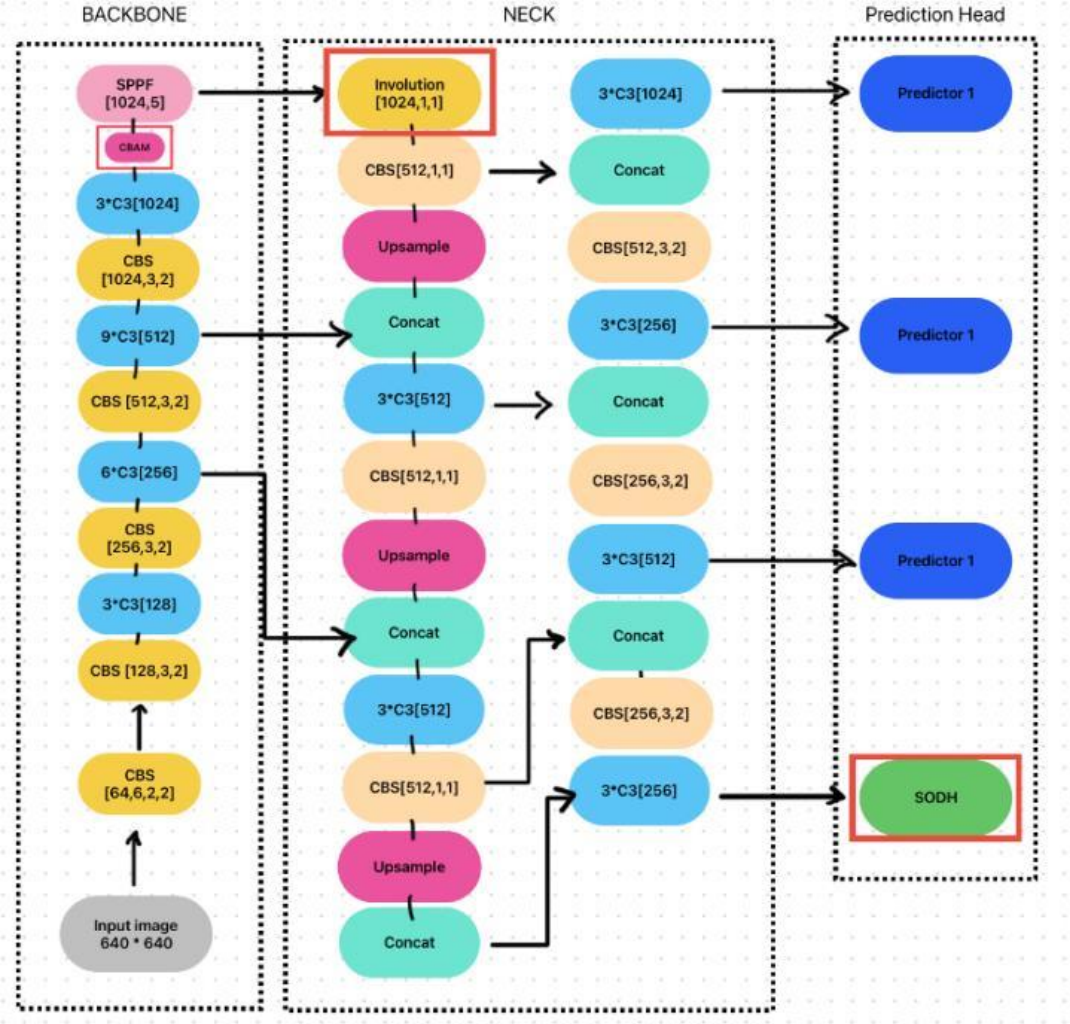



**Figure 19 HIC-YOLOv5 Architecture [39]**

In conclusion, HIC-YOLOv5 represents a notable advancement in object detection technology, with particular strengths in detecting small objects. These improvements make it a valuable tool for applications requiring precise and efficient object detection in complex environments.

## 3.3. EfficientNet

EfficientNet's [51] architecture is built upon principles similar to those used in MnasNet, with the key difference being that EfficientNet-B0 is slightly larger due to its higher FLOPS (floating-point operations per second) target. This design choice allows EfficientNet to achieve a better balance between accuracy and computational efficiency.

At its core, EfficientNet is based on the inverted bottleneck residual blocks introduced in MobileNetV2. These blocks help in reducing the computational cost while maintaining high accuracy by efficiently capturing spatial features through depth wise separable convolutions. Additionally, EfficientNet incorporates squeeze-and-excitation (SE) blocks, which adaptively recalibrate channel-wise feature responses, further enhancing the network's representational power.

EfficientNet is both a convolutional neural network (CNN) architecture and a scaling method designed to uniformly scale all three dimensions of a network: depth (the number of layers), width (the number of channels in each layer), and resolution (the input image size). This is achieved by using a compound coefficient that systematically scales these dimensions based on a fixed set of scaling coefficients.

The scaling method works as follows:

- Network Width: This refers to the number of channels in each layer. By scaling the width, the model can capture more fine-grained features.
- Network Depth: This refers to the number of layers in the network. By scaling the depth, the model can capture more complex patterns.
- Input Resolution: This refers to the size of the input images. By scaling the resolution, the model can work with higher detail images, improving accuracy on tasks requiring fine detail recognition.

It's compound scaling method provides a systematic way to scale up models in a balanced manner, ensuring that computational resources are utilized effectively. This approach contrasts with traditional methods that might scale only one dimension (such as depth) at a time, which can lead to suboptimal performance.

EfficientNet-B0, the baseline model, was designed with a specific FLOPS target, which makes it slightly larger than MnasNet. However, the model family includes a range of models from EfficientNet-B0 to EfficientNet-B7, each scaled up progressively to achieve higher accuracy with increasing computational resources. Each subsequent model in the EfficientNet series (B1 to B7) is scaled using the compound coefficient, ensuring that the models grow in a balanced manner, which contributes to their efficiency and performance on various benchmark tasks.

Overall, EfficientNet represents a significant advancement in neural network design, combining innovative architectural elements with an effective scaling strategy to create models that are both powerful and efficient.

## 3.4. Transformers

A transformer model (Figure 20) is a neural network that can learn context by tracking relationships in sequential data such as words in a sentence. These models are able to apply an evolving set of mathematical techniques, either attention or self-attention, with the goal of detecting subtle ways that elements in a series can depend on each other. An attention mechanism is the only method used by a Transformer model architecture to identify global dependencies between input and output, as opposed to recurrence.

Transformers can be found when translating text and speech in real-time, aiding hearing-impaired listeners in classrooms and conferences. They can also be used by researchers to help them understand the chains of DNA genes as well as amino acids in proteins with the goal of improving speed drug design. In 2019 it was noticed that since its publication in 2017 [52], 70 percent of arXiv papers concerning AI mentioned transformers.

Before transformers were invented, neural networks had to be trained using immense, labor-intensive, labeled datasets. Transformers reduce that necessity by statistically identifying patterns between parts, making available the trillions of photos and petabytes of text data stored in corporate systems and on the web.

Key Features of Transformer Models:

- Self-Attention: The core innovation of the transformer model is the self-attention mechanism, which computes the relevance of each element in the input sequence to every other element. This enables the model to weigh the importance of different words (or elements) dynamically, capturing long-range dependencies and contextual nuances that are critical for understanding the overall meaning.
- Global Dependencies: Unlike RNNs, which rely on sequential processing and can struggle with long-range dependencies, transformers can identify global dependencies between inputs and outputs simultaneously. This is achieved without the need for recurrence, making transformers more efficient and scalable for large datasets.

Transformer models represent a revolutionary advancement in neural network architecture, enabling the efficient and accurate analysis of sequential data. Their ability to capture complex dependencies through attention mechanisms has led to significant improvements in various applications, from real-time translation and assistive technologies to genomic research and AI development. By reducing the need for labeled data and leveraging vast amounts of available information, transformers have fundamentally transformed the landscape of machine learning and artificial intelligence.

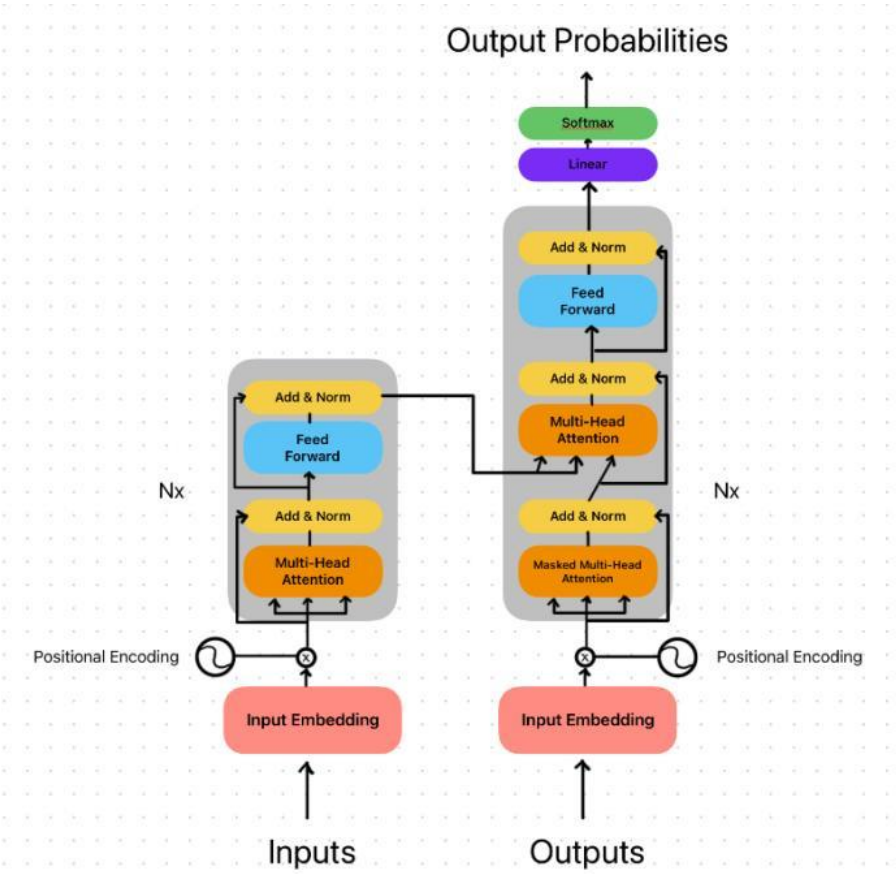



**Figure 20 Transformer Model Architecture [52]**

## 3.5. YOLOv5-TST

In 2024, Yoo et al. [4] created a state-of-the-art algorithm whose goal is to improve the already existing YOLO in polyp detection when it comes to detection speed and accuracy. They claim that as of recently, computer-aided detection (CADe) systems are of great help and are able to enhance the efficiency and accuracy of polyp detection. The only drawback is that CADe systems that are based on CNNs usually demand significant computational recourses, something that makes them unsuitable for deployment in resource-restrained environments. That is why they propose a lightweight polyp detection algorithm that combines a Transformer layer the YOLO algorithm named YOLOv5m-TST (Figure 21). This presents a reduced computational complexity and number of parameters without compromising the performance of detection.

This model integrates a Transformer layer into the YOLO (You Only Look Once) architecture, specifically optimizing the neck part responsible for feature fusion and rescaling. The main objective is to reduce computational complexity and the number of parameters without sacrificing detection performance. This makes the model more suitable for deployment in resource-constrained environments, potentially enhancing accessibility to advanced diagnostic tools in medically underserved regions.

The model is structured with a backbone for feature extraction, a neck for feature aggregation, and a head for the final detection task. The architecture integrates convolutional and transformer layers to handle both local and global information effectively. The model's balance between accuracy and computational efficiency, achieves impressive results in terms of recall, precision, and F1 scores, while maintaining low latency for real-time application. This ensures that the system can enhance polyp detection rates during colonoscopies without imposing significant computational demands, making it suitable for clinical use. Also, in the convolution block, batch normalization is performed after the dimension transformation.

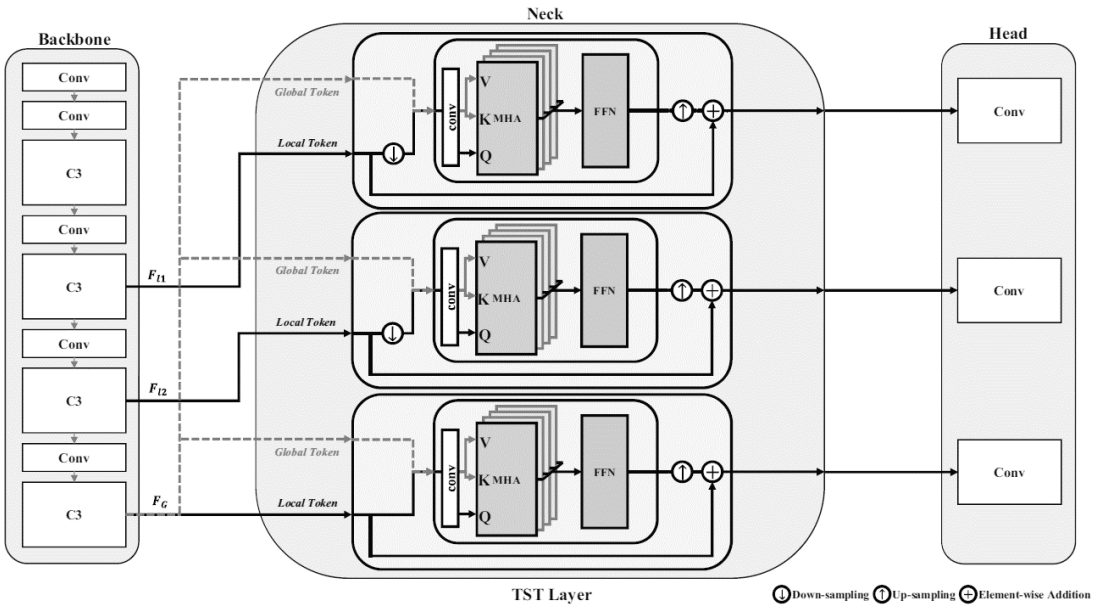



**Figure 21 YOLOv5m-TST Architecture [4]**

When it comes to testing the network, three datasets were chosen: SUN, KUMC [41], and KVASIR-SEG [29]. Each dataset was split in an 8:2 training-testing ratio. Results first showed that when YOLOv5m-TST was compared with the original YOLOv5m, YOLOv5m-Tiny, and ENDOMIND, YOLOv5m-TST performed the best out of all four algorithms using each dataset separately.

