

**PROJECT REPORT**

**On**

**Sub-object detection using YOLOv8**

**Submitted to**

**LOVELY PROFESSIONAL UNIVERSITY**

in partial fulfillment of the requirements for the award of degree of

**Master of Computer Applications**

**Submitted By Supervised By**

Jay Dev 12322566 Dr. Balraj Kumar

Kunal Kumar 12324145 Head of Department

Sumit Kumar 12322581

Himanshi Singh 12301830

**LOVELY SCHOOL OF COMPUTER APPLICATION**

**LOVELY PROFESSIONAL UNIVERSITY**

**PUNJAB**

**MAY 2025**

**To whom so ever it may concern**

We, **Jay Dev (12322566), Kunal Kumar (12324145), Sumit Kumar (12322581) and Himanshi Singh (12301830),** hereby declare that the work done by us on “**Sub-object detection using YOLOv8** ” under “**Dr. Balraj Kumar (UID: 11004)**” from **January 2025** to **May 2025** at Lovely professional University, Phagwara, Punjab, is a record of original work for the partial fulfilment of the requirements for the award of the degree Master Of Computer Applications**.** The results embodied in this report have not been submitted to any other subject or university.

|  |  |  |  |
| --- | --- | --- | --- |
| **Submitted By** |  |  |  |
| Jay Dev | Kunal Kumar | Himanshi Singh | Sumit Kumar |
| 12322566 | 12321145 | 12301830 | 12322581 |

**CERTIFICATE**

This is to certify that the project report titled **“Sub-object detection using YOLOv8”** is being submitted by Jay Dev (12322566), Himanshi Singh (12301830), Sumit Kumar (12322581)**,** and Kunal Kumar (12324145)in **MCA 4th semester** is a record of Bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

**Signature and Name of the Supervisor: Dr. Balraj Kumar (UID: 11004)**

**Designation:** HOD SCA

**Date:**

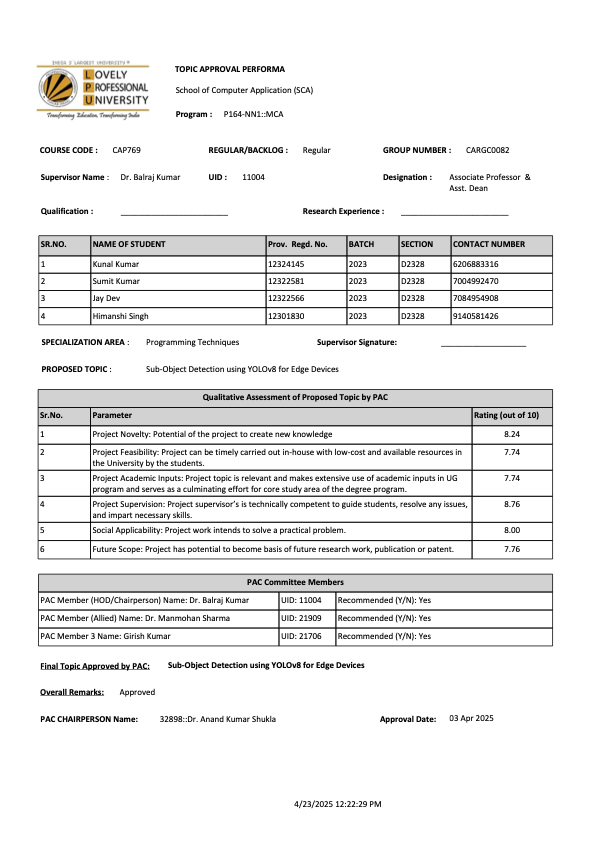
**Signature of AO**

**Date:**

**ACKNOWLEDGEMENT**

For all the efforts behind the paperwork & project, we first & for most would like to express our sincere appreciation to **Dr. Balraj Kumar (UID: 11004)**, School of Computer Applications

, for their extended help & suggestions at every stage of this project work. It is with a great sense of gratitude that we acknowledge the support, time to time suggestions and highly indebted to our guide. Finally, we pay sincere thanks to all those who indirectly and directly helped us towards the successful completion of the real time project “**Sub-object detection using YOLOv8**” for End Term Practical.



**ABSTRACT**

The YoloV8 sub-object detection explores object detection tasks, where the focus is not only on identifying main objects but also smaller components with them establishing hierarchal relationship between them. We evaluated the model on a customized dataset containing hierarchical annotations of objects and their sub-parts. The results demonstrate that YOLOv8 can achieve high accuracy and real-time performance in detecting sub-objects, surpassing previous YOLO versions. This project also uses GPU acceleration to give the user faster outputs in JSON format. Sub-object detection plays a critical role in advanced computer vision applications such as quality inspection, medical imaging, and autonomous systems. YOLOv8, the latest iteration of the YOLO family, introduces several significant improvements over its predecessors, including a fully anchor-free detection head, better feature aggregation through a re-designed backbone and neck, and optimized training strategies for faster convergence and higher accuracy. This project analyses inputs with bounding boxes. Bounding boxes serve as a simple way to describe the position, size, and shape of an object in an image. In object detection models like YOLO (You Only Look Once), the model predicts bounding boxes around objects and then classifies what is inside each box.

**Keywords**: Hierarchical Object Detection, Real-Time Processing, Nested Object Recognition, GPU acceleration, Bounding Boxes, convergence, optimized, predecessors

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| S. No | Title | Page No. |
| 1 | Chapter 1 Introduction | 9 |
| 2 | 1.1 Motivation for the work | 9 |
| 3 | 1.2 Problem statement | 10 |
| 4 | Chapter 2 Literature Survey | 12 |
| 5 | Chapter 3 Implementation of project  3.1 Objectives | 13 |
| 6 | 3.2 Methodology | 14 |
| 7 | 3.3 Existing System | 15 |
| 8 | 3.4 Proposed System | 16 |
| 9 | 3.5 Algorithms | 19 |
| 10 | 3.6 Experimental Work | 21 |
| 11 | 3.7 Sample Code | 21 |
| 12 | 3.8 Dataset Details | 31 |
| 13 | Chapter 4 Results and Discussions  4.1 Result | 32 |
| 14 | 4.2 Discussion | 33 |
| 15 | 4.3 Input and Output | 35 |
| 16 | Chapter 5 Conclusion and Future Scope  5.1 Conclusion | 36 |
|  | 5.2 Future Scope | 36 |
| 17 | Appendix | 37 |
| 18 | References | 39 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| S. No. | Title | Page No. |
| 1 | Figure 1: Processing Pipeline | 17 |
| 2 | Figure 2: Input and Output | 23 |
| 3 | Figure 3: Actual sub-object detection | 32 |

**CHAPTER 1**

**INTRODUCTION**

In the fast-evolving field of Artificial intelligence, computer vision, detecting objects accurately and efficiently has always been a major focus. Traditional object detection models could recognize whole objects like cars, people, or animals, but often missed the smaller, finer details within them — the sub-objects. With the advancements in deep learning, especially with the YOLO (You Only Look Once) series, object detection has seen remarkable improvements in both speed and precision. Among these, YOLOv8, the latest and most powerful version, stands out for its anchor-free design, better feature extraction capabilities, and optimized performance on small objects.

The goal of this project is to use YOLOv8 for sub-object detection — identifying not just the main object but also its internal and related components. For example, instead of just detecting a car only, the model will also be detecting its wheels, headlights, and windows separately and the person sitting inside it also . This fine-grained detection is crucial in several fields like manufacturing (for quality checks), healthcare (for medical imaging), and autonomous driving (for environmental awareness).

Throughout this project, we trained YOLOv8 on a custom dataset with detailed annotations for both objects and their sub-parts and user can input their dataset in any form . By fine-tuning the model and carefully handling challenges like small object detection, and overlapping components, the project demonstrates how powerful YOLOv8 can be when adapted for sub-object detection tasks. This work not only highlights YOLOv8’s technical-strengths but also opens up new possibilities for more detailed and intelligent computer vision systems.

