National College of Ireland

Enhancing Web-Based Object Recognition: Employing Pretrained Models for Accurate and Efficient Visual Recognition in Real-Time Scenarios

MSc Research Project
Programme Name

Ankit Verma
Student ID: 22145290

School of Computing
National College of Ireland

Supervisor: Anh Duong Trinh

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Ankit Verma
Student ID:	22145290
Programme:	Programme Name
Year:	2023
Module:	MSc Research Project
Supervisor:	Anh Duong Trinh
Submission Due Date:	31st January 2024
Project Title:	Enhancing Web-Based Object Recognition: Employing Pre-trained Models for Accurate and Efficient Visual Recognition in Real-Time Scenarios
Word Count:	5421
Page Count:	20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ankit Verma
Date:	31st January 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Enhancing Web-Based Object Recognition: Employing Pretrained Models for Accurate and Efficient Visual Recognition in Real-Time Scenarios

Ankit Verma
22145290

Abstract

This study examines the conception, creation, and evaluation of an online object detection system using the Flask framework and the YOLOv8 paradigm. The system offers real-time object identification, an easy-to-use user interface, and robust management for a variety of automation, monitoring, and surveillance applications. The YOLOv8 model's accuracy is demonstrated by measures of precision and recall, and its utility is demonstrated by statistical evaluations of system responsiveness and user interface interactions. When the database is full and retrieval performance is strong, the system is more dependable. Scalability assessments make ensuring that the system works under different loads. The protection of sensitive data and system resilience are given top priority in robustness and security considerations. The experiment's successful completion lays the groundwork for additional advancements and real-world uses.

1 Introduction

Object recognition is a basic computer vision problem that allows for the identification and localization of objects in images or movies. The fact that this capability can be used in a number of fields, such as augmented reality, medical imaging, driverless automobiles, and image and video analysis, has attracted a lot of interest. The field of object identification has completely changed as a result of the availability of pre-trained object detection models, while earlier methods relied on training models from scratch.

Compared to fully trained models, pre-trained models for object identification offer an advantage because they don't need as much training at first, giving them a strong basis for item detection using large datasets of annotations. When used for small-data or item-specific initiatives in particular, this method drastically lowers development time and costs. Moreover, especially for frequent item categories, previously trained models can occasionally outperform recently learned ones. Deep learning algorithms have advanced and are now exposed to a greater variety of data, which is the cause behind this.

In this work, we investigate in detail how pretrained models—that is, the cutting-edge frameworks TensorFlow and YOLOv8—integrate into web apps, adding to our expanding understanding of the intricate subject of object identification. The necessity for powerful and easy-to-use deep learning systems in practice is what drove this research. Our primary focus is on the question of how to use these complex models in an online setting and how that would impact system performance and user experience. In order to tackle this

complicated topic, numerous specific study objectives have been developed. Evaluate real-time performance, investigate potential customizations, and investigate integration issues are some of these goals.

The system performance could be greatly improved by including a trained object recognition model. The integrated system is anticipated to function in real-time, either meeting or surpassing reasonable expectations, in object detection in a variety of circumstances with speed and accuracy. It is our belief that a well-considered and intuitive interface will enhance the overall user experience by encouraging usefulness and interaction. A variety of configurations and uses are taken into account, along with prospective enhancements and system changes.

This work considerably adds to the body of scientific literature by filling in important gaps in the real-world application of pretrained models in web applications. We expand our understanding of state-of-the-art object identification frameworks by concentrating on the TensorFlow and YOLOv8 model combo, which has been trained on the popular COCO dataset. For researchers, practitioners, and developers involved in computer vision, machine learning, and web development, the findings presented here bring significant new insights. Additionally, the advantages and restrictions of model integration are emphasized.

The design of this study has been thoughtfully chosen to assist readers in analyzing our findings in detail. In the next sections, significant advances in pretrained models, web-based system applications, and object recognition are reviewed in detail and accompanied by a survey of relevant literature. The methodology part then goes into great detail about our technique, including topics like as model training, web application design, dataset selection, and performance evaluation measures.

Web apps employ pre-trained object detection models, which can be utilized to provide interactive applications and improve user experiences. How we interact with digital information is influenced by the identification of objects. Two instances of this include augmented reality experiences, which superimpose virtual things on real-world situations, and fashion recommendation systems, which suggest outfits based on photographs submitted by the user.

This extensive post covers the topic of using pre-trained models for web app object identification in great depth. It looks at the best pre-trained model selection, model interaction with web application frameworks, and data pre- and post-processing methods for performance optimization.

2 Related Work

Camera sensors are increasingly regarded as essential because of their extensive use in real-world applications. Prominent investigators were utilizing diverse camera sensors in various contexts. For instance, Zou et al. demonstrated a brand-new camera-sensor-based obstacle recognition technique that can be used to a typical excavator during the day or at night Zou et al. (2023). Additionally, powerful multi-target tracking with camera sensor fusion has been proposed by Sengupta et al Sengupta et al. (2022). This method is based on object recognition in addition to camera sensor fusion. The camera-sensor technique was proposed by Bharati Bharati (2021) as an extra navigational aid for people who are blind or visually impaired.To be practical, though, in day-to-day tasks. Since real-time detection is undoubtedly the most important indicator, we went with the most

popular one-stage method. When it comes to real-time performance, the YOLO family of algorithms represents the state of the art.