Then, the Transformer was also implemented into the YOLOv3 and YOLOv8m algorithms and along with the original YOLOv5m they were compared against the YOLOv3-TST, YOLOv5m-TST, and YOLOv8m-TST algorithms using the combined datasets of SUN, KUMC, and KVASIR-SEG for the input size of 320 × 320, 384 × 384, and 480 × 480. Results showed that once again, the versions of the algorithms where the Transformer had been implemented performed better than the original versions.

## 3.6. Proposed Methodology

In this work, we will be proposing an innovative version of YOLOv5 that uses EfficientNet as the backbone and also includes a Feature Pyramid Network (FPN) [55] and Squeeze-and-Excitation Blocks (SE Blocks). The usage of EfficientNet contributes to the reduction of free parameters and the reduction of complexity regarding the floating-point operations per second (FLOPs), its execution utilizes PyTorch. FPN is chosen because it enhances YOLO's capability to detect objects of various sizes by providing multi-scale feature maps, improving small object detection, and enriching contextual information, which collectively leads to better overall detection performance. Lastly, by adding SE Blocks to YOLO improves the model's performance by enhancing its ability to capture important features while reducing less relevant information. SE blocks, introduced in the SE-Net architecture, work by adaptively recalibrating channel-wise feature responses.

When it comes to FPN [55], it is a feature extractor that uses an arbitrary-sized single-scale image as input and produces many levels of correspondingly scaled feature maps using a fully convolutional architecture (Figure 22). The convolutional architectures at the backbone have no bearing on this procedure. As such, it serves as a general method for constructing feature pyramids within deep convolutional networks for applications such as object detection.

There are two pathways involved in building the pyramid: a top-down and a bottom-up pathway. The feedforward computation of the mainframe ConvNet, which computes a feature hierarchy made up of feature maps at various scales with a scaling step of two, is the bottom-up approach. Each stage of the feature pyramid has a single pyramid level defined for it. Each stage's final layer's output serves as a reference set of feature maps. The feature activations produced by the final residual block of each stage for ResNets are employed. The top-down approach upsamples geographically coarser but semantically stronger feature mappings from higher pyramid levels, creating the illusion of greater resolution features. Lateral connections are used to add features from the bottom-up pathway to these already existing features. Each lateral link combines feature maps from the top-down and bottom-up pathways that have the same spatial size. Because the bottom-up feature map was subsampled fewer times, its activations are more precisely localized while having lower-level semantics.

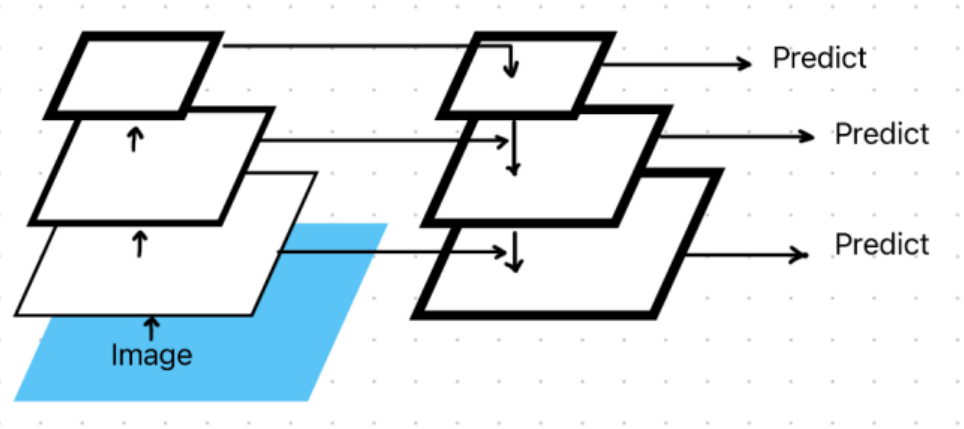



**Figure 22 Feature Pyramid Network**

Furthermore, the SE Block is an architectural block that allows the network to dynamically undertake channel-wise feature recalibration, hence increasing its representational power (Figure 23). The method is:

- The input of the block is a convolutional block.
- Using average pooling, each channel is "squeezed" into a single numerical value.
- The output channel complexity is lowered by a ratio and non-linearity is added by a dense layer and ReLU.
- A smooth gating function is provided for each channel by a sigmoid and another dense layer.
- In the end, a weight is assigned to every feature map in the convolutional block according to the "excitation" of the side network.

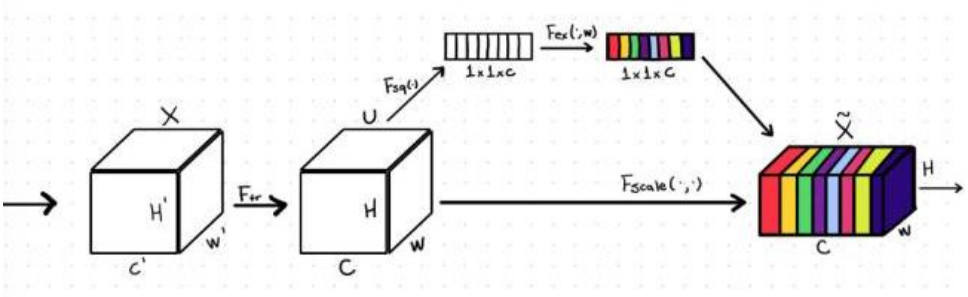



**Figure 23 SE Block**

Lastly, this method applies BottleneckCSP, which further refines the features by applying a bottleneck structure after a succession of convolutions and transformations to the input that were retrieved from earlier layers. This process improves the network's representational capability and makes it easier for users to identify increasingly intricate patterns in the feature maps.

# CHAPTER 4 EXPERIMENTS AND RESULTS

## 4.1. Datasets

### 4.1.1. KVASIR-SEG Dataset

The KVASIR-SEG [29] dataset contains a total of 1000 high-resolution images of polyps captured during colonoscopy procedures. Each image is accompanied by a corresponding ground truth mask that shows the polyp regions, providing a clear and accurate reference for segmentation tasks. The image resolution can vary from 332 × 487 to 1920 × 1072 pixels. The polyp images and segmentation masks (Figure 24) have been verified by an experienced gastroenterologist concerning their accuracy. The dataset consists of one (1) class named "Polyp", meaning "Polyp" = 0. The dataset was split into two folders, "images" and "masks", each consisting of 1,000 JPG files. It is an open-access dataset.

The dataset was split up into: 65% train, 20% validation, and 15% test.

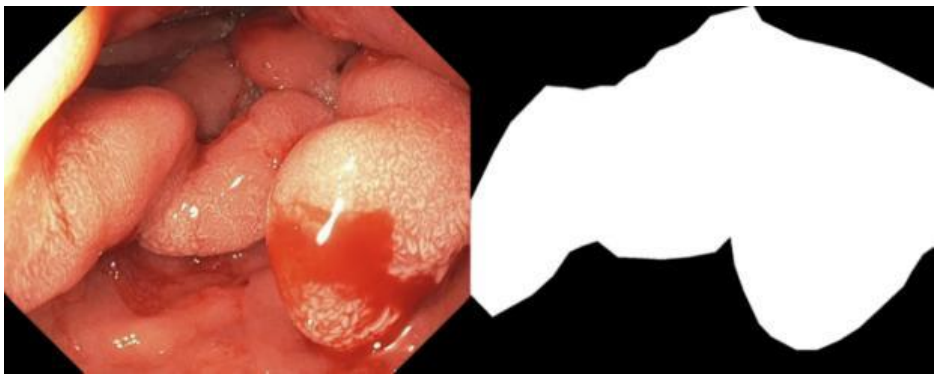


**Figure 24 Image of polyp and corresponding mask [29]**

### 4.1.2. KUMC Dataset

The KUMC Dataset [41] was constructed by the University of Kansas Medical Center and consists of 80 colonoscopy video sequences that are manually labeled with bounding boxes as well as the polyp classes for the entire dataset (Figure 25). The image resolution can vary from 384 × 288 to 768 × 576. The dataset consists of two (2) classes, "Hyperplastic" and "Adenomatous", with "Hyperplastic" = 0 and "Adenomatous" = 1.

The dataset is split up into three folders, "test2019", "train2019", and "val2019". The folder "test2019" contains a folder "Annotation" inside of which are 24 folders each consisting of HTML files, and it also contains a folder named "Image" inside of which are 24 folders consisting JPG files. The folder "train2019" contains the folders "Annotation" and "Image" inside of which are 28,773 HTML and JPG files respectively. Finally, in the "val2019" folder there are two folders, the first is "Annotation" inside of which are 17 folders each consisting of HTML files and then there is a folder named "Image" inside of which are 17 folders consisting of JPG files. The dataset is open-access.

The dataset was split up into: 80% train, 20% test [4].

**Figure 25 KUMC Video Frames [41]**

### 4.1.3. Lacmus Drone Dataset

The Lacmus Drone Dataset [54] consists of 1365 carefully annotated images that have been captured by drone (Figure 26). The dataset consists on one (1) class “pedestrian” with “pedestrian” = 0. The dataset is split into two folders, “annotations” and “images”, each consisting 1365 files in TXT and JPEG format respectively. The annotations are in VOC (Visual Object Classes) format, providing detailed information about the bounding boxes and labels of detected individuals. The altitude ranges anywhere from 50 to 100 meters with image resolutions set at 3000 × 4000 pixels and an average human-size representation of 50 × 100 pixels. It is an open-access dataset.

The dataset was split up into: 65% train, 20% validation, 15% test.

**Figure 26 Lacmus Drone dataset sample [54]**

## 4.2. Experimental Methodology

### 4.2.1. Experimental Summary

This experiment is split up in three parts: a) Firstly, the Lacmus dataset will be used to train and evaluate the YOLOv5, HIC-YOLOv5, and the improved YOLOv5 with EfficientNet

algorithms, b) a plethora of algorithms will be trained and the results will be compared using the Kvasir-SEG dataset, and finally c) YOLOv9 and the improved version of YOLOv5 with EfficientNet will be trained and tested using the KUMC dataset along with the same algorithms with the implementation of Transformer. The goal is to determine which algorithm comes first in accuracy as well as detection speed, which is why a plethora of datasets are chosen.

### 4.2.2. Experimental Parameters

Equipment: For this search process, the algorithms were tested using Google Colab Pro. The algorithm is implemented by PyTorch.

The learning rate is equal to 0.01 which is the default for Google Colab Pro.

For the Lacmus Dataset: Input image size was set to 320 × 320, batch size was set to 16, and training epoch was set to 100 (the dataset was too large for 160 epochs).

For the Kvasir-SEG dataset: Input image size was set to 320 × 320, batch size was set to 64, and training epoch was set to 160. These were the parameters set by Yoo et al. [4] which we will be comparing our results to.

For the KUMC dataset Input image size was set to 320 × 320, batch size was set to 16, and training epoch was set to 160.

The YOLO algorithm does not use earlystopping because the file best.pt will always contain the best model, regardless of when training stops.

### 4.2.3. Evaluation Criteria

The criteria used to evaluate the performance results include IoU, Precision, Recall, and mAP [39].

1) IoU: Intersection over Union. It can be calculated by taking the overlap area between the predicted region and the ground truth, and then dividing it by the combined area of the two. Its value is between 0 and 1. It can be calculated using the following function:

$$\text{IoU} = (A \cap B)/(A \cup B)$$

2) Precision: True Positives over the sum of True Positives and False Positives. It emphasizes on the correctness of positive predictions. Answers the question: What proportion of positive identifications was actually correct?

$$P_x = \frac{TP_x}{TP_x + TP_x}$$

3) Recall: True Positives over the sum of True Positives and False Negatives. It focuses on capturing all relevant instances. Answers the question: What proportion of actual positives was identified correctly?

$$R_x = \frac{TP_x}{TP_x + FN_x}$$

4) mAP: Average Precision (AP), measures Precision scores at different thresholds along the Precision Recall (PR) curve and can be calculated as a weighted mean. Mean Average Precision (mAP) is the mean of AP values for all classes.

5) Mean Average Precision at IoU 0.50 (mAP@50): It measures the accuracy of the model in detecting objects by calculating the average precision across all classes at an Intersection over Union (IoU) threshold of 0.50. A prediction is considered correct if the IoU is at least 0.50.

6) Mean Average Precision at IoU thresholds ranging from 0.50 to 0.95 (mAP@50-95): Unlike mAP@50, which considers only one IoU threshold of 0.50, mAP@50-95 averages the precision across multiple IoU thresholds, starting from 0.50 to 0.95 in steps of 0.05. This metric provides a more nuanced evaluation of the model's ability to accurately detect objects and their boundaries.

True positive is the instance where an item is correctly identified by the model as what it is, whereas a true negative is when the model can correctly predict a negative class. Similarly, a false positive is when the model falsely predicts a positive class, and false negative when a model incorrectly predicts a negative class.

## 4.3 Results

To begin, the results of Yoo et al.'s [4] publication were compared against our own. This was done to validate the accuracy of said results. Following is a graph that lists the Precision, Recall, and mAP for YOLOv5m, YOLOv5m-Tiny and YOLOv5m with Transformer which they called YOLOv5m-TST, comparing their results and ours. The comparison was performed by training on the Kvasir-SEG dataset.

| **YOLOv5m** | **P** | **R** | **mAP@.5** | **mAP@.5:.95** |
|---|---|---|---|---|
| Publication [4] | 0.9323 | 0.8906 | 0.8988 | 0.6631 |
| Ours | 0.914 | 0.851 | 0.906 | **0.574** |
| | | | | |
| **YOLOv5m-Tiny** | **P** | **R** | **mAP@.5** | **mAP@.5:.95** |
| Publication [4] | 0.9072 | 0.8889 | 0.9008 | 0.6672 |
| Ours | 0.914 | 0.855 | 0.913 | **0.581** |
| | | | | |
| **YOLOv5m-TST** | **P** | **R** | **mAP@.5** | **mAP@.5:.95** |
| Publication [4] | 0.9369 | 0.8915 | 0.9150 | 0.6855 |
| Ours | 0.926 | 0.809 | 0.902 | **0.58** |

**Table 8 Comparison of Yoo et al.'s [4] results and ours**

As it can be seen, the results presented by Yoo et al. were had little difference from ours when trained on the same dataset with the same parameters, a fact that validates the results.