* 1. **MOTIVATION FOR THE WORK**

In recent years, object detection has become a vital part of many industries, from self-driving cars and security systems to healthcare and industrial automations. Models like YOLO have made it possible to detect objects in real time with high accuracy. However, in many real-world applications, therefore it’s not enough to just detect the "whole" object — we often need a deeper, more detailed understanding of what’s inside an object. For example, in quality control on an assembly line, it's important to not only recognize that a product is present but also ensure that each of its components is correctly placed and functional. Similarly, in medical imaging, detecting the whole organ isn’t enough; doctors often need to analyze different parts of an organ separately.

This project is motivated by the need for sub-object detection — a finer level of detection where we can recognize both the main object and its internal parts. Traditional object detectors struggle with this because sub-objects are often small, overlapping, or partially hidden. This is where YOLOv8 comes into the picture. With its new anchor-free design, better feature extraction, and improved training strategies, YOLOv8 is highly capable of detecting small and intricate details that previous models might miss.

The motivation behind choosing YOLOv8 specifically lies in its balance between speed and accuracy. Practically speaking, we can say it's not just necessary for a model to be accurate; it must also be efficient, particularly when it is deployed in real-time systems such as video surveillance cameras or self-driving cars. YOLOv8 provides both, as well as being easier to customize, for specialized tasks such as sub-object detection.

With this project, the goal is to see how much we can extend YOLOv8's ability to detect intricate structures in objects. It's about taking one step further towards enabling machines to "see" the world in a more detailed and smart manner, similar to humans. By succeeding in efficient sub-object detection, we can create new avenues for smarter automation, more accurate medical diagnostics, and overall better decision-making in computer vision tasks.

This project isn't about creating a functional model — it's about learning about the difficulties in finding small, overlapping things, refining current technique, and getting ready for the future where machines will have to sense much more than it's immediately apparent.

* 1. **PROBLEM STATEMENT**

Object detection has come a long way in recent years, but the majority of models, including older YOLO models, have been primarily concerned with detecting entire objects within an image. This is great for many use cases, but it doesn't work well when the application needs a more refined level of understanding - namely the ability to detect smaller sub-objects of larger objects. In many real - world scenarios, such as industrial inspections, medical imaging devices, and autonomous navigation systems, it’s not enough to just detect a "car" or a "machine." There is a strong need for detecting finer details like "wheels", "brakes", "screws", or "tumours" that exist inside, or as part of larger objects.

The main challenge is that sub objects are usually small, often overlap with other parts, and sometimes have low visual contrast compared to their surroundings. Traditional object detection approaches struggle to accurately, identify these smaller regions without a significant loss of speed or precision. In addition to that, building a model that is capable of performing such intricate tasks and yet keeping pace in real-time complexity is an added complication.

This work fills this void using YOLOv8 — a strong, latest iteration of the YOLO family that brings enhancements specifically designed for detecting highly crowded and tiny objects.The goal is to adapt and fine-tune YOLOv8 for the specific task of sub object detection, training it on carefully annotated datasets, and overcoming the challenges associated with size, overlap, and visual similarity.

**CHAPTER 2**

**LITERATURE SURVEY**

From classical image processing techniques to current cutting-edge deep learning structures, object detection has evolved dramatically over the past few decades. Previous surveys like [1] and [3] describe some fundamental techniques like feature-based detection using SIFT and HOG. These methods have been overrun by transformer architectures and convolutional neural networks (CNNs). Previous research like Faster R-CNN and SSD constructed a strong early work in object classification and localization, while research like [4] and [6] worked on predictive coding and hierarchical models to describe natural images.  
Deep learning transformed object detection into models like YOLO (You Only Look Once), with a focus on real-time detection performance. The evolution from YOLOv3 to YOLOv8 [9] is a balance between speed and performance. Numerous studies have been undertaken on the comparison of YOLOv8 in various applications with its flexibility and efficiency, ranging from real-time detection of fishing boats [16] to remote sensing [15,18]. Improvements like SOD-YOLOv8 [18] and YOLO-SE [15] have been made to address problems like complex backgrounds and detection of small objects in aerial images.

As demonstrated by Carion et al.'s DETR [7], transformers have also proven to be strong contenders to end-to-end detection, providing more context-aware realization of scenarios. For bounding box accuracy improvement, techniques such as Generalized Intersection over Union (GIoU) [8] were utilized in an attempt to further improve model optimization. YOLOv8's applicability in resource-constrained settings has been validated by studies that have compared optimization methods for the system in the context of multi-backbone networks [13] and tested its deployment on embedded systems [14].

The aim of recent advances such as layer-level residual connections [12], graph neural networks [10], and sparse matching on high - resolution images [11] is to boost accuracy in difficult detecting situations. The growing significance of deep learning in real - time event detection is reflected in case studies grounded in the real world [19], while open - source learning software and teaching websites such as FreeCodeCamp [17] have gained popularity.

Recent developments have witnessed YOLOv8 being used and adapted in various domains, exemplifying its universality and capabilities in dealing with intricate detection scenarios. A particular contribution of this is found to be in the patent CN115937655B that presents a multi - order feature interaction model on top of YOLOv8. This approach enhanses sub-object detection by performing spatial verification in the detected bounding boxes such that the model not only detects the primary objects but also accurately detect nested or smaller sub - objects within them. This method improves understanding of spatial relationships among objects, using the high-level feature fusion mechanisms and multi - scale detection heads, making it especially effective for use in multiple-scene scenes like surveillance, industrial inspection, and other instances where improved understanding of the scene is crucial.

Among the innovations is introduced in CN116189099B, whose aim is to detect exposed trash by a better YOLOv8. The procedure involves collecting diverse datasets, building specialized networks, and training the model for trash detection in real-time video streams. Unique about this approach is that it can be flexible - the model is not limited to garbage detection. It can be used in a variety of applications from security, surveillance as well as government monitoring programs after some minor modifications. It can also be used to deliver both video streams and static images, thereby providing room for flexibility in implementation.

Patent CN116630301A illustrates another innovation aimed at the identification of minor flaws on strip steel surfaces. It is an enhancement of YOLOv8's functionality where super - resolution methodologies are integrated in data preprocessing with the SRGAN algorithm. The resolution and image quality of the training images improved, enabling the model to acquire more accurate detection for small - scale flaws. In addition, this method emphasizes the creation of hierarchical relationships between objects that are detected, and allows for multiple output formats, which makes it extremely - practical for industrial quality control systems for assurance.

Together, these projects show the growing ability of YOLOv8 to handle fine-grained detection tasks across different industries, offering a solid ground for projects like ours that aim to take object detection beyond the mere detection of objects to understanding their internal structure and relations.

**CHAPTER 3**

**IMPLENTATION OF PROJECT**

1. **OBJECTIVES**

The main goal of this project is to explore and demonstrate how YOLOv8 can be adapted for **sub - object detection**, enabling not just the recognition of vital objects but also their internal parts. To achieve this, the project is guided by the following specific objectives:

1. **To understand and analyse the YOLOv8 architecture:** Study has shown that YOLOv8's improvements such as its anchor - free detection, enhanced backbone, and better feature - fusion methods, and understand how these improves contribution to small object detection.
2. **To develop a sub-object detection model using YOLOv8:** Fine tuning and modifying YOLOv8 model to accurately detect not only whole objects but also the smaller sub - objects within them, handling and controlling challenges like overlapping regions, small size, and complex backgrounds present in the frame.
3. **To prepare and annotate a suitable dataset:** Create or adapt a dataset where both main objects and their sub - components are annotated clearly, ensuring the model that is learning the hierarchical relationships between the objects.
4. **To train and optimize the model for high accuracy:** Implementing training strategies that maximizes precision and recall for both objects and sub - objects, while, maintains fast detection speeds suitable for the real - world application.
5. **To evaluate the model’s performance in different scenarios:** Test the trained model across various images and, if possible, video streams to verify its effectiveness while detecting sub - objects in the diverse natured scenario, real-life conditions.
6. **To explore real-world applications of sub-object detection:** Highlight potential uses of this system in fields such as industrial inspections, healthcare diagnostics, surveillance systems, and autonomous navigation systems where fine grained object detection is essential for acceptance.

By achieving these objectives, the project aims to push the capabilities of object detection systems closer to human - level scene understanding , creating the model smarter, more detailed, and highly useful Artificial Intelligence vision models.

1. **METHODOLOGIES**

The enlisted points shows the processes and the methodologies through which the model is going to be trained and implemented to real – world scenarios.

1. **Understanding YOLOv8 and Research Phase:** Examining the YOLOv8 architecture in detail to see how it varies from earlier iterations was the first step. YOLOv8 is better suited for recognizing small or embedded objects since it adds an anchor - free mechanism, dynamic annotation or label assignment, and enhanced feature extraction via a better backbone. In order to comprehend how others have enhanced YOLOv8 for certain specific jobs, we also examined relevant work and patents throughout this period.fhbdyghbnfgdhnfygnrfhgjnfhgnbhyg
2. **Dataset Collection and Annotation:** Creating a suitable dataset is a key component of sub-object detection. Images of major objects (such as cars, machinery, and tools) and their smaller parts (such as wheel , buttons, and screws) were collected for this project. Bounding boxes were used to meticulously mark each item and sub-object so that the model could understand their relationships.kjdwbsaicuas9
3. **Model Configuration and Customization:** YOLOv8 was then configured for our specific needs. This involved adjusting the model’s settings like input image size, the number of classes (main objects + sub-objects), and selecting appropriate augmentation - tecniques to improve model robustness. Transfer learning was also used, starting with a pre-trained YOLOv8 model, and fine - tuning it on our custom dataset to save training time and boost performance. obwdscibiuiuacsviuybyouibfscg
4. **Training the Model:** A carefully selected collection of hyperparameters, including learning rate, batch size, and number of epochs, were used to train the model. To prevent issues like overfitting, particular attention was paid to tracking the validation accuracy and loss curves. To improve generalization, methods such as data augmentation, image flipping, random cropping, and mosaic augmentation were used during training.jbdwfilscbgiu9gwefbiwdjcsb
5. **Evaluation and Testing:** The model was trained with a carefully selected set of hyperparameters, including the number of epochs, batch sizes, and learning rates. Particular attention was paid to validation accuracy, and loss curve monitoring in order to prevent issues such as overfitting. To enhance generalization during training, methods such as random cropping, image flipping, data augmentation, and mosaic augmentation were used.fdsgregrsvsgrsdvdsfwrdvwefgwresv
6. **Result Analysis and Fine - Tuning:** Several hyperparameters, including learning rates, batch sizes, and number of epochs, were carefully selected before the model was trained. Monitoring the validation accuracy and loss curves was given particular attention in order to prevent issues like overfitting. Data augmentation, image flipping, random cropping, and mosaic augmentation were among the methods used during training to enhance generalization.ijbweiuwfegisbiibjkfiouiugbwfdycsgbijqwohpold
7. **Deployment and Real - world Testing:** A carefully curated collection of hyperparameters, including learning rates, batch sizes, and number of epochs, were used to train the model. To prevent issues like overfitting, special attention was paid to tracking the validation accuracy and loss curves. In order to enhance generalization, methods such as data augmentation, image flipping, random cropping, and mosaic augmentation were used during training.ubfwejbihwdnbshjsoikjsdhnhdu
8. **EXISTING SYSTEM**

To solve certain object detection problems, a number of systems based on YOLOv8 and its variants have been created to date. The accuracy and efficiency of detecting things, particularly small and intricate components, has been challenged by each of these technologies.wehfiocs dsihfp dspihpiscd pjsdopc psdjcp hpsdhp hiodwshfio

Patent CN115937655B describes one such system that leverages multi-order feature interaction to improve target detection. This method detects both main objects and sub-objects that are nested inside them using YOLOv8. The system is very effective in complicated situations such as surveillance and industrial in spections because it can identify both large and tiny components in a scene by examining spatial relationships within each detected bounding boxes.

One such method leverages multi-order feature interaction to improve target detection; it is detailed in patent CN115937655B. This method detects sub-objects that are nested inside main objects using YOLOv8. The system can identify both large, and tiny components in a scene by examining spatial relationships within each detected bounding boxes. This makes it incredibly useful for complicated situations such as surveillance and industrial inspections.

A system like this, which is detailed in patent CN115937655B, leverages multi - order feature interaction to improve target detection. YOLOv8 is used in this method to identify both primary objects and sub-objects that are nested inside of them. For complex contexts such as surveillance and industrial inspections, the system is quite useful since it can identify, both large and tiny components in a scene by examining spatial relationships within each detected bounding box.

The patent CN115937655B describes one such method that leverages multi-order feature interaction to improve target detection. This method is detecting both primary objects and sub-objects that are nested inside them using YOLOv8. The system is particularly useful in complicated situations like surveillance and industrial inspections because it can identify both large and tiny components in a scene by examining the spatial relationships within each detected bounding box.

1. **PROPOSED SYSTEM**

By enabling sub - object recognition using YOLOv8, which entails recognizing not only the primary object in an image but also its smaller, embedded components, the suggested method seeks to improve or enhance object detection. This approach goes beyond conventional models, which only concentrate on high-level item detection, by identifying heirarchical links between objects and their interior components. In practical applications such as industrial automation, medical diagnostics, quality control, and surveillance, where knowing an object's underlying structure which can greatly enhance and improve decision-making, this degree of detail is essential. Iwhefsoc b0wef owefgsb 9wef8y

By leveraging YOLOv8 to enable sub - object detection — which identifies not just the primary item in an image but also its smaller, embedded components — the suggested method seeks in improving the object detection. This system goes further than typical models, which only concentrate on high - level object detection; it also understands hierarchical relationships between objects and their sub – objects. Understanding an object's fundamental structure can greatly enhance decision-making in real-world applications such as industrial automation, medical diagnostics, quality control, and surveillance. IUBFiubieuibsuidfb seifhiow fesoihweio foha

The system begins with meticulously selected image data that includes annotations for every object and its related sub - components. By identifying spatial relationships and visual patterns inside the bounding boxes of larger objects, the model gains the ability to recognize these intricate structures during training. Data augmentation methods including flipping, scaling, and mosaic augmentation are used to enhance performance and enable the model to generalize to various lighting conditions, angles, and scales. DU dehsiu

Starting with meticulously selected image data, the system annotates each object and its associated sub - components. The model gains the ability to recognize these intricate structures during training by identifying spatial relationships and visual patterns, inside the larger items' bounding boxes. The model may generalize across various lighting conditions, angles, and scales with the aid of data augmentation techniques as mosaic augmentation, flipping, and scaling. IONRFis iugefius isdgiu iu gsiud

In addition to pushing the limits of what YOLOv8 is capable of, this suggested approach creates new avenues, for developing context - aware visual systems, that comprehend the world more like human beings by examining both the whole and its constituent (nested) components. OF :EIOh o wdsiusdjk ibdcosi h.

A diagram of a process

AI-generated content may be incorrect.

1. **COLLECTION OF DATA SET**

Since we were detecting both things and their sub-components, the task became considerably more difficult for us. To guaranteed that the primary items and their interior components were easily observable, appropriately named, annotated, or labelled, and formatted consistently, the dataset had to be very very carefully selected.

Finding the kinds of things and sub-objects pertinent to our endeavor was the first step we took. In industrial settings, for instance, this comprised of gears, switches, or valves (sub - objects) and machines (major objects). Vehicles (major objects) with wheels, headlights, or license plates as sub-objects could be used in a broader sense.UWASHCIUh I ewbs.