### 4.3.1. Testing on Lacmus Dataset

In this section the Lacmus dataset [54] will be used to train and test the performance of:

- YOLOv5
- HIC-YOLOv5
- YOLOv5+EffNet+FPN+SE Blocks

The goal is to determine which algorithm performs best when it come to drone images where the target objects appear small in an image. When Tang et al. [27] created the state-of-the-art version of YOLOv5, HIC-YOLOv5, they added an extra prediction head for small object detection (SODH), by adding an involution block between the backbone and neck (CFFI), and by adding an attention mechanism at the end of the backbone to decrease computational cost (CBAM). This was done to tackle the low accuracy that occurs when trying to detect small objects using YOLOv5. The dataset was split by 65% train, 20% validation, and 15% test. The batch size was set to 16 and the network was trained for a total of 100 epochs.

| Algorithm | P | R | mAP@.5 | mAP@.5:.95 | AUC | GPU |
|---|---|---|---|---|---|---|
| **YOLOv5** | 0.866 | 0.74 | 0.796 | 0.437 | 0.64 | 3.6 GB |
| **HIC-YOLOv5** | 0.8823 | 0.7 | 0.7441 | 0.4451 | 0.6 | 5.7 GB |
| **YOLOv5+EffNet+FPN+ SE Blocks** | 0.87 | 0.776 | 0.802 | 0.487 | 0.66 | 3.6 GB |

**Table 9 Metric comparison when trained on Lacmus dataset**

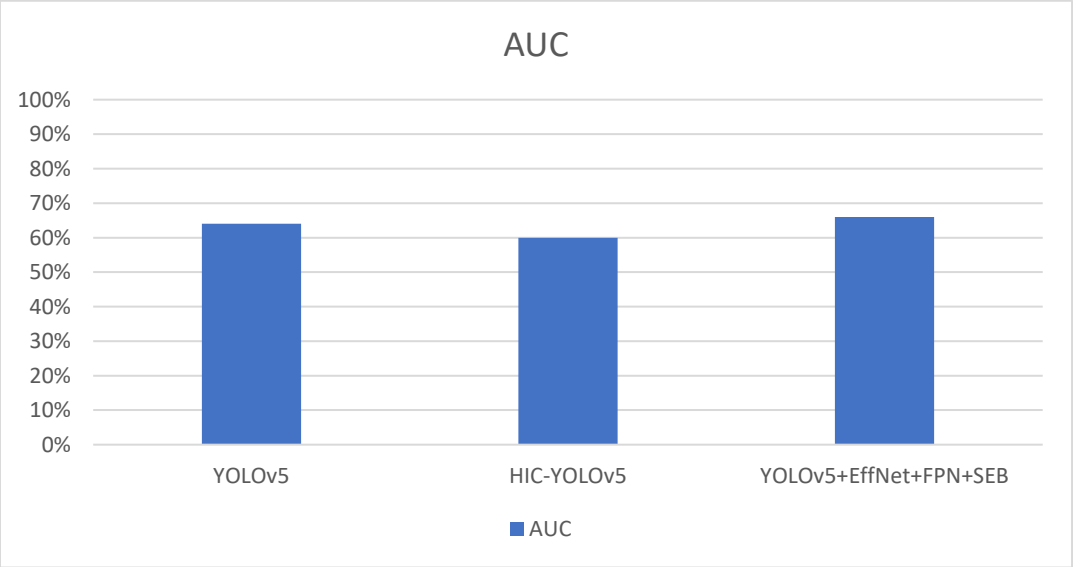



**Figure 27 Visual representation of AUC comparison**

- ROC Graphs:

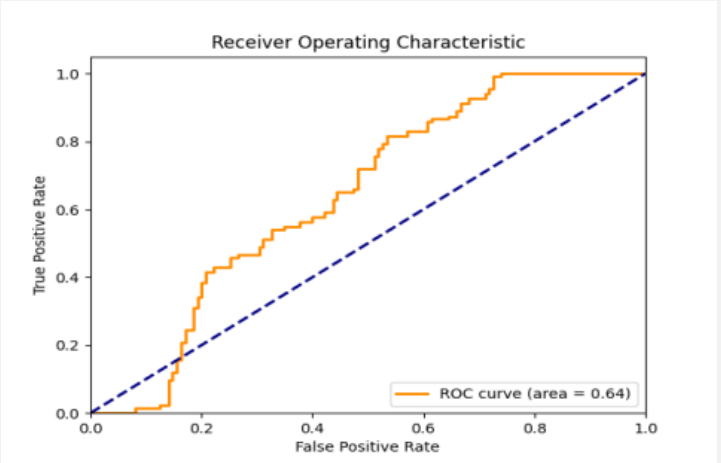
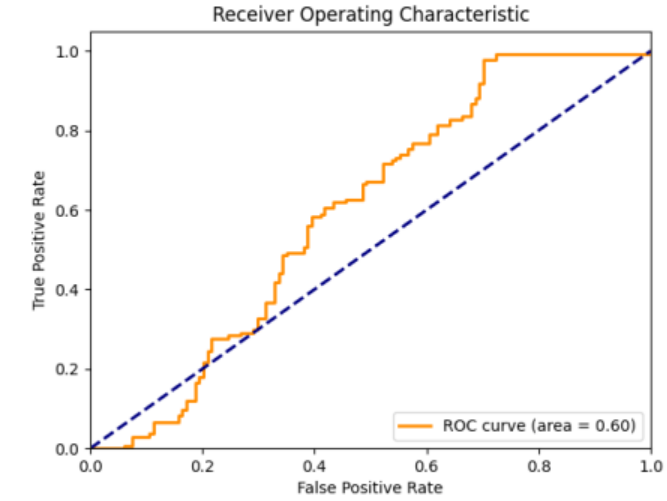
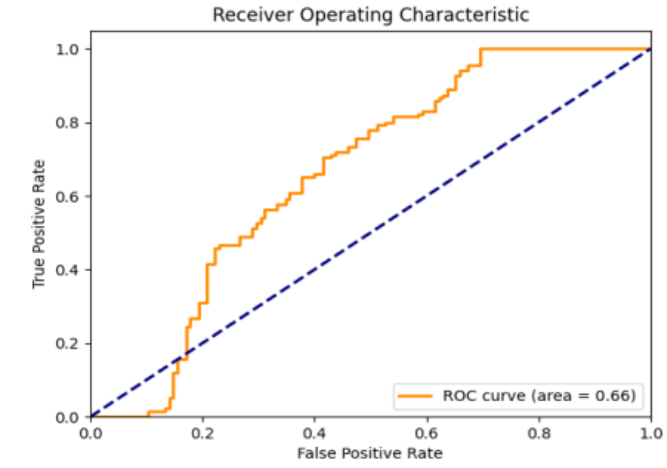
| |
|---|
| **YOLOv5** |
|  |
| **HIC-YOLOv5** |
|  |
| **YOLOv5+EffNet+FPN+SE Blocks** |
|  |

**Table 10 ROC graph for each algorithm**

- Training Time Stamps:

| Algorithm | 1 Epoch | 100 Epochs |
|---|---|---|
| **YOLOv5** | 243 sec approx. | 25,206 sec |
| **HIC-YOLOv5** | 250 sec approx. | 28,386 sec |
| **YOLOv5+EffNet+FPN+SE Blocks** | 255 sec approx. | 28,499 sec |

**Table 11 Training time stamps**

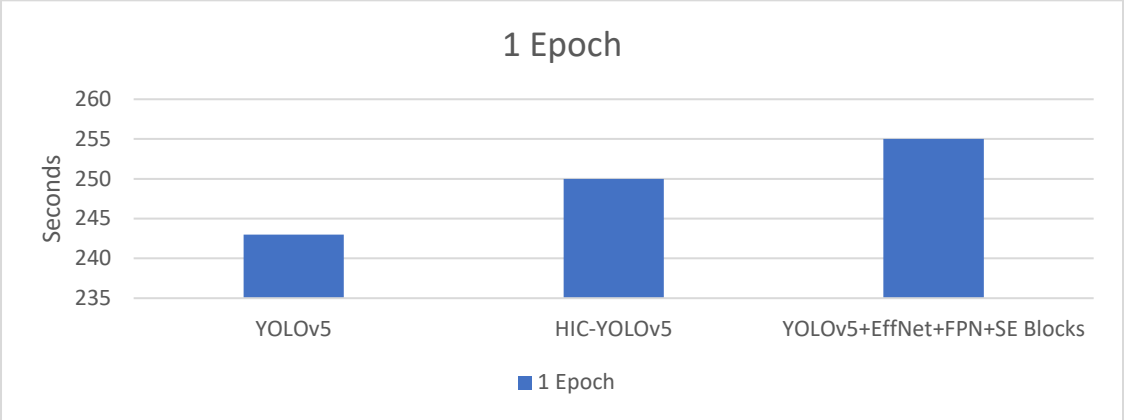



**Figure 28 Visual representation of time training time stamps for 1 epoch**

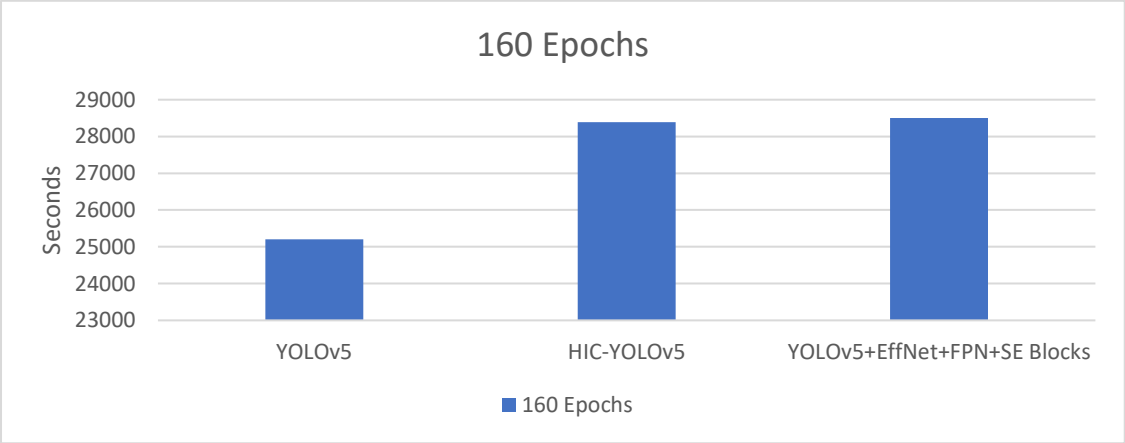



**Figure 29 Visual representation of time training time stamps for 160 epochs**

- Detection Time Stamps:

| Algorithm | Pre-Processing | Inference | NMS per image | Total Time |
|---|---|---|---|---|
| **YOLOv5** | 0.7 ms | 10.0 ms | 9.9 ms | 20.6 ms |
| **HIC-YOLOv5** | 0.9 ms | 17.5 ms | 11.4 ms | 29.8 ms |
| **YOLOv5+EffNet+FPN+SE Blocks** | 0.6 ms | 10.1 ms | 11.7 ms | 22.4 ms |

**Table 12 Detection time stamps**

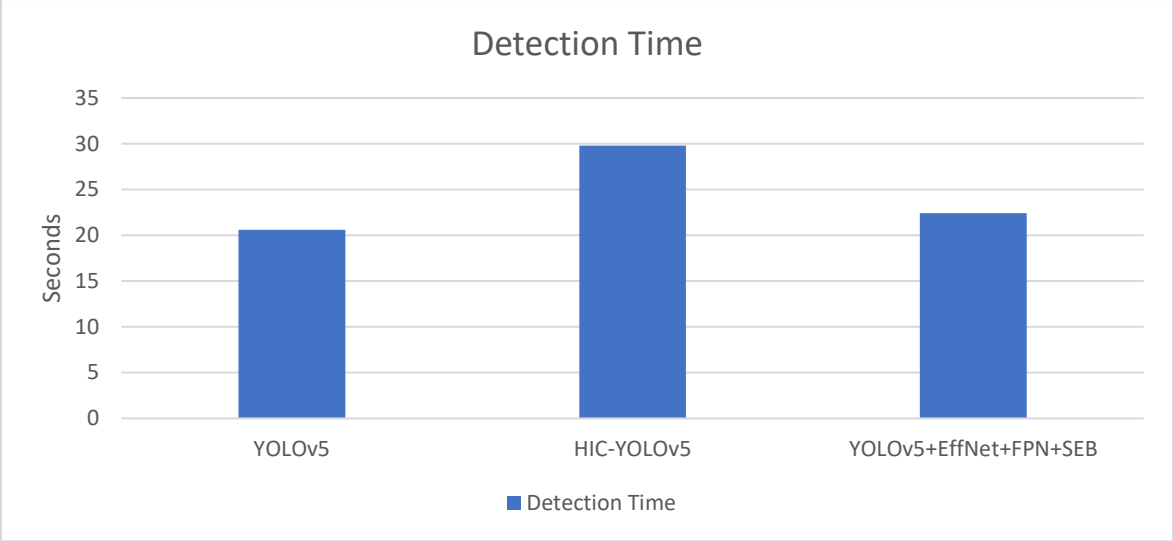



**Figure 30 Visual representation of time detection time stamps**

- Detected Images:

| YOLOv5 | HIC-YOLOv5 | YOLOv5+EfficientNet+FPN+ SE Blocks |
|---|---|---|
| **Correct Detection** | **False and Missed Detection** | **Correct Detection** |
|  |  |  |
| **False Detection** | **Correct Detection** | **False Detection** |
|  |  |  |
| **Correct Detection** | **Correct Detection** | **Correct Detection** |
|  |  |  |

| Correct Detection | False Detections | False Detection |
|---|---|---|
|  |  |  |
| **Correct Detection** | **Partial Detection** | **Correct Detection** |
|  |  |  |

**Table 13 Detection images samples**

The Lacmus dataset was selected specially to evaluate HIC-YOLOv5's performance. This is due to the fact that the algorithm is purportedly made to identify small objects in images, including humans in drone photos such as the ones in this dataset.

In order to evaluate the algorithms' performance using Lacmus, HIC-YOLOv5 was put to the test against both the original and enhanced versions of YOLOv5, YOLOv5+EfficientNet+FPN+SE Blocks. With an AUC value of 0.66 and a training duration of roughly 255 seconds per epoch, YOLOv5+EffNet+FPN+SE Blocks is the method with the best performance. YOLOv5 comes in second, and HIC-YOLOv5 places last. The fastest detection time is held by the original YOLOv5 algorithm with a total time of 20.6 ms.

The training time per epoch takes much longer compared to the other datasets due to the large size of the images that are included in the Lacmus dataset.