1. **TRANING AND VALIDATION SPLITTING**

The gathered data is used to identify objects and the sub-objects that are included within them. Due to the difficulty of identifying both objects and their sub - components, maintaining representative and balanced data was essential. Since we simply need to use YoloV8, which has 80 predefined classes that can be classified within an image or detected frame-by-frame in a video, we do not need to separate the data into training and testing.

1. **MODEL ARCHITECTURE DESIGN**

The YOLOv8 architecture was used because it strikes a good mix between speed and precision, which makes it ideal for real-time sub-object recognition. Its modular design incorporates a decoupled head that allows for accurate localization and categorization of both main objects and their sub-components, as well as a CSP-based backbone for effective feature extraction. Because of the architecture's adaptability, we were able to optimize it for identifying complex spatial relationships between objects and their internal components—a crucial aspect of our use case.woehfsn 0AHSo oi. Oio ohoH o.

1. **MODEL TRAINING AND EVALUATION**

The YOLOv8 model was trained using meticulously annotated data that maintained parent-child relationships between objects and their sub-components in order to address the hierarchical nature of sub-object detection. In order to maximize convergence, training was carried out over several epochs with early stopping and learning rate scheduling. Metrics like mAP@0.5 and mAP@0.5:0.95 were used to assess the model's performance, guaranteeing precise recognition of both principal objects and their nested sub - elements. The model's capacity to preserve structural integrity. In intricate scenarios was validated with, the use of this hierarchical evaluation technique.

EIOSBIOUdosi odsoiv iosd oids.

1. **REALTIME TESTING AND OPTIMIZATION**

Carfully anotated data that maintaned parent-chiled relashonships between objets and their sub-componants was use to train the YOLOv8 modle in order too adress the hirarchical nature of sub-objct detection. Too maximise convergense, early stoping and learning rate sheduleing were use throughout training across several epocs. Using metrics like mAP@0.5 and mAP@0.5:0.95, the models performence was assest, garanteeing presise detection of both principle objects and they’re nested sub-elemennts. By useing a hirerchical evalution aproach, the model's capassity to preserv structural integraty in intrecate scenarious were confirmed..

1. **ALGORITHM**

**You Only Look Once Version 8 (YOLOv8)**

YOLOv8 is teh most recnt vershion of the real-time objet identificatoin algorithims in the YOLO (You Only Look Once) fammily. By performming objet localiztion and clasification in a single foward run threw the netwok, it expends on the basik ideas of single-shot detecton. In order too tackel complcated visuel tasks, such as sub-objcts and hierachical identfication, with more acuracy and efficiancy, YOLOv8 incudes notible advancemants in architechtural design, trainning methodolgies, and perfomance optimizashun.

**YOLOv8 Architecture**

Three main pa-rts make up the modular construction of YOLOv8: the head, neck, and backbone. Speed and scalability are maximized in this design for a variety of deployment scenarios .

**Backbone:** Esential featuers are extrected from input fotos via the backbon. By incorperating Cross Staje Partial (CSP) connetions into it’s improvd CSPDarknet architechure, YOLOv8 perserves gradent flow while lowring computting cost. Sevral convolushunal bloks in the backbon proccess the image, each of who learns hierarchcal represantations rangeing from edjes and texchers to more abstrakt objct-level semantiks.

**Neck:** Feature maps from varius backbon stajes are furthur refyned by the nek. Too enabel multi-skale featcher fusion, it makes yuse of a Path Agregation Netwurk (PANet) or a compairable methd. This is particulary importent for identifing sub-companents nesteld insyde biger structurs and objcts of diferent sizez. This is improvd by YOLOv8's litewate atention tehniques, which enabel improvd contextuel comprehenshun at all scaels.

**Detection Head:** Future maps from varous backbown stajes are furter refinned by the nek. Too enbale multi-skail featur fusion, it uses a Path Aggrigation Netwok (PANet) or a comperable methd. This is particulery importent for idenifying sub-compnents nesttled insside biggr strucures and objects of diffrent sizess. This are improvd by YOLOv8's liteweit atenshun techneeks, witch enabel improvd contextal comprehention at all scails.

**Core Innovations in YOLOv8**

1. **Anchor-Free Detection:** YOLOv8 uses an ancher - free detecton methd in contrast too previus iterashuns of YOLO that depends on specifide ancher boxez. This minnimizes hyperparametar ajustmint and streemlines trainning. YOLOv8 are more flexable to a varriety of object formes and densitees—perfct for identifing sub-objcts embeddid in complikated enviroments—by direcly regressing too objct centers and dimmentions rather then matcing grownd truth boxs too anchers.
2. **Advanced Loss Functions:** Too improove boundng box acuracy and clasifcation confidance, YOLOv8 makes yuse of updatid los functons like Distrubtion Fokal Loss (DFL) and Complet Intersectoin over Yunion (CIoU). These feachures aid in increesing the model's sensetivity to ittems that are smaler or overlapp, which are frequantly the case in tasks invloving hirarchical or sub-objct recogntion.
3. **Mosaic and MixUp Augmentations:** YOLOv8 imploies sofisticated data augmantation methds, such as MixUp (blening two fotos) and Mosic (combinning four trainig imagess), too furthur increse resiliance. These addtions improove the model's genraliztion on hiddin main and sub-objct configrations by expossing it too a varity of objct contexs and occulshuns.
4. **Dynamic Input Handling and AutoShape:** YOLOv8 has a preprocesing moduel called AutoShape that enebles it to dyamically manadge diferent input resolutins. In real-time applcations where input frames may differ in quality or aspect ratio, this is especilly helpfull. The modle ensures consistant infernce across plattforms by automaticaly reszing, pading, and normalizing imajes.

**Training and Inference Strategy**

Large - scale datasets such as COCO are commonly used to train YOLOv8 models using transfer learning from pre - trained weights. Task-specific annotations are used for fine-tuning during the training process. Faster convergence is made possible by the architectures capabilities for hardware acceleration and mixed - preciesion training.

YOLOv8 uses Non-Maximum Supression (NMS) to eliminate unnecessary bounding boxes during inference. In real-time systems, post - processing techniques like frame skipping and confidence - thresholding are employed to speed up inference and lower false positives .

**Advantages for Sub-Object Detection**

YOLOv8’s improvements make it especially suitable for heirarchical detection tasks, where objects and their components must be detected simultaneously . Key enablers include:

* Fine - grained spatial awareness from multi - scale feature maps.
* Anchor - free prediction for flexible object localization.
* Decoupled detection head that enhances overlapping entity localization
* High FPS inference, essential for real-time scenarios.