### 4.3.2. Testing on Kvasir-SEG

In this section we will be training a plethora of YOLO algorithms, using the Kvasir-SEG dataset for polyp detection. The final goal is to determine which algorithm performs best when it comes to detection speed and accuracy. The algorithms used are the following:

1. YOLOv5s
2. HIC-YOLO5 [39]
3. YOLOv5 modified (with EfficientNet + FPN + SE Blocks)
4. YOLOv5 modified (with EfficientNet + FPN + SE Blocks) + Transformer
5. YOLOv9
6. YOLOv9 + Transformer

The dataset was split as: 65% train, 20% validation, and 15% test. The training epochs were set to 160, with a batch size of 64.

The YOLOv5s algorithm was altered with the final goal to improve the detection time and accuracy when it comes to polyp detection. To do so the backbone was changed from CSPBarknet53 to EfficientNet. Also, a Feature Pyramid Network (FPN) and Squeeze-and-Excitation (SE) Blocks were added. Results such as time stamps, Precision, Recall, and mAP were collected for 160 epochs.

| Algorithm | P | R | mAP@.5 | mAP@.5:.95 | AUC | GPU |
|---|---|---|---|---|---|---|
| **YOLOv5** | 0.898 | 0.863 | 0.908 | 0.564 | 0.85 | 4.7 GB |
| **HIC-YOLOv5** | 0.8899 | 0.72 | 0.8571 | 0.497 | 0.67 | 7.3 GB |
| **YOLOv5 EfficientNet + FPN + SE Blocks** | 0.86 | 0.82 | 0.894 | 0.57 | 0.89 | 4.7 GB |
| **YOLOv5 EfficientNet + FPN + SE Blocks + Transformer** | 0.932 | 0.844 | 0.909 | 0.572 | 0.89 | 4.7 GB |
| **YOLOv9** | 0.749 | 0.644 | 0.698 | 0.358 | 0.67 | 10.3 GB |
| **YOLOv9 + Transformer** | 0.797 | 0.627 | 0.725 | 0.382 | 0.64 | 14 GB |

**Table 14 Metric comparison when trained on Kvasir-SEG dataset**

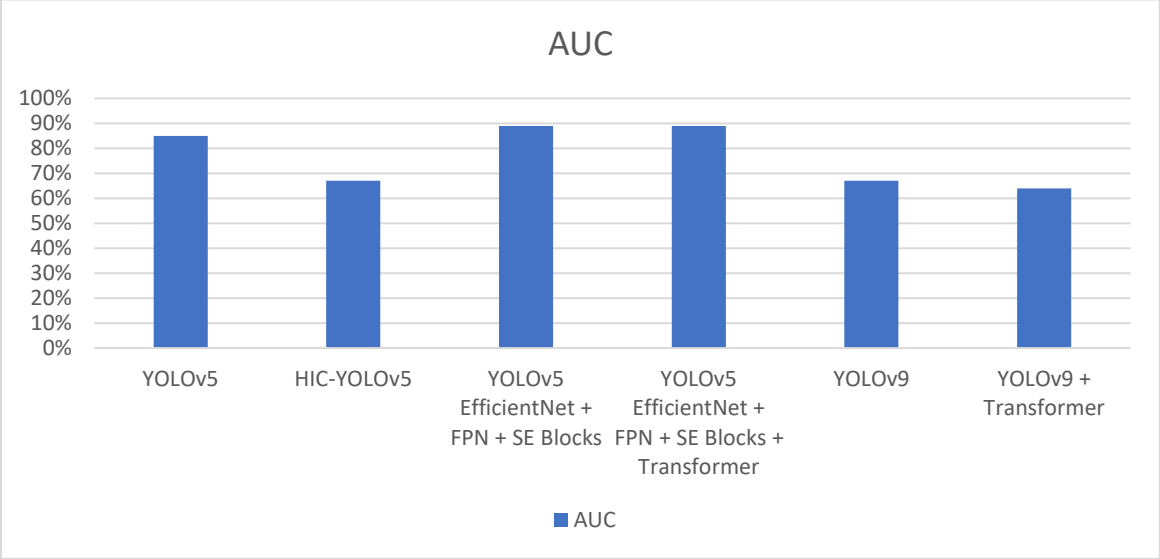
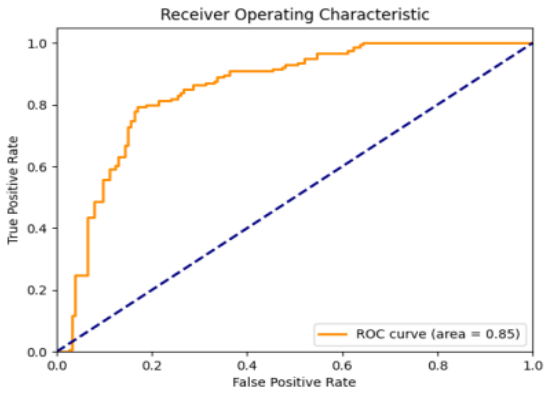
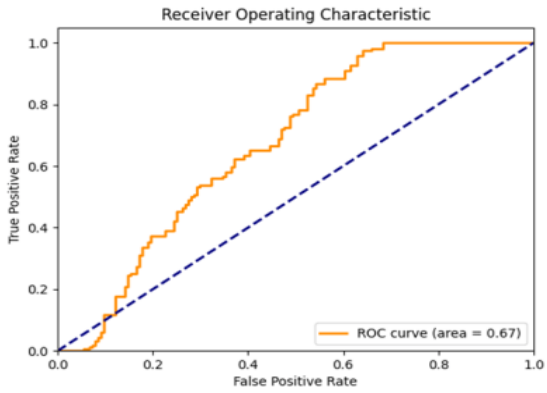
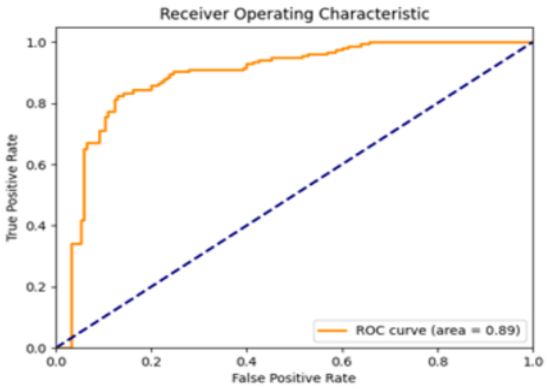
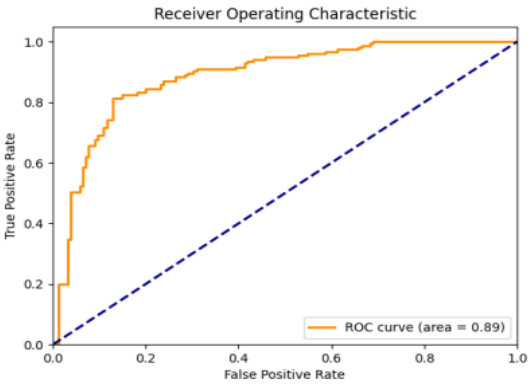



**Figure 31 Visual representation of AUC comparison**

- ROC Graphs:

| YOLOv5 | HIC-YOLOv5 |
|---|---|
|  |  |
| **YOLOv5 + EfficientNet + FPN + SE Blocks** | **YOLOv5 + EfficientNet + FPN + SE Blocks + Transformer** |
|  |  |

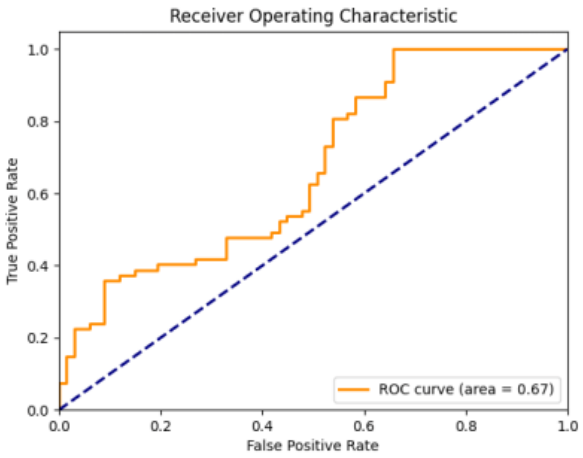
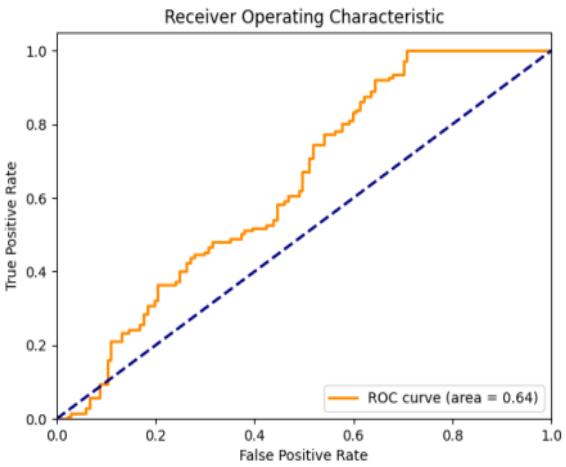
| YOLOv9 | YOLOv9 + Transformer |
|---|---|
|  |  |



**Table 15 ROC graph for each algorithm**

- Time Stamps:

| Algorithm | 1 Epoch | 160 Epochs |
|---|---|---|
| **YOLOv5** | 35 sec approx. | 3,980 sec |
| **HIC-YOLOv5** | 37 sec approx. | 5,670 sec |
| **YOLOv5 EfficientNet + FPN + SE Blocks** | 30 sec approx. | 3,864 sec |
| **YOLOv5 EfficientNet + FPN + SE Blocks + Transformer** | 32 sec approx. | 5,132 sec |
| **YOLOv9** | 20 sec approx. | 3,662 sec sec |
| **YOLOv9 + Transformer** | 27 sec approx. | 5,814 sec |

**Table 16 Training time stamps**

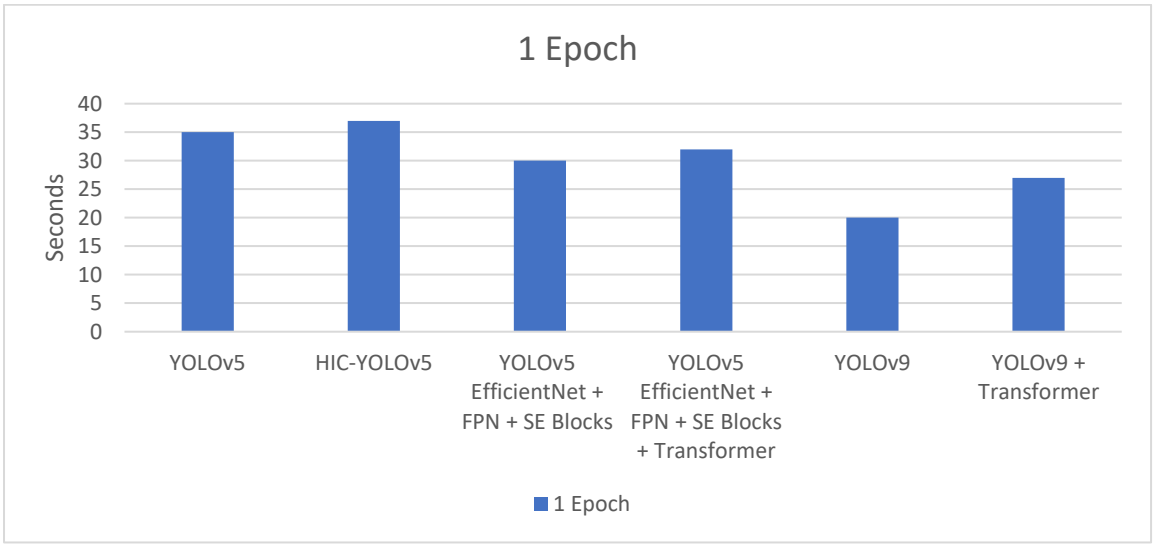



**Figure 32 Visual representation of training time stamps for 1 epoch**

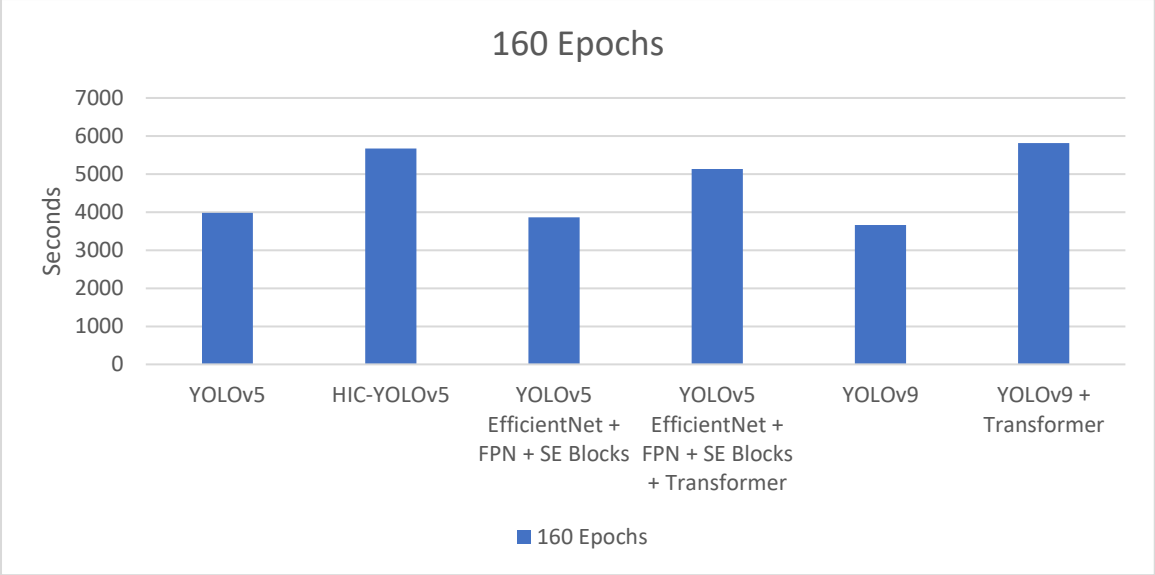



**Figure 33 Visual representation of training time stamps for 160 epochs**

- Detection Time Stamps:

| Algorithm | Pre-Processing | Inference | NMS per image | Total Time |
|---|---|---|---|---|
| **YOLOv5** | 0.5 ms | 11.8 ms | 5.1 ms | 17.4 ms |
| **HIC-YOLOv5** | 0.5 ms | 17.1 ms | 4.4 ms | 22.0 ms |
| **YOLOv5 EfficientNet + FPN + SE Blocks** | 0.5 ms | 13.0 ms | 4.8 ms | 18.3 ms |
| **YOLOv5 EfficientNet + FPN + SE Blocks + Transformer** | 0.5 ms | 13.7 ms | 7.4 ms | 21.6 ms |
| **YOLOv9** | 0.5 ms | 84.9 ms | 4.9 ms | 90.3 ms |
| **YOLOv9 + Transformer** | 0.5 ms | 80.0 ms | 4.5 ms | 85.0 ms |

**Table 17 Detection time stamps**

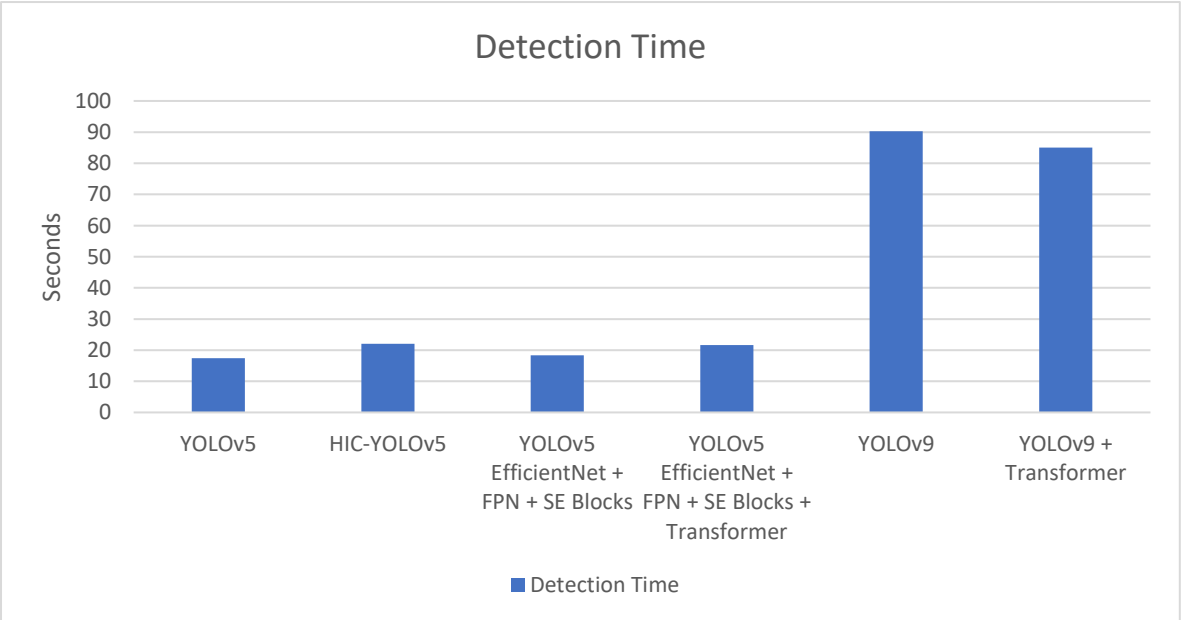



**Figure 34 Visual representation of detection time stamps**

- Detected Images:

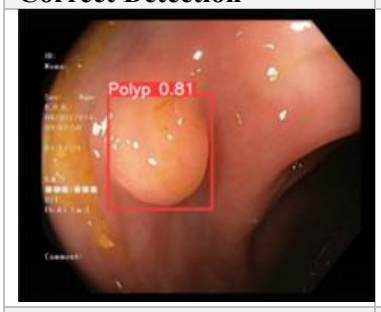
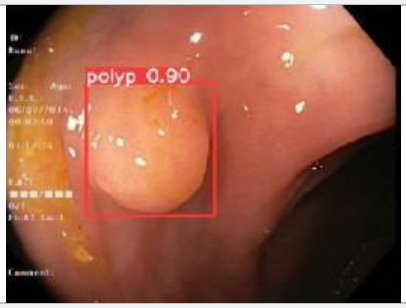
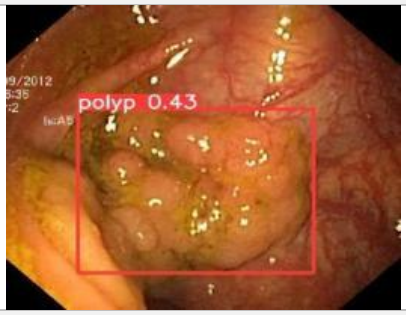
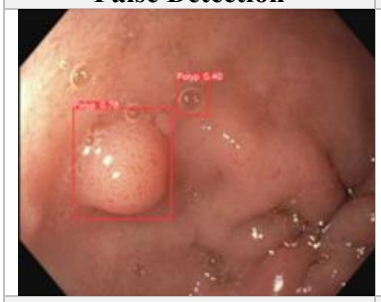
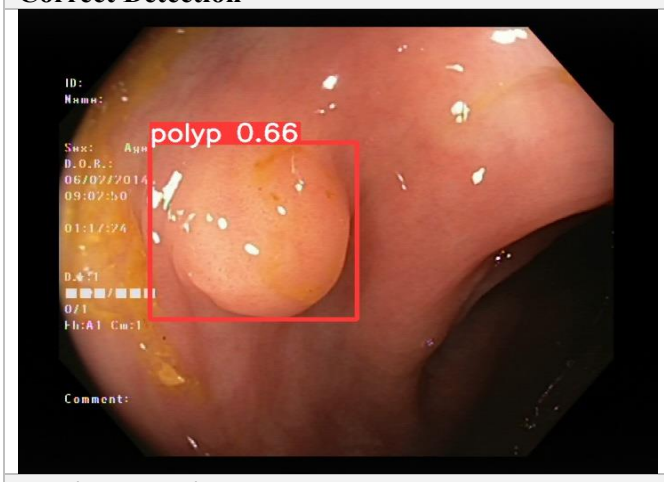
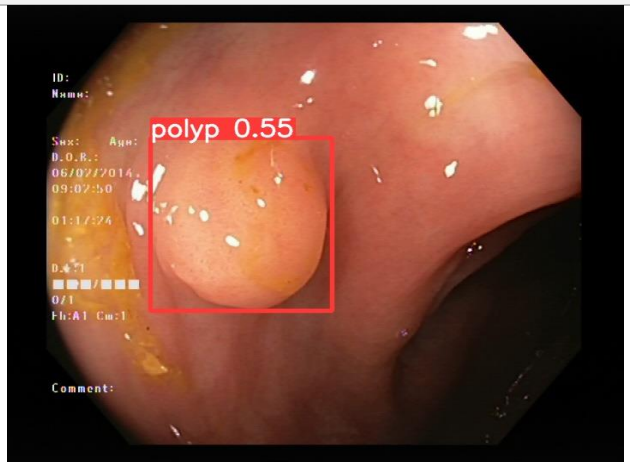
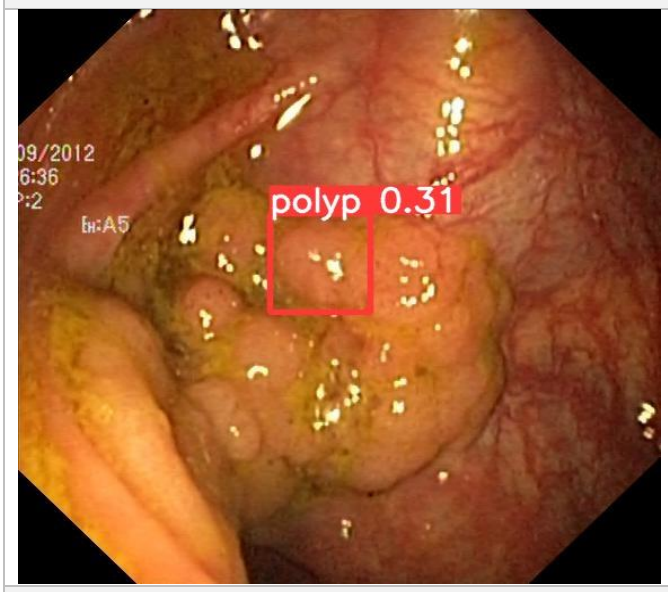
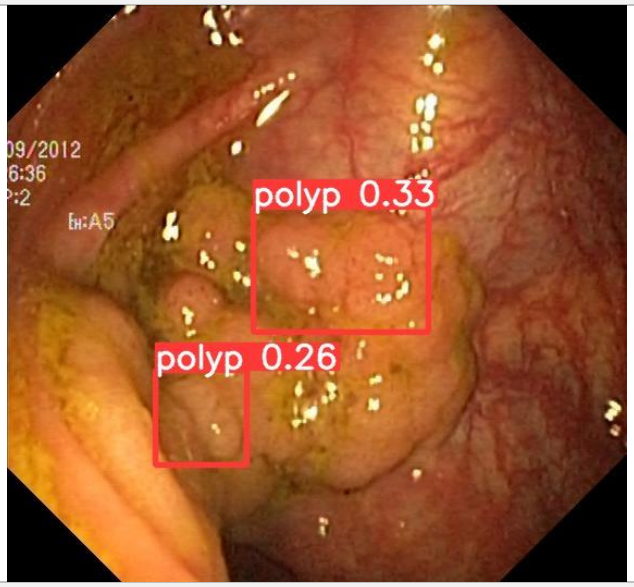
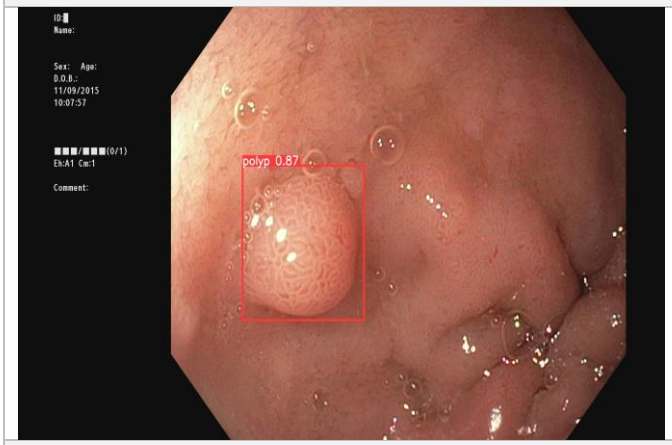
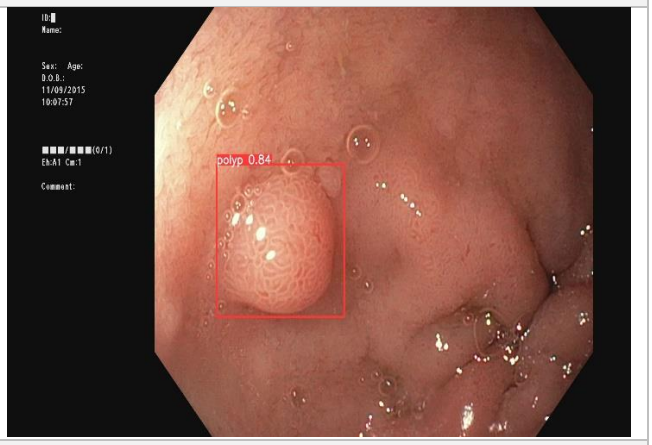
| YOLOv5 | HIC-YOLOv5 | YOLOv5+EfficientNet+FPN+SE Blocks |
|---|---|---|
| **Correct Detection** | **Correct Detection** | **Correct Detection** |
|   |  |   |
| **Missed Detection** | **Correct Detection** | **Correct Detection** |
|  |  |   |
| **False Detection** | **Correct Detection** | **Correct Detection** |
|  |  |  |
| **Missed Cluster** | **Correct Detection** | **Missed Cluster** |
|   |  |   |

**Table 19 Detection image samples**

| YOLOv9 | YOLOv9 + Transformer |
|---|---|
| **Correct Detection** | **Correct Detection** |
|   |   |
| **Partial Detection** | **Partial Detection** |
|   |   |
| **Correct Detection** | **Correct Detection** |
|   |   |
| **Partial Detection** | **Partial Detection** |

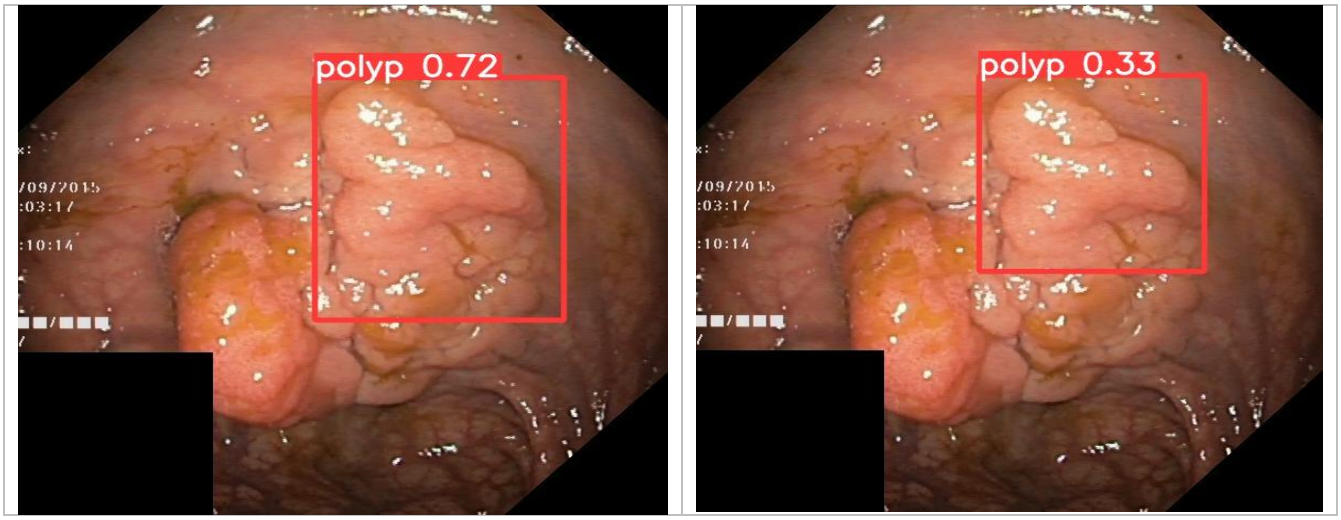



**Table 20 Detection image samples**

As it can be seen, when the algorithms are tested using the Kvasir-SEG dataset, YOLOv5+EfficientNet+FPN+SE Blocks places first. More specifically, this algorithm can detect the polyps with the highest accuracy percentage as it can be seen in the detection images, presents the highest AUC with a value of 0.89 (tied with YOLOv5+EfficientNet+FPN+ SE Blocks+Transformer) and the training process took only 30 sec/epoch approximately. It is followed by YOLOv5+EfficientNet+FPN+SE Blocks+Transformer which holds an AUC value of 0.89 and a training time of 32 sec/epoch approximately. The fastest detection time is held by the original YOLOv5 algorithm with a total time of 17.4 ms.