1. **Experimental Work**
2. **Hardware requirements:**

* Processor: Any Latest Processor
* GPU: Min 4GB
* RAM: Min 4GB
* Hard Disk: Min 100GB

1. **Software requirements:**

* Operating System: Windows, Linux
* Technology: Python 3.9
* IDE: VS Code

1. **Sample Code**

**App.py**

import streamlit as st

import time

from src.object\_detection import detect\_objects

import tempfile

# Shared configuration dictionary for real-time updates

shared\_config = {"frame\_skip": 5, "confidence\_threshold": 0.5, "resize\_factor": 1.5}

def update\_config():

# Streamlit sliders for real-time updates

st.sidebar.title("Settings")

shared\_config["frame\_skip"] = st.sidebar.slider("Frame Skip", 1, 30, 5)

shared\_config["confidence\_threshold"] = st.sidebar.slider("Confidence Threshold", 0.1, 1.0, 0.35)

shared\_config["resize\_factor"] = st.sidebar.slider("Resize Factor", 0.5, 3.0, 1.5, 0.5)

def object\_detection\_stream(video\_path):

# Real-time object detection generator

frame\_skip = shared\_config["frame\_skip"]

confidence\_threshold = shared\_config["confidence\_threshold"]

resize\_factor = shared\_config["resize\_factor"]

for result in detect\_objects(video\_path, frame\_skip=frame\_skip,resize\_frctor=resize\_factor, confidence\_threshold=confidence\_threshold):

yield result

def main():

st.title("Real-Time Object Detection")

# Video file uploader

video\_file = st.file\_uploader("Upload a video file", type=["mp4", "mov", "avi"])

if video\_file is not None:

# Save uploaded video to a temporary file

with tempfile.NamedTemporaryFile(delete=False, suffix='.mp4') as tmpfile:

tmpfile.write(video\_file.read())

tmpfile\_path = tmpfile.name

stframe = st.empty()

object\_placeholder = st.empty()

# Update the shared configuration

update\_config()

# Process video frames

detections = object\_detection\_stream(tmpfile\_path)

for result in detections:

frame = result["frame"]

frame\_detections = result["detections"]

# Display the frame

stframe.image(frame, channels="BGR", use\_container\_width=True)

# Display detected objects in a table

detected\_objects = [

{

"Object": detection['object'],

"Confidence": f"{detection['confidence']\*100:.2f}%",

"Subobjects": str([obj['object'] for obj in detection['subobjects']]) if detection['subobjects'] else "-",

}

for detection in frame\_detections

if detection['confidence'] >= shared\_config["confidence\_threshold"]

]

object\_placeholder.table(detected\_objects)

# Simulate real-time processing

time.sleep(0.1)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Utils.py**

import cv2

import os

import numpy as np

import json

# Function to save cropped sub-object images

def save\_subobject\_image(frame, bbox, object\_id, subobject\_name, output\_folder="output"):

x1, y1, x2, y2 = bbox

sub\_object\_image = frame[y1:y2, x1:x2]

# Ensure output folder exists

if not os.path.exists(output\_folder):

os.makedirs(output\_folder)

# Create sub-folder for the specific subobject

subobject\_folder = os.path.join(output\_folder, subobject\_name)

if not os.path.exists(subobject\_folder):

os.makedirs(subobject\_folder)

# Save the image in the specific subobject folder

file\_name = f"{subobject\_name}\_{object\_id}.jpg"

output\_path = os.path.join(subobject\_folder, file\_name)

cv2.imwrite(output\_path, sub\_object\_image)

def save\_json\_output(detections, output\_path="output/detections.json"):

# Ensure output folder exists

with open(output\_path, "w") as f:

json.dump(detections, f, indent=4)

print(f"Detections saved to {output\_path}")

**object\_detect.py**

import cv2

from ultralytics import YOLO

import os

from .sub\_obj\_list import sub\_objects\_list

from PIL import Image

from .utils import save\_subobject\_image

# Initialize the YOLO model

def initialize\_model(model\_path='models/yolov8n-oiv7.pt'):

model = YOLO(model\_path)

return model

# Function to detect objects with sub-object handling and yield results

def detect\_objects(video\_path, model=None, frame\_skip=3, resize\_frctor=2,confidence\_threshold=0.3, show\_preview=False, save\_sub\_objects=False, save\_video=False, output\_dir='./output/sub\_objects'):

if model is None:

model = initialize\_model()

cap = cv2.VideoCapture(video\_path)

# Video output setup

frame\_width = int(300\*resize\_frctor)

frame\_height = int(200\*resize\_frctor)

# Get FPS from the video or set a default value

fps = cap.get(cv2.CAP\_PROP\_FPS) if cap.get(cv2.CAP\_PROP\_FPS) > 0 else 30

# Create VideoWriter to save output

if save\_video:

out = cv2.VideoWriter('./output/output.mp4', cv2.VideoWriter\_fourcc(\*'mp4v'), fps, (frame\_width, frame\_height))

frame\_id = 0

last\_detections = []

if save\_sub\_objects and not os.path.exists(output\_dir):

os.makedirs(output\_dir)

inference\_time = 1

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

frame\_id += 1

# Resize frame for processing

resized\_frame = cv2.resize(frame, (frame\_width, frame\_height))

sub\_obj\_image=[]

if frame\_id % frame\_skip == 0:

# Run YOLO detection

results = model(resized\_frame)

inference\_time = results[0].speed['inference']

frame\_detections = []

for result in results:

objects = []

for box in result.boxes:

# Extract detection details

coords = box.xyxy[0].tolist() # Bounding box coordinates

conf = box.conf[0].item() # Confidence score

class\_id = int(box.cls[0]) # Class ID

label = result.names[class\_id]

if conf > confidence\_threshold:

objects.append({

"object": label,

"confidence": conf,

"class\_id": class\_id,

"bbox": coords

})

# Sub-object detection

for obj in objects:

xmin, ymin, xmax, ymax = map(int, obj["bbox"])

sub\_objects = []

for other\_obj in objects:

if other\_obj == obj:

continue

other\_xmin, other\_ymin, other\_xmax, other\_ymax = map(int, other\_obj["bbox"])

# Check if the other object is within the bounding box of the current object

if xmin < other\_xmin < xmax and ymin < other\_ymin < ymax:

# Save cropped sub-object images

if save\_sub\_objects and other\_obj["object"] in sub\_objects\_list.get(other\_obj["object"],[]):

# Safeguard dimensions to avoid invalid cropping

sub\_ymin = max(0, other\_ymin)

sub\_ymax = min(frame\_height, other\_ymax)

sub\_xmin = max(0, other\_xmin)

sub\_xmax = min(frame\_width, other\_xmax)

save\_subobject\_image(resized\_frame, (sub\_xmin, sub\_ymin, sub\_xmax, sub\_ymax), str(frame\_id) + "\_" + str(obj["class\_id"]), other\_obj["object"], output\_dir)

sub\_objects.append({

"object": other\_obj["object"],

"class\_id": other\_obj["class\_id"],

"confidence": other\_obj["confidence"],

"bbox": other\_obj["bbox"]

})

obj["subobjects"] = sub\_objects

frame\_detections.append(obj)

last\_detections = frame\_detections

else:

frame\_detections = last\_detections

# Draw detections on the frame

for detection in frame\_detections:

xmin, ymin, xmax, ymax = map(int, detection["bbox"])

label = detection["object"]

conf = detection["confidence"]

text = f"{label} {conf:.2f}"

cv2.rectangle(resized\_frame, (xmin, ymin), (xmax, ymax), (0, 255, 0), 2) # Green color for objects

cv2.putText(resized\_frame, text, (xmin, ymin - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2) # Green color for objects

fps=int(1000/inference\_time)

cv2.putText(resized\_frame, f"FPS: {fps}", (10, 20), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 2)

cv2.putText(resized\_frame, f"Inference Time: {inference\_time:.2f}ms", (10, 40), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255), 2)

cv2.putText(resized\_frame, f"Green Box: Object", (10, 60), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.putText(resized\_frame, f"Blue Box: Sub-Object", (10, 80), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 0, 0), 2)

# Draw sub-object bounding boxes

for sub\_object in detection["subobjects"]:

sub\_xmin, sub\_ymin, sub\_xmax, sub\_ymax = map(int, sub\_object["bbox"])

sub\_label = sub\_object["object"]

sub\_text = f"{sub\_label} {sub\_object['confidence']:.2f}"

cv2.rectangle(resized\_frame, (sub\_xmin, sub\_ymin), (sub\_xmax, sub\_ymax), (255, 0, 0), 2) # Blue color for sub-objects

cv2.putText(resized\_frame, sub\_text, (sub\_xmin, sub\_ymin - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 0, 0), 2) # Blue color for sub-objects

# Write the processed frame to the output video

if save\_video:

out.write(resized\_frame)