After examining the performed detections, it can be noticed that the HIC-YOLOv5 algorithm presents fewer false detections as well as fewer missed detections compared to the original YOLOv5. Also, when objects appear in clusters, HIC-YOLOv5 is able to detect each object separately whereas YOLOv5 identifies them as one cluster. Finally, HIC-YOLOv5 is known to have better overall detection of small objects.

### 4.3.3. Testing on KUMC Dataset

In this section we will be training and examining the results of a few algorithms when using the KUMC dataset [41]. Inspired by the work of YOO et al. [4], a Transformer will be added to the following algorithms:

- YOLOv5 + EfficientNet + FPN + SE Blocks
- YOLOv9

The algorithms with Transformer will be tested against the original versions using the KUMC dataset to see if any significant improvement has been made. The dataset was split into 80% train and 20% test with a batch of 16 and 160 epochs.

| YOLOv5+EffNet+FPN+SE Blocks | P | R | mAP@.5 | mAP@.5:.95 | AUC | GPU |
|---|---|---|---|---|---|---|
| Original | 0.74 | 0.539 | 0.627 | 0.418 | 0.72 | 4.3 GB |
| Transformer | 0.801 | 0.637 | 0.747 | 0.492 | 0.77 | 4.3 GB |
| | | | | | | |
| **YOLOv9** | **P** | **R** | **mAP@.5** | **mAP@.5:.95** | **AUC** | **GPU** |
| Original | 0.605 | 0.388 | 0.426 | 0.263 | 0.62 | 10.4 GB |
| Transformer | 0.628 | 0.554 | 0.663 | 0.376 | 0.69 | 13.6 GB |

**Table 21 Metric comparison when trained on KUMC dataset**

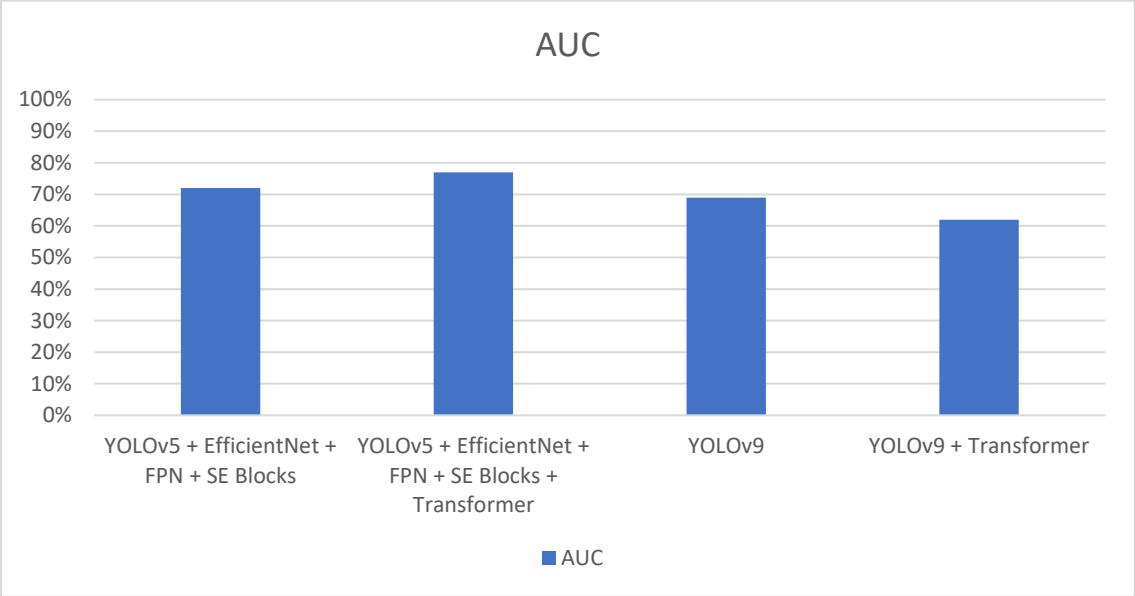
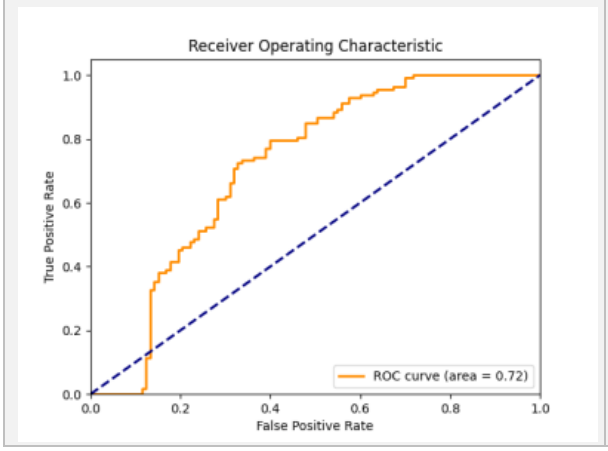
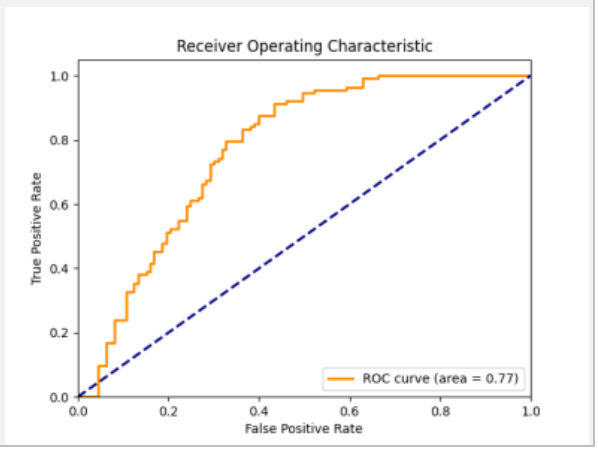



**Figure 35 Visual representation of AUC comparison**

- ROC Graphs:

| YOLOv5 + EfficientNet + FPN + SE Blocks | YOLOv5 + EfficientNet + FPN + SE Blocks + Transformer |
|---|---|
|  |  |

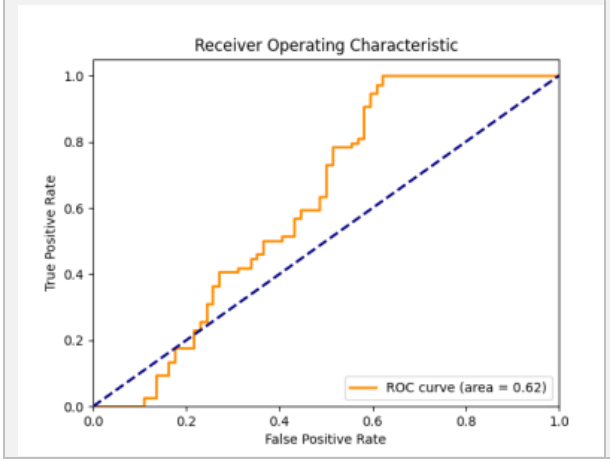
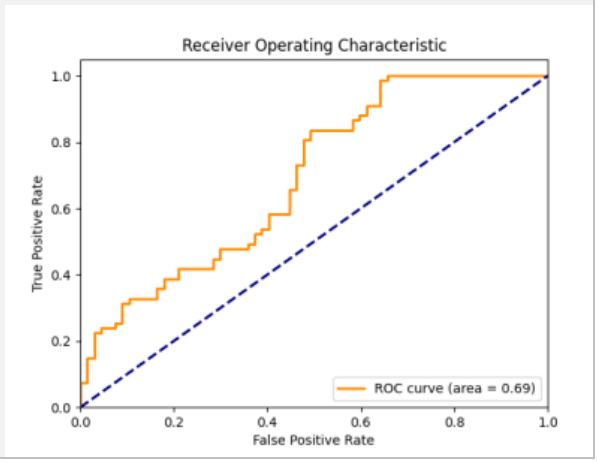
| YOLOv9 | YOLOv9 + Transformer |
|---|---|
|  |  |



**Table 22 ROC graph for each algorithm**

- Time Stamps:

| Algorithm | 1 Epoch | 160 Epochs |
|---|---|---|
| **YOLOv5 + EfficientNet + FPN + SE Blocks** | 25 sec approx. | 6,303 sec |
| **YOLOv5 + EfficientNet + FPN + SE Blocks + Transformer** | 45 sec approx. | 6,989 sec |
| **YOLOv9** | 22 sec approx. | 4,262 sec |
| **YOLOv9 + Transformer** | 35 sec approx. | 6,898 sec |

**Table 23 Training time stamps**

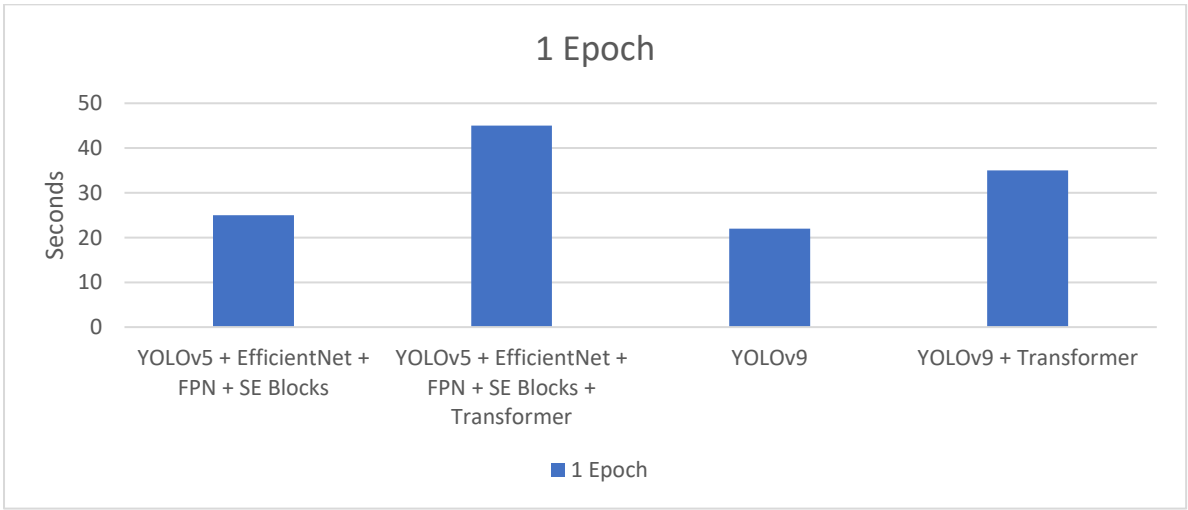



**Figure 36 Visual representation of training time stamps for 1 epoch**

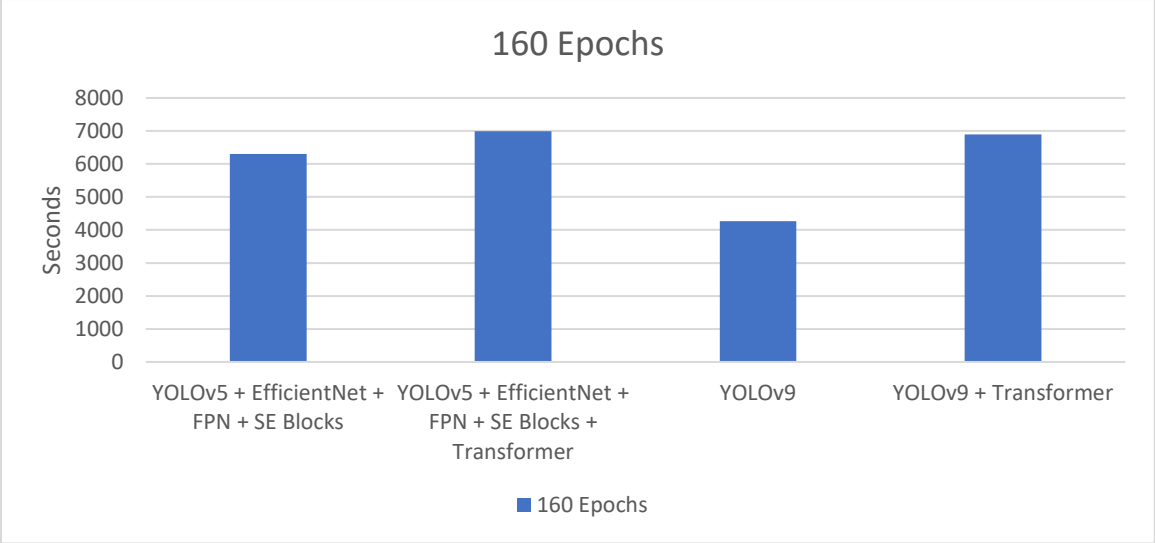



**Figure 37 Visual representation of training time stamps for 160 epoch**

- Detection Time Stamps:

| Algorithm | Pre-Processing | Inference | NMS per image | Total Time |
|---|---|---|---|---|
| **YOLOv5 + EfficientNet + FPN + SE Blocks** | 0.5 ms | 9.7 ms | 5.6 ms | 15.8 ms |
| **YOLOv5 + EfficientNet + FPN + SE Blocks + Transformer** | 0.4 ms | 10.0 ms | 4.0 ms | 14.4 ms |
| **YOLOv9** | 0.4 ms | 24.8 ms | 5.5 ms | 30.7 ms |
| **YOLOv9 + Transformer** | 0.4 ms | 77.2 ms | 4.1 ms | 81.7 ms |

**Table 24 Detection time stamps**

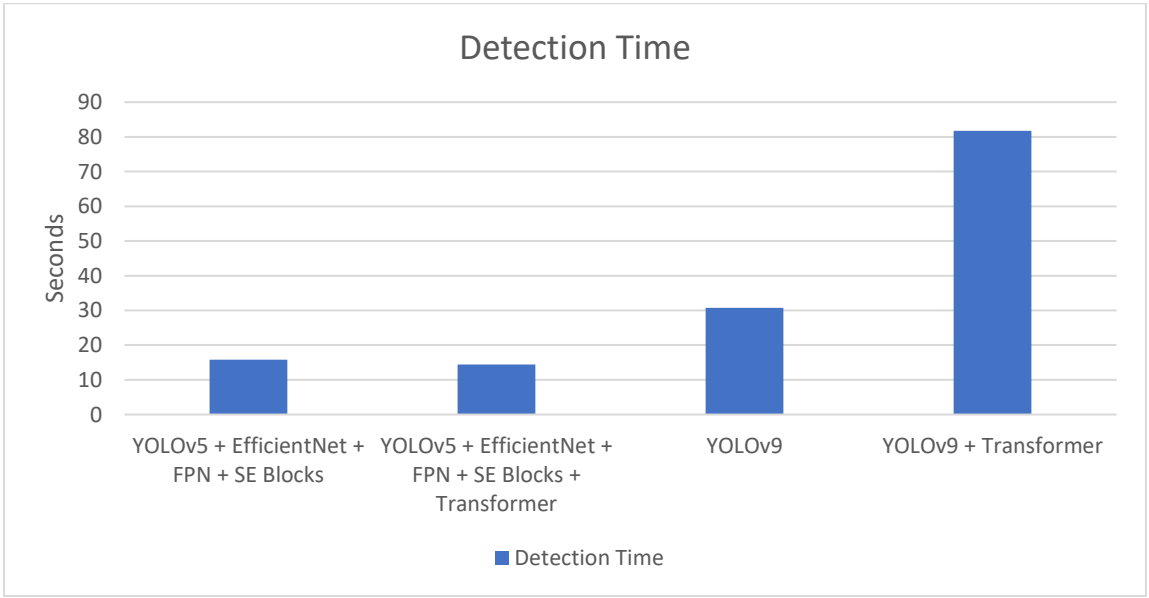



**Figure 38 Visual representation of detection time stamps**

- Detected Images:

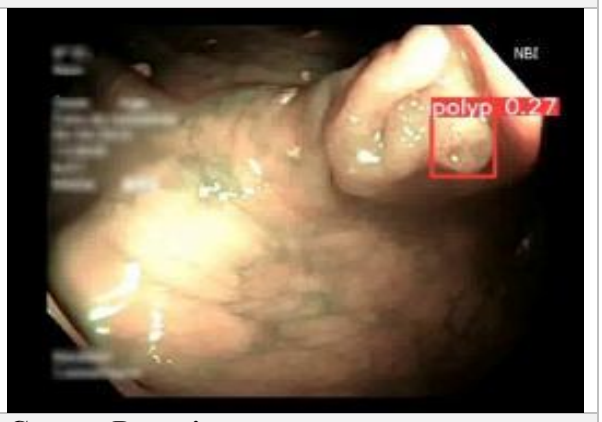
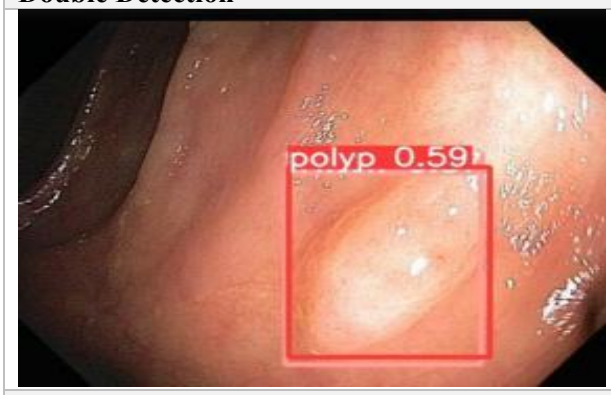
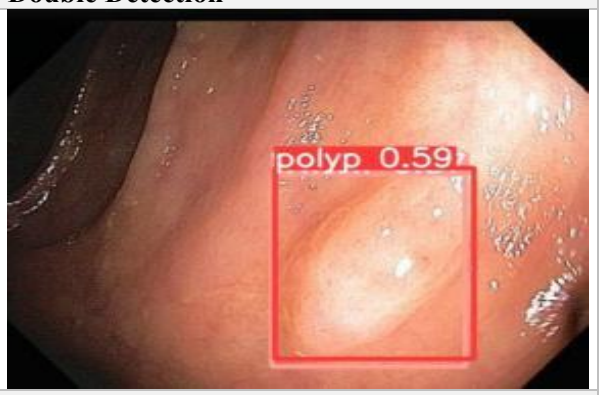
| YOLOv5 + EfficientNet + FPN + SE Blocks | YOLOv5 + EfficientNet + FPN + SE Blocks + Transformer |
|---|---|
| **Missed Detection** | **Correct Detection** |
|  |  |
| **Correct Detection** | **Correct Detection** |
|  |  |
| **Double Detection** | **Double Detection** |
|  |  |
| **Missed Detection** | **Missed Detection** |
|  |  |

**Table 25 Detection image samples**

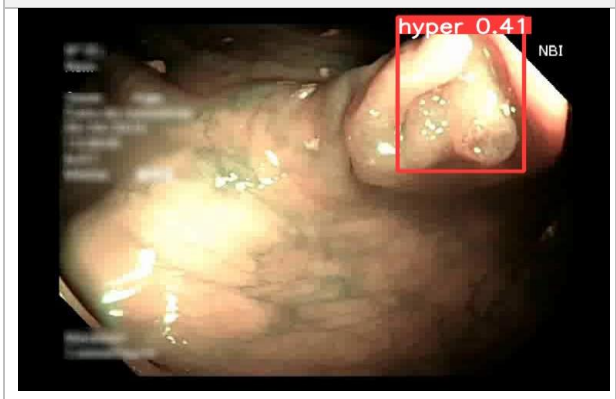
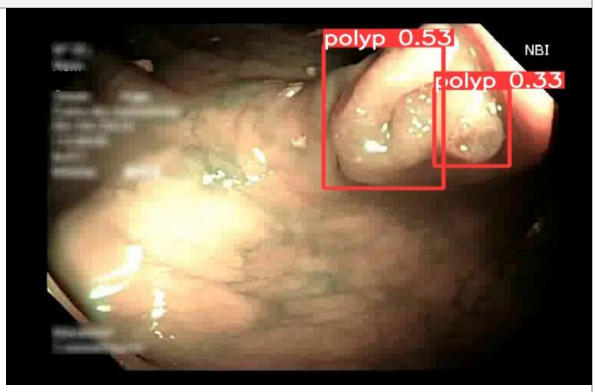
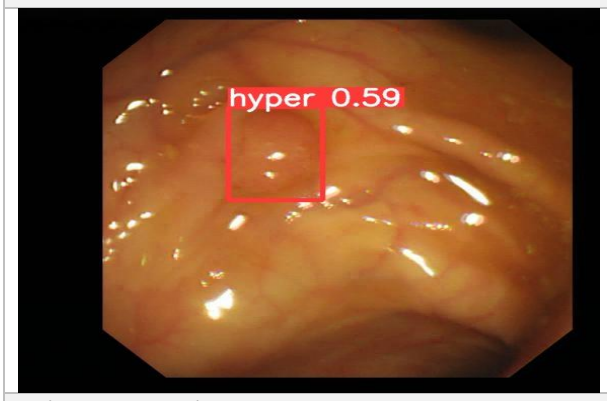
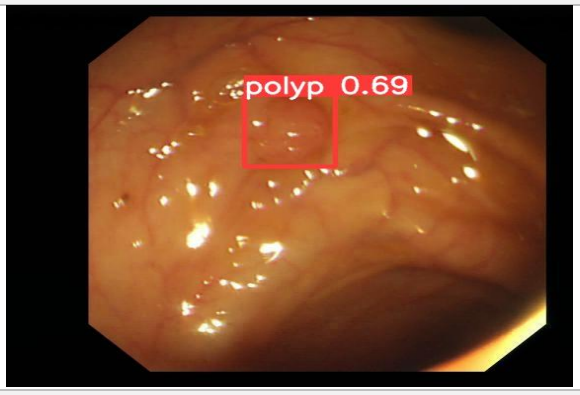
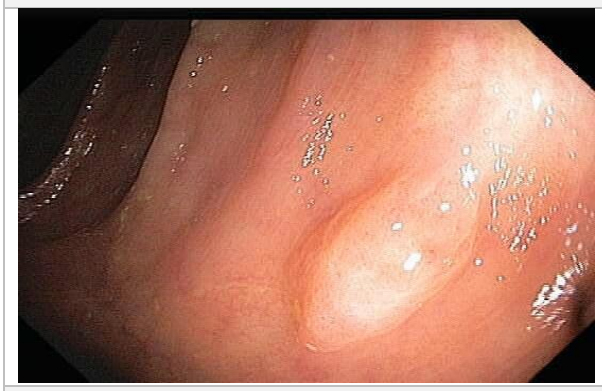
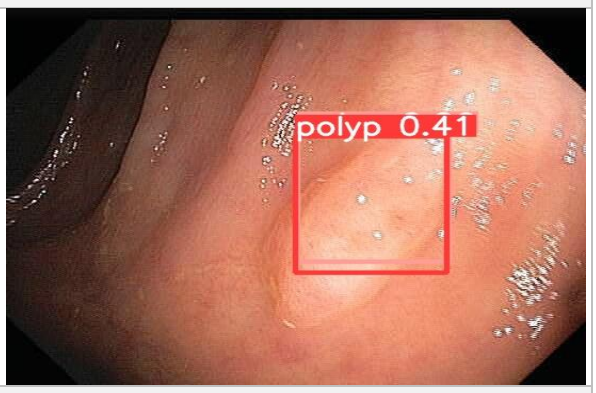
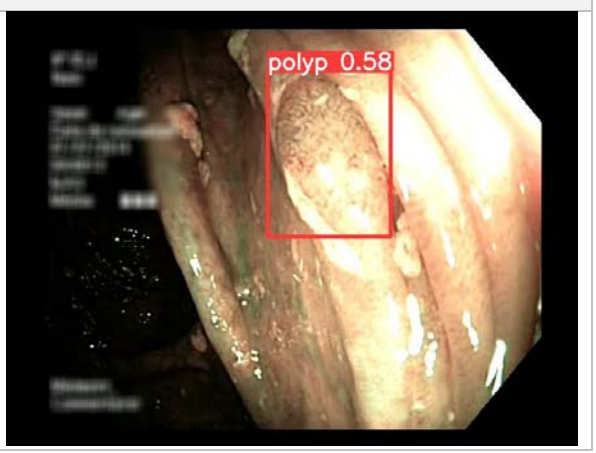
| YOLOv9 | YOLOv9 + Transformer |
|---|---|
| **Correct Detection** | **Correct Detection** |
|   |   |
| **Correct Detection** | **Correct Detection** |
|   |   |
| **Missed Detection** | **Correct Detection** |
|  |   |
| **Missed Detection** | **Correct Detection** |
|  |   |

**Table 26 Detection image samples**

It can be seen that for the KUMC dataset the best results appear to be held by the modified YOLOv5 algorithm with Transformer, with P equal to 0.801, R to 0.637, mAP@.5 to 0.747, and mAP@.5:.95 to 0.492 and an AUC value of 0.77. In addition to this when examining the detection images, it can be seen that it makes more correct detections than the other algorithms. Finally, even though YOLOv9 has a fast train time, its AUC value is 0.62 and its detection performance is severely lacking. The fastest detection time is held by the YOLOv5 + EfficientNet + FPN + SE Blocks + Transformer algorithm with a total time of 14.4 ms.

# CHAPTER 5

## 5.1. Conclusion

In conclusion, this study presents a comprehensive evaluation and comparison of several advanced object detection methodologies, focusing on their application in many different scenarios. The algorithms evaluated include the conventional YOLOv5, the enhanced HIC-YOLOv5, and the YOLOv5s-TST, each offering unique improvements in terms of detection speed and accuracy. The research introduces a novel enhancement method designed to further increase the efficiency and precision of these algorithms.

The study utilized extensive datasets, including colonoscopy images that feature a variety of polyp sizes, types, and locations, as well as images captured by drones during flight. This diversity in data sources ensured that the algorithms were rigorously tested and refined to optimize their detection capabilities across different conditions and applications.

Key findings from this research indicated that specialized modifications to the YOLOv5 algorithm could lead to better detection accuracies and faster processing times, which are critical for the effective real-time detection of polyps. The integration of new approaches, such as Transformer architectures and enhanced EfficientNet backbones, not only pushed the boundaries of detection performance but also provided insights into the scalability and adaptability of these models in medical imaging environments.

In evaluating the performance of object detection algorithms on different datasets, notable differences in outcomes highlight each algorithm's strengths and limitations. For the Lacmus dataset, specifically curated to assess the detection of small objects from aerial imagery, the modified YOLOv5 algorithm enhanced with EfficientNet, FPN, and SE blocks showcased the highest AUC value of 0.66, outperforming the original and HIC-enhanced versions of YOLOv5, despite the longer training duration of approximately 255 seconds per epoch due to the dataset's large image sizes. This version also demonstrated the fastest detection speeds among tested models.

In contrast, when utilizing the Kvasir-SEG dataset, which focuses on the medical detection of polyps, both YOLOv5 enhanced with EfficientNet, FPN, SE blocks, and the version further enhanced with a Transformer, tied with an impressive AUC of 0.89. However, it is noteworthy that HIC-YOLOv5, while not leading in AUC, offered fewer false positives and superior detection of objects in close proximity—a significant advantage in clustered object environments.

Lastly, the KUMC dataset revealed that YOLOv5 enhanced with EfficientNet, FPN, SE Blocks and a Transformer achieved the best balance between precision and recall among the algorithms, showing its suitability for medical applications where both sensitivity and specificity are critical. Despite YOLOv9’s rapid training time, its lower AUC of 0.62 and poorer detection capabilities underscore the challenge of balancing speed with detection performance.

These findings underscore the trade-offs between accuracy, speed, and computational efficiency across different algorithms and datasets. While some algorithms excel in speed,

others offer greater precision, indicating the necessity of selecting the appropriate algorithm based on specific use-case requirements and dataset characteristics.

Additionally, the thesis provided valuable comparative analyses, highlighting the relative advantages and disadvantages of each model. This comparative aspect extends beyond mere algorithmic efficiency, exploring the practical implications of deploying these models in real clinical settings, offering a clear path for future research to enhance early detection systems for colorectal cancer.

The potential impact of these advancements is profound, offering possibilities for significantly improved diagnostic tools that are faster, more accurate, and potentially accessible to a wider range of healthcare facilities. Future work could explore the integration of these algorithms with other diagnostic technologies, enhancing their applicability and effectiveness in a clinical setting.