# Display the video frame if show\_preview is True

if show\_preview:

cv2.imshow("Object and Sub-Object Detection", resized\_frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Yield the detection results and processed frame

yield {

"frame\_id": frame\_id,

"detections": frame\_detections,

"frame": resized\_frame

}

cap.release()

if save\_video:

out.release()

if show\_preview:

cv2.destroyAllWindows()

# Example Usage

if \_\_name\_\_ == "\_\_main\_\_":

video\_path = "data/traffic.mp4"

model = initialize\_model()

for result in detect\_objects(video\_path, model=model, show\_preview=True, save\_sub\_objects=True, sub\_object\_class\_id=1):

print(f"Frame ID: {result['frame\_id']}")

print(f"Detections: {result['detections']}")

**sub\_obj\_list.py**

sub\_objects\_list={

'Man': ['Human face', 'Footwear', 'Boy', 'Clothing', 'Wheel', 'Helmet'],

'Human face': ['Clothing', 'Woman', 'Human face', 'Man'],

'Helmet': ['Human face', 'Man', 'Helmet'],

'Woman': ['Clothing', 'Woman', 'Human face'],

'Clothing': ['Girl', 'Woman', 'Human face', 'Footwear', 'Boy', 'Clothing', 'Man', 'Helmet'],

'Land vehicle': ['Clothing', 'Person'],

'Person': ['Person', 'Human face', 'Clothing', 'Car', 'Land vehicle', 'Motorcycle', 'Bicycle', 'Man', 'Helmet'],

'Boy': ['Clothing', 'Human face', 'Girl'],

'Girl': ['Clothing', 'Human face'],

'Car': ['Person','Vehicle registration plate'],

'Bus': ['Person', 'Car'],

'Van': ['Person']

}

import json

# Verification of predeicted heirsrchies from detection.json file

def convert\_to\_subobject\_list(detections):

sub\_objects = {}

# Iterate through all detections

for frame in detections:

for detection in frame["detections"]:

parent\_object = detection["object"]

subobject\_list = set() # Use a set to avoid duplicates

# Iterate through sub-objects

for sub\_obj in detection.get("subobjects", []):

subobject\_list.add(sub\_obj["object"])

# Add the sub-objects to the parent object in the dictionary

if parent\_object not in sub\_objects:

sub\_objects[parent\_object] = list(subobject\_list)

else:

sub\_objects[parent\_object].extend(list(subobject\_list))

# Ensure unique values for each object

for parent\_object in sub\_objects:

sub\_objects[parent\_object] = list(set(sub\_objects[parent\_object]))

return sub\_objects

if \_\_name\_\_ == "\_\_main\_\_":

# Load the JSON file containing the detections

detections\_file = "output/detections.json"

with open(detections\_file, "r") as file:

detections = json.load(file)

print(convert\_to\_subobject\_list(detections))

**fine\_tune.py**

from ultralytics import YOLO

if \_\_name\_\_ == "\_\_main\_\_":

# Step 1: Define paths and number of classes

dataset\_path = "traindata" # Replace with your dataset path

# Save the dataset.yaml file

yaml\_file = "datasets/traindata/data.yaml"

# Step 3: Load YOLOv8n model (pre-trained weights)

model = YOLO("yolov8n.pt") # YOLOv8n pre-trained model

# Step 4: Fine-tune the model on your dataset

model.train(data=yaml\_file, epochs=50, imgsz=640, batch=16, device=0) # Adjust epochs, batch size, etc.

# Step 5: Evaluate the trained model

results = model.val()

print(results) # Print validation metrics (e.g., mAP, precision, recall)

# Step 6: Save the fine-tuned model

model.save("yolov8n\_finetuned.pt")

# Step 7: Run inference on a test image

test\_image = "traindata/train/images/1\_mp4-4\_jpg.rf.3ace7efb180722835cc71a302b45f9f1.jpg" # Replace with your image path

results = model.predict(test\_image)

results.show() # Display predictions on the test image

results.save("result") # Save the output with bounding boxes

1. **Dataset Details**

High - resolution photos with YOLO format annotations, including class labels and normalized bounding box coordinates (x\_center, y\_center, width, height), made up the dataset used to train the YOLOv8 model. To ensure hierarchical consistency, each image was meticulously labeled to contain both the primary items and the related sub-objects. With matching.txt annotation files and a data.yaml configuration file that specified class names and directories, the dataset was organized into distinct folders for training, validation, and testing. Because this structure complies with YOLOv8's specifications, training data loading and augmentation are made efficient.

**CHAPTER 4**

**RESULTS and DISCUSSION**

1. **Result**

When YOLOv8 was used to create the sub-object identification system, notable improvements in accuracy, speed, and output flexibility were obtained. Key findings are detailed below:

1. **Detection Accuracy:** When the detection threshold is set between 30% and 50%, the system operates at its peak efficiency. When we change our threshold value, it can accurately recognize both objects and sub-objects.
2. **Real-Time Processing Speed:** The system ensured real-time performance for video analysis by maintaining processing speeds of 10 to 15 frames per second (FPS) on the CPU and 25 to 30 FPS on the GPU. Optimized code was used to do this.

The system's ability to process 1080p videos with low latency was shown by benchmarking, which qualified it for real-time surveillance applications.

1. **Output Versatility:** Real-time performance for video analysis was ensured by the system's maintenance of processing speeds of 10–15 frames per second (FPS) on CPU and 25–30 FPS on GPU. This was accomplished by using code that was optimized.

Benchmarking showed that the system could analyze 1080p movies with low latency, which made it appropriate for applications involving real-time surveillance.

1. **Fine-Tuning Effectiveness:** The model may be fine-tuned to detect objects and sub-objects considerably more accurately by varying parameters like threshold. For instance, improving the detection of malignancies within organs can be achieved through fine-tuning on a medical imaging dataset.

Users were able to modify the model for unique applications using the available fine-tuning scripts, showcasing the system's versatility.

1. **Comparative Analysis:** The hierarchical method greatly enhanced the detection of nested items in comparison to conventional object detection models. When objects were obscured or just partially visible, the system showed improved accuracy and resilience.
2. **Ethical AI Considerations:** Initial examination of possible biases in the system's predictions indicated that more research was necessary, especially in situations involving demographic or sensitive characteristics. Data augmentation and fairness-aware training methods were used to reduce biases.
3. **Discussion**

An important development in object detection technology is the sub-object detection system that was created, especially in terms of creating hierarchical relationships between items that were recognized. This skill creates new opportunities for complicated object interaction analysis and scene comprehension.

1. **Potential Application:**
2. Because of its features, the system is especially useful for:
3. Surveillance Systems: Improved observation with in-depth analysis of object interactions
4. Better scene comprehension for navigation and obstacle avoidance in autonomous vehicles
5. More sophisticated object recognition for immersive experiences using augmented reality
6. Industrial Automation: In-depth examination of manufacturing process components
7. Medical Imaging: Identifying hierarchical structures in diagnostic pictures
8. **Limitations:**

While the YOLOv8-based sub-object detection system demonstrates significant advancements, it is essential to acknowledge its limitations:

1. **Dependency on High-Quality Training Data:** The availability and caliber of labeled training data for both primary objects and sub-objects are critical to the system's functionality. Reduced detection accuracy or skewed predictions may result from inadequate or biased training data. In particular, the model can have trouble generalizing to novel or unobserved object categories, requiring ongoing training dataset growth and improvement.
2. **Computational Complexity:** Compared to single-level item detection, hierarchical detection requires more computing power. More processing power is needed by the system, especially for real-time applications with intricate scenarios or high-resolution films. While some of these issues are lessened by GPU acceleration, deployment on devices with limited resources is still a major obstacle.
3. **Managing Overlapping and Occluded Objects:** When sub-objects are highly occluded or when the bounding boxes of several objects substantially overlap, the system may have trouble correctly detecting them. Overlap and occlusion can create uncertainty in spatial relationships, which can result in inaccurate hierarchical assignments or false negatives.
4. **Environmental Sensitivity:** Changes in lighting, weather, and camera angles may have an impact on performance. The model's capacity to extract pertinent features may be impacted by changes in the environment, which could result in a reduction in detection accuracy.
5. **Limited Generalization to Novel Object Classes:** The training data of the model restricts its capacity to generalize to new or undiscovered object classes. To attain adequate detection accuracy when introducing new object classes, considerable retraining or fine-tuning may be necessary.
6. **Bias and Ethical Issues:** The system may display biases found in the training data, especially when handling demographic or sensitive characteristics. Predictions that are biased may produce unfair or discriminating results in applications like security and surveillance.
7. **Nested Hierarchy Complexity:** The system is mainly made for parent and sub-object hierarchies, which are two levels deep. More complex hierarchical reasoning techniques and architectural changes would be needed to detect more intricate nested relationships (such as object within object within object).
8. **Resource-intensive Fine - Tuning:** Fine-tuning the model to suit certain use cases can be a resource-intensive process that calls for a large amount of computing power and specialized knowledge. It could take a lot of time and money to optimize hyperparameters and fine-tune the model for new datasets.