Overall, this work contributed a comparative study that highlighted the pros and cons of various object detection algorithms based on the YOLO concept. The research towards this direction can contribute to the critical field of medical diagnostics, potentially improving outcomes through better early detection methodologies.

## 5.2. Future Work

Looking ahead, there are several directions for future research based on the findings of this work:

- Conducting extensive clinical trials to validate the performance of these algorithms in real-world clinical settings is essential. This will involve collaboration with medical professionals and institutions to test the algorithms on live data during procedures like colonoscopies.

- Another area for future exploration is the combination of these algorithms with other medical imaging technologies. For instance, integrating these advanced detection models with imaging software used in other types of cancer detection could potentially increase the accuracy and speed of diagnoses across different medical fields.

- Additionally, there is room for further customization and improvement of the algorithms. For example, finding ways to make the algorithms faster without sacrificing accuracy could make them more useful for other applications that require real-time data processing, such as in emergency medical services or surgical procedures.

- Increasing the size and diversity of training datasets, including more varied types of polyps and different anatomical regions, can help improve the robustness and generalizability of the detection algorithms. This includes creating synthetic datasets using data augmentation techniques and GANs (Generative Adversarial Networks).

- Exploring semi-supervised and unsupervised learning techniques can help in scenarios where labeled data is scarce. These techniques can leverage large amounts of unlabeled data to improve model performance.

- Extending the application of these detection algorithms to other medical imaging modalities, such as MRI, CT scans, and ultrasound, can broaden their impact. This includes detecting various types of abnormalities beyond polyps, such as tumors, cysts, and other pathological features.

- Finally, as technology evolves, these models could be adapted for use on new and emerging hardware platforms, potentially making them more accessible and affordable for healthcare providers worldwide. This could lead to broader adoption and ultimately better health outcomes in various clinical settings.

## REFERENCES


[1] Redmon, J., Divvala, S.K., Girshick, R.B., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. ***2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, 779-788.

[2] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. ***2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, 6517-6525.

[3] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going deeper with convolutions. ***2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)***, 1-9.

[4] Yoo, Y., Lee, J. Y., Lee, D., Jeon, J., & Kim, J. (2024). Real-Time Polyp Detection in Colonoscopy using Lightweight Transformer. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2024, Pp. 7809-7819.

[5] Bochkovskiy, A., Wang, C., & Liao, H.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv, abs/2004.10934*.

[6] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. ***ArXiv, abs/1804.02767***.

[7] https://docs.ultralytics.com/yolov5/tutorials/architecture_description/#1-model-structure

[8] Fanxu Meng, Hao Cheng, Jiaxin Zhuang, Ke Li, and Xing Sun. Rmnet: Equivalently removing residual connection from networks. arXiv preprint arXiv:2111.00687, 2021.

[9] Xu, S., Wang, X., Lv, W., Chang, Q., Cui, C., Deng, K., Wang, G., Dang, Q., Wei, S., Du, Y., & Lai, B. (2022). PP-YOLOE: An evolved version of YOLO. ***ArXiv, abs/2203.16250***.

[10] Huang, X., Wang, X., Lv, W., Bai, X., Long, X., Deng, K., Dang, Q., Han, S., Liu, Q., Hu, X., Yu, D., Ma, Y., & Yoshie, O. (2021). PP-YOLOv2: A Practical Object Detector. ***ArXiv, abs/2104.10419***.

[11] Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., Shen, H., Ren, J., Han, S., Ding, E., & Wen, S. (2020). PP-YOLO: An Effective and Efficient Implementation of Object Detector. ***ArXiv, abs/2007.12099***.

[12] Kang, M., Ting, C., Ting, F.F., & Phan, R.C. (2023). CST-YOLO: A Novel Method for Blood Cell Detection Based on Improved YOLOv7 and CNN-Swin Transformer. ***ArXiv, abs/2306.14590***.

[13] Hurtík, P., Molek, V., Hula, J., Vajgl, M., Vlasánek, P., & Nejezchleba, T. (2020). Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3. ***Neural Computing and Applications, 34***, 8275 - 8290.

[14] Wu, Z., Zou, X., Zhou, W., & Huang, J. (2022). YOLOX-PAI: An Improved YOLOX, Stronger and Faster than YOLOv6.

[15] Wang, J., & Yu, N. (2022). UTD-Yolov5: A Real-time Underwater Targets Detection Method based on Attention Improved YOLOv5. Arxiv.org.

[16] Kang, M., Ting, C., Ting, F.F., & Phan, R.C. (2023). CST-YOLO: A Novel Method for Blood Cell Detection Based on Improved YOLOv7 and CNN-Swin Transformer. ***ArXiv, abs/2306.14590***.

[17] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. ***ArXiv, abs/2209.02976***.

[18] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors (pp. 7464–7475). Institute of Electrical and Electronics Engineers (IEEE).

[19] https://yolov8.org/yolov8-architecture-explained/

[20] Guo, Z., Wang, C., Yang, G., Huang, & Li, G. (2022). MSFT-YOLO: Improved YOLOv5 Based on Transformer for Detecting Defects of Steel Surface. *Sensors*, *22*(9).

[21]Cui, M., Lou, Y., Ge, Y., & Wang, K. (2023). LES-YOLO: A lightweight pinecone detection algorithm based on improved YOLOv4-Tiny network. *Computers and Electronics in Agriculture*, *205*.

[22] Wu, T. H., Wang, T. W., & Liu, Y. Q. (2021). Real-Time Vehicle and Distance Detection Based on Improved Yolo v5 Network. In 2021 3rd World Symposium on Artificial Intelligence, WSAI 2021 (pp. 24–28). Institute of Electrical and Electronics Engineers Inc.

[23] Li, S., Li, Y., Li, Y., Li, M., & Xu, X. (2021). YOLO-FIRI: Improved YOLOv5 for Infrared Image Object Detection. *IEEE Access*, *9*, 141861–141875.

[24] Wang, C., Yeh, I., & Liao, H. (2024). YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. ***ArXiv, abs/2402.13616***.

[25] Grzybowski, A., Pawlikowska-Łagód, K., & Lambert, W. C. (2024). A History of Artificial Intelligence. *Clinics in dermatology*, S0738-081X(23)00268-7. Advance online publication.

[26] McCarthy, J. (1998). WHAT IS ARTIFICIAL INTELLIGENCE.

[27] Tang, S., Fang, Y., & Zhang, S. (2023). HIC-YOLOv5: Improved YOLOv5 For Small Object Detection. ***ArXiv, abs/2309.16393***.

[28] Fan, J., Lee, J., Jung, I., & Lee, Y. (2021). Improvement of object detection based on faster R-CNN and YOLO. 2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC).

[29] Jha, D., Smedsrud, P. H., Riegler, M., Halvorsen, P., De Lange, T., & Johansen, D. (2019b). KVASIR-SEG: a segmented Polyp Dataset. In Lecture Notes in Computer Science (pp. 451–462).

[30] Shussman, N., & Wexner, S. D. (2014). Colorectal polyps and polyposis syndromes. *Gastroenterology report*, *2*(1), 1–15.

[31] ELKarazle, K., Raman, V., Then, P., & Chua, C. (2023). Detection of Colorectal Polyps from Colonoscopy Using Machine Learning: A Survey on Modern Techniques. Sensors (Basel), 23(3), 1225.

[32] Zhao, Z., Zheng, P., Xu, S., & Wu, X. (n.d.). Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems.

[33] Zhu, X., Lyu, S., Wang, X., & Zhao, Q. (2021). TPH-YOLOV5: Improved YOLOV5 based on transformer prediction head for object detection on drone-captured scenarios. arXiv (Cornell University).

[34] Sun, J., Ge, H., & Zhang, Z. (2021). AS-YOLO: An Improved YOLOv4 based on Attention Mechanism and SqueezeNet for Person Detection. ***2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 5***, 1451-1456.

[35] Fan, J., Lee, J., Jung, I., & Lee, Y. (2021b). Improvement of object detection based on faster R-CNN and YOLO. 2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC).

[36] Du, L., Zhang, R., & Wang, X. (2020). Overview of two-stage object detection algorithms. Journal of Physics: Conference Series, 1544(1), 012033.

[37] Linh, T. D., & Arai, M. (2019). Two-stage deep neural network for general object detection. Journal of Information Processing.

[38] Zhang, H., & Cloutier, R. S. (2022). Review on One-Stage object detection based on deep learning. EAI Endorsed Transactions on e-Learning, 7(23), 174181.

[39] Tang, S., Fang, Y., & Shu, Z. (2023). HIC-YOLOV5: Improved YOLOV5 for small object detection. arXiv (Cornell University).

[40] Suhail, K., & Brindha, D. (2024). Microscopic urinary particle detection by different YOLOv5 models with evolutionary genetic algorithm based hyperparameter optimization. Computers in Biology and Medicine, 169, 107895.

[41] Li, K., Fathan, M. I., Patel, K., Zhang, T., Zhong, C., Bansal, A., Rastogi, A., Wang, J. S., & Wang, G. (2021). Colonoscopy polyp detection and classification: Dataset creation and comparative evaluations. *PloS one*, *16*(8), e0255809.

[42] Real-Time Polyp Detection in Colonoscopy using Lightweight Transformer. (2024, January 3). IEEE Conference Publication | IEEE Xplore.

[43] Ragab, M. G., Abdulkadir, S. J., Muneer, A., Alqushaibi, A., Sumiea, E. H., Qureshi, R., Al-Selwi, S. M., & Alhussian, H. (2024). A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023). IEEE Access, 1.

[44] Huang, S., Sirejiding, S., Lu, Y., Ding, Y., Liu, L., Zhou, H., & Lu, H. (2024). YOLO-MED : Multi-Task Interaction Network for Biomedical Images. *ArXiv, abs/2403.00245*.

[45] Aldughayfiq, B., Ashfaq, F., Jhanjhi, N. Z., & Humayun, M. (2023). YOLO-Based Deep Learning Model for pressure Ulcer detection and Classification. Healthcare, 11(9), 1222.

[46] Elyan, E., Vuttipittayamongkol, P., Johnston, P., Martin, K., McPherson, K., Moreno-García, C. F., Jayne, C., & Sarker, M. M. K. (2022). Computer vision and machine learning for medical image analysis: recent advances, challenges, and way forward. Artificial Intelligence Surgery.

[47] Ali, S., Jha, D., Ghatwary, N., Realdon, S., Cannizzaro, R., Salem, O. E., Lamarque, D., Daul, C., Riegler, M. A., Anonsen, K. V., Petlund, A., Halvorsen, P., Rittscher, J., De Lange, T., & East, J. E. (2023). A multi-centre polyp detection and segmentation dataset for generalisability assessment. Scientific Data, 10(1).

[48] Li, K., Fathan, M. I., Patel, K., Zhang, T., Zhong, C., Bansal, A., Rastogi, A., Wang, J. S., & Wang, G. (2021). Colonoscopy polyp detection and classification: Dataset creation and comparative evaluations. PloS One, 16(8), e0255809.

[49] Pacal, I., Karaman, A., Karaboga, D., Akay, B., Basturk, A., Nalbantoglu, U., & Coskun, S. (2022). An efficient real-time colonic polyp detection with YOLO algorithms trained by using negative samples and large datasets. Computers in Biology and Medicine, 141, 105031.

[50] Lalinia, M., & Sahafi, A. (2023). Colorectal polyp detection in colonoscopy images using YOLO-V8 network. Signal, Image and Video Processing.

[51] Tan, M., & Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv, abs/1905.11946*.

[52] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *Neural Information Processing Systems*.

[53] Sirisha, U., Praveen, S. P., Srinivasu, P. N., Barsocchi, P., & Bhoi, A. K. (2023). Statistical analysis of design aspects of various YOLO-Based deep learning models for object detection. ˜the œInternational Journal of Computational Intelligence Systems/International Journal of Computational Intelligence Systems, 16(1).

[54] LADD: Lacmus Drone Dataset - Dataset Ninja. (n.d.). Dataset Ninja.

[55] Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2016, December 9). Feature pyramid networks for object detection. arXiv.org.

[56] Popescu, M., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. E. (2009). Multilayer perceptron and neural networks. ResearchGate.

[57] Yang, R., & Yu, Y. (2021). Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis. Frontiers in Oncology, 11.

[58] Shou, Y., Meng, T., Ai, W., Xie, C., Liu, H., & Wang, Y. (2022). Object detection in medical images based on hierarchical transformer and mask mechanism. Computational Intelligence and Neuroscience, 2022, 1–12.

[59] Zhou, T., Ye, X., Lu, H., Zheng, X., Qiu, S., & Liu, Y. (2022). Dense convolutional network and its application in medical image analysis. BioMed Research International, 2022, 1–22.

[60] Arabahmadi, M., Farahbakhsh, R., & Rezazadeh, J. (2022). Deep Learning for Smart Healthcare—A Survey on Brain Tumor Detection from Medical Imaging. Sensors, 22(5), 1960.

[61] Huang, Y., Yang, X., Liu, L., Zhou, H., Chang, A., Zhou, X., Chen, R., Yu, J., Chen, J., Chen, C., Liu, S., Chi, H., Hu, X., Yue, K., Li, L., Grau, V., Fan, D., Dong, F., & Ni, D. (2024). Segment anything model for medical images? Medical Image Analysis, 92, 103061.

[62] Ou, S., Gao, Y., Zhang, Z., & Shi, C. (2021). Polyp-YOLOV5-Tiny: a lightweight model for Real-Time Polyp Detection. 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA).

[63] Pacal, I., & Karaboga, D. (2021). A robust real-time deep learning based automatic polyp detection system. Computers in Biology and Medicine, 134, 104519.

[64] Nogueira-Rodríguez, A., Domínguez-Carbajales, R., Campos-Tato, F., Herrero, J., Puga, M., Remedios, D., Rivas, L., Sánchez, E., Iglesias, Á., Cubiella, J., Fdez-Riverola, F., López-Fernández, H., Reboiro-Jato, M., & Glez-Peña, D. (2021). Real-time polyp detection model using convolutional neural networks. Neural Computing & Applications, 34(13), 10375–10396.

[65] Yu, T., Lin, N., Zhang, X., Pan, Y., Hu, H., Zheng, W., Liu, J., Hu, W., Duan, H., & Si, J. (2022). An end-to-end tracking method for polyp detectors in colonoscopy videos. Artificial Intelligence in Medicine, 131, 102363.

[66] https://www.algotive.ai/blog/what-is-computer-vision-and-how-does-it-work-with-artificial-intelligence