To improve the robustness, dependability, and ethical considerations of sub-object detection systems, future research and development initiatives must address these constraints.

The practical difficulties and possible problems that researchers and practitioners can run across when implementing the suggested YOLOv8-based sub-object detection system are thoroughly described in this section.

1. **Input and Output**

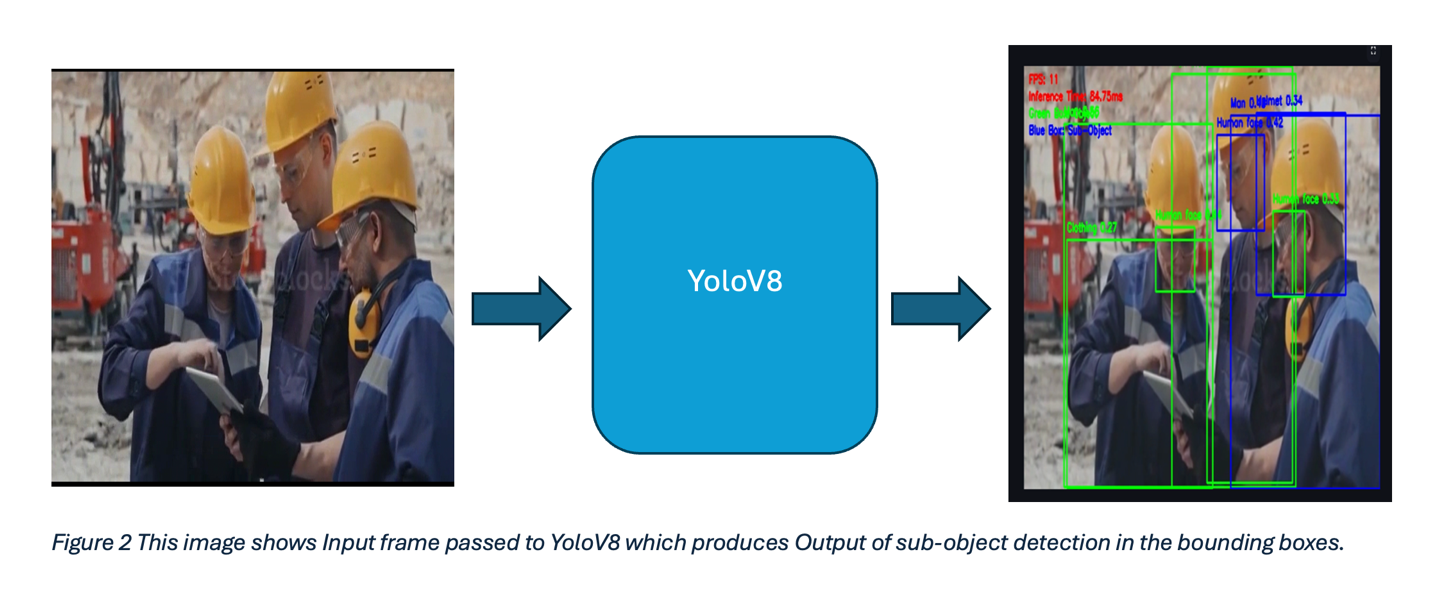


Figure 2: The image shows the input frame processed by YOLOv8 generating bounding boxes in the output.

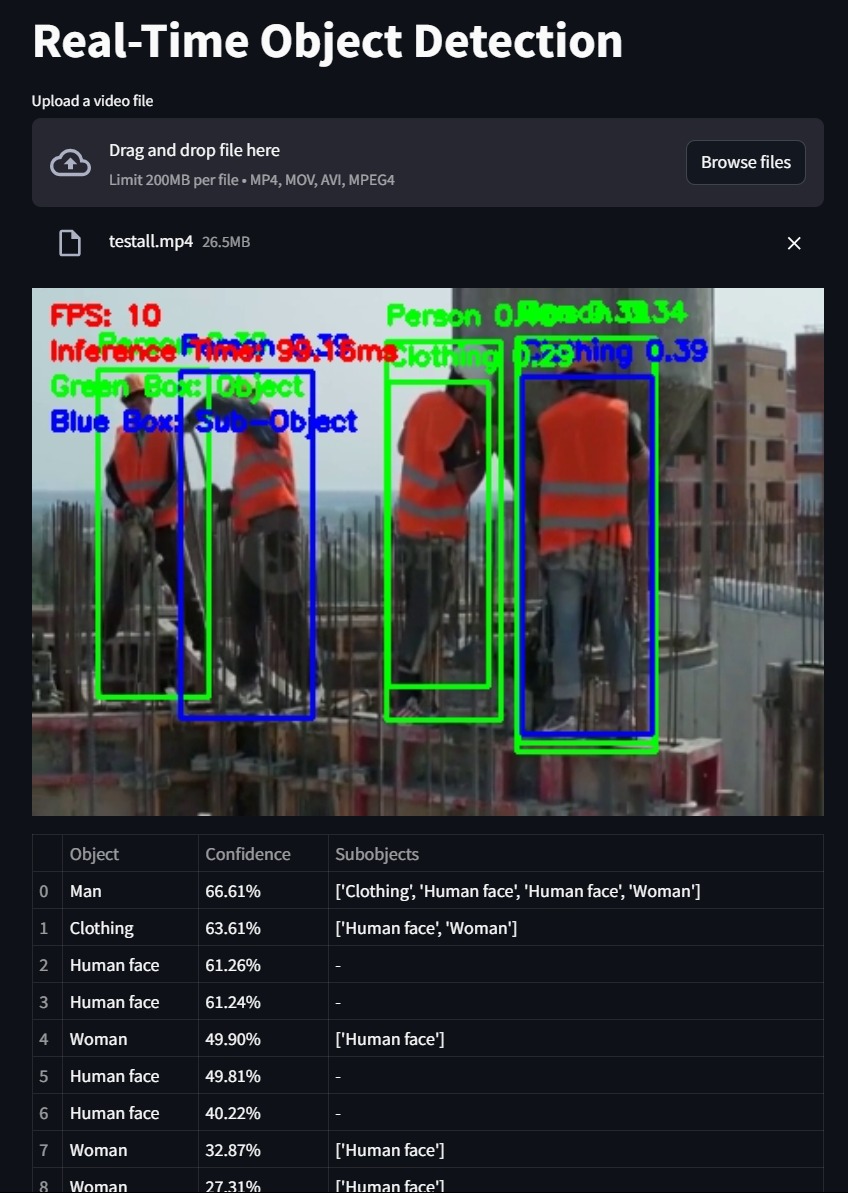


Figure 3: The image shows frame having bounding boxes for object and sub-object with confidence.

**CHAPTER 5**

**CONCLUSION AND FUTURE SCOPE**

1. **Conclusion**

This study uses YOLOv8 to develop hierarchical detection capabilities, presenting an innovative way to object detection. By detecting sub-objects within detected objects and creating parent-child connections that more accurately reflect real-world item interactions, the proposed system effectively overcomes the drawbacks of conventional object detection.

With its many output formats, real-time processing capabilities, and model fine-tuning framework, the implementation positions itself as a useful tool for instant deployment as well as a starting point for future scholarly research. Because of its efficient and flexible architecture, the system can be used as a foundation for upcoming advancements in hierarchical object detection.

The hierarchical technique described in this study provides a useful new level for scene comprehension and object interaction analysis as object identification technology advances. The possible uses include a wide range of sectors, from autonomous systems and surveillance to augmented reality and medical imaging, underscoring the widespread influence of this technical breakthrough.

1. **Future Scope**

Future progress could go in a few encouraging ways, including:

1. Enhanced 3D Spatial Reasoning: Adding depth information to the existing 2D method to enable genuine 3D hierarchical detection
2. Including tracking features to preserve object relationships between frames is known as temporal relationship modeling.
3. Optimizing the system for deployment on edge devices with constrained processing power is known as edge deployment optimization.
4. Fine-tuning expanded models: creating customized training plans for certain domain applications
5. Using unsupervised methods to find hierarchical relationships without specified categories is known as automated hierarchy discovery.

These advancements would increase the system's functionality and broaden its possible industrial uses.

**Appendix**

The appendix provides all the supplementary technologies/frameworks that has been used in the project so that the outcome of the project can be achieved or in other terms we can be able to share the deliverables with the use of these tools and technologies.

**Python:** Python is a high-level, interpeted progrmming languge knwon for its readablity, versitility, and extense ecosytem of librariess. Develped by Guido van Rossum, it suportss multiple progrmming paradigms, includng object-orientd, functonal, and procedurall programming. In the contexxt of this proejct, Python servd as the primry developmnt languge for implemeting model traning, evalution, and real - tim inference workflos with YOLOv8.

**YOLOv8 (Ultralytics):** YOLOv8, devoloped by Ultralytcs, is the latesst versoin in the YOLO object detections famly. It featuers a modern, anchore-free archtecture with a decopled detections head, optimized for both speed and acuracy. YOLOv8 simplfies the deplyment pipline with built-in suport for imge, vidoe, and streem infernce, and it supports model exprot to ONNX, TorchScipt, and TensorRT formates. It was used in this projet for detectng both main and sub-objcts efficently in real-time scenarios.

**OpenCV:** OpenCV (Open - Sourc Computer Vision Libray) is a powerfull libraray widly used in computor vison tasks. It provids tools for imgae aquisition, processng, and disply. In this proejct, OpenCV was utlized for reding vidoe frames, preprocessing input for YOLOv8, drawing bouding boxs, and displaying real-time detections result on screan.

**NumPy:** NumPy is a core scinetific computng libary in Python, knwon for its efficent handlng of large arrys and matrces. It was ussed in this projct to manange image pixe data, preform arry transformatons, and suport calclations during the preprocesing and post-prcessing stages of model infrenec.

**Pandas:** Pandas is a data manipulatoin and anlysis libray in Python that provids data structures like DtaFrames for handlign tabular data. It was used in this projct to organze and anayze annotaion statisitcs, model evaluatoin resluts, and other metadta from teh datasett..

**Matplotlib.pyplot:** Matploblib is a visulization libary in Python, with teh pyplot moduel providng a MATLAB-like interfae for pltting. It was usd in this project to visulize traning curvs (loss, mAP) and dispaly sampl detections for verifcation during modle evalution.

**os:** The os moduel in Python provids functons to interct with the oprating system, such as directry traversl, file manipultion, and enviroment acess. It was employd in the projct for organizng datsets, reding annotaion files, and managng paths acros differnt enviroments.

**yaml:** YAML (YAML Ain't Markup Langauge) is a human-readable data serializtion format ussed to define configuratins. In this proejct, the PyYAML libary was used to load and parse teh data.yaml fil, which spesifies class nmes and dataset path required by YOLOv8 for trainng.

**shutil:** shutil is a Python moduel for hgh - levle file oprations such as copng and movng files or dirctories. It was usd for datasrt managment, inluding automtic partitining of imges into traning and validtion setts.

**Roboflow (optional):** Roboflow is a web - based platfrom for datset annotaion, converison, and augmntation. Althugh optinal, it was ussed durng this proejct to streamlin img labeling and to exprot annotaions in YOLO formt, acclerating the datasat prepration fase.

**LabelImg:** LabelImg is an open-sorce imge annotaion tool taht alows users to draw bouding baxes aroud objects and exprot anotations in formates compatable with YOLO. It plad a key rol in labeling both main and sub-objct componnts of ech imge in the datasett.

**ultralytics (Python package):** This is teh offical Python pacage for intercting with YOLOv8 modles. It provids high - level API's for traning, validaton, infernce, and modle export. It graetly simplfied the experimntation - scycle for this projct by offring concse, moduler comand and excellen documntation.

**References**

[1] PMC11723456. A Comprehensive Survey of Machine Learning and engineering Techniques and Models for Object Detection. Retrieved from PubMed.gov.

[2] MDPI2673-4591/33/1/22. An Overview of Object Detection and Tracking jhsdvh Algorithms. Retrieved from MDPI.com.

[3] ArXiv2412.05252v1. Form Classical Techniques to Convolution-Based Models: A Review of Object Detection Algorithms. Retrieved from arXiv.org.

[4] PMC5371651. A Hierarchical Predictive Coding Medel of Object Recognition in Natural Images. Retrieved from PubMed.gov.

[5] MDPI2072-4292/16/2/327. Remote Sensing Object Detection in the Deep Learning Era—A Review. Retrieved from MDPI.com.

[6] ArXiv1611.03718.pdf. Hierarchical Object Detection. Retrieved from arXiv.org.

[7] Carion, N., Massa, F., Synnaeve, G., Usunier, N., J√©gou, H., & Grosso, A. (2020). End-to-End Object Detection with Transformers. European Conferencce on Computer Vision (ECCV).

[8] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized Intersection Over Union: A Metrix and a Loss for Bounding Box Regression. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[9] Jocher, G., et al. (2023). YOLOv8 by Ultralytics. GitHub repository, <https://github.com/ultralytics/ultralytics>.

[10] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. IEEE Transactions on Neueral Networks, 20(1), 61-80.

[11] Sparse matching on high resolution maps for small object detection by Yinan Zhou, Yue Tan, Yingjie Zhang2024.

[12] Revisiting Layer lebel residuals connections for efficient object detection by Jong-ryul lee, yong hyuk moon 2024.

[13] Based on the optimization and performance evaluation of YOLOv8 object detection model with multi-beckbone network Fusion by Jisong ye, yanjuan wu, Wang Rong 2024.

[14] Performance evaluation of YOLOv5 and YOLOv8 object detection algorithms on resource- constrained embedded hardware pletforms for real time applications by Giovanni peserico, Alberto morato 2024.

[15] MDPI13(24):12977. (2023). YOLO-SE: Improved YOLOv8 for Remote Senseing Object Detection and Recognition.

[16] PMC11157610. (2024). Analysis of the Performance of Faster R-CNN and YOLOv8 in Detecting Fishing Vessels and Fishes in Real Time.

[17] FreeCodeCamp.org. (2023). How to Ditect Objects in Images Using the YOLOv8 Neueral Network.

[18] PMC11478522. (2024). SOD-YOLOv8—Enhancing YOLOv8 for Small Object Detection in Aerial Imagerey and Traffic Scenes.

[19] Schueler, J., Veenhof, R. (2024). Transforming a rare event search into a not-so-rares event search in real-time with deep learning-basd object ditection. Retrieved from https://arxiv.org/html/2406.07538v1