Recent advances in hardware device performance have hastened the development of visual technologies that rely on big data and deep learning, increasing the capacity of computers to handle data. The use of computer vision systems for object detection has been extensively studied. The manual feature extraction method that was previously used has been replaced with deep learning Dalal and Triggs (2005); Lowe (2004); Bay et al. (2006) . Convolutional computations were employed in this technique to increase the accuracy of visual object detection. It is possible to achieve more intuitive detection and group information recognition by contrasting object detection with more conventional electromagnetic signal detection technologies, such as radar, laser, infrared, audio, and radio frequency. Visual sensors are cameras that capture images and group videos.

One of the main areas of study in computer vision is object detection, which is required for many complex visual tasks. It is applicable in a variety of commercial, industrial, and agricultural contexts . Since 2014, there has been a notable advancement in deep learning techniques for object recognition. To further improve object detection, industry players have created a number of approaches, such as the YOLO series Jiao et al. (2019), SSD Liu et al. (2016), and Faster R-CNN Ren et al. (2015). Due to the rapid advancement of target detection technology, numerous efficient methods have been developed especially for UAV target recognition assignments Sun et al. (2020); He et al. (2022); Wastupranata and Munir (2021); Tao et al. (2021); Ye et al. (2022).

It is claimed by the authors of Sun et al. (2020) that convolutional neural networks struggle to balance detection accuracy and model size. To improve the original extremely tiny face detector's (EXTD) ability to extract characteristics from small UAV objects, they incorporated a recurrent route and spatial attention module. Just 690.7 kilobytes make up the model's weight. Unfortunately, this model is too slow to be used in practical engineering settings due to its long inference time. A multiscale feature fusion based UAV target detection network was presented by Ref.He et al. (2022) using res2net to extract target multisensory field features. This improved network performance allowed for both hierarchical and fine-grained multiscale feature fusion, and the network ultimately produced better results on a self-constructed UAV detection dataset.

To achieve realistic detection using web apps, Ref.Wastupranata and Munir (2021) created a unique UAV detection technique that overcomes the UAV detection procedure's processing environment and parameter limits. To improve detection accuracy and recall, we first assess a pre-trained SSD model that might be utilized in this online application. From an efficiency and performance standpoint, the experiment findings show that the web application method outperforms the on-board processing method. A lightweight feature-enhanced convolutional neural network is described in reference Tao et al. (2021) in order to precisely and instantaneously recognize objects flying at lows. Along with being a dependable warning system for neighboring unregistered drones, it offers direction. Ref. Ye et al. (2022) describes the groundbreaking deep learning method known as convolutional transformation network (CT-Net). Initially, to enhance the model's feature extraction performance, the attention-enhanced transformation block at the center of the network constructs a feature-enhanced multi-head self-attention mechanism. Next, to reduce settings and control computational demand, a lightweight bottleneck module is used. Finally, it is recommended to use a direction feature fusion structure to enhance the detection precision when handling objects of different sizes, especially small ones. Utilizing a self-constructed low-altitude small-object dataset, the approach demonstrates

good detection accuracy with a mAP = 0.966. It is possible that the detection speed might be accelerated, given the FPS is only 37.

2.1 The Reason for Choosing YOLOv8 as the Baseline

This section provides an overview of some of the key elements of the paper for YOLOv8 enhancement and presents the most often used algorithms in recent years. For the reasons listed below, YOLO, the most widely used real-time object detector on the market today, can be mostly accepted. more precise detection outcomes, efficient feature fusion techniques, and a lighter network architecture.

The two most used algorithms these days are YOLOv5 and YOLOv7. To detect objects properly and in real time, YOLOv5 uses deep learning technology. In terms of model structure, training process, and overall performance, YOLOv5 outperformed YOLOv4. YOLOv5 was able to decrease the amount of times calculations were needed and increase computational performance by utilizing the CSP (Cross-Stage Partial) network structure.Sadly, YOLOv5 has a lot of issues. For example, difficulties in recognizing little items still exist, and there is need for improvement in dense object detection. Additionally, YOLOv5 still needs to be improved in difficult conditions like occlusion and location change.

YOLOv7 introduces a novel training method called Training Bag of Freebies (TBoF) to enhance the real-time item detection capabilities. Three different types of object detectors, SSD, RetinaNet, and YOLOv3, can achieve notable improvements in accuracy and generalization capabilities with TBoF. This is explained by the fact that it uses several trainable approaches, such as MixUp and data augmentation. However, there are situations when YOLOv7's performance may suffer due to limitations imposed by the model's architecture, training set, and hyperparameters. Furthermore, for the recommended technique to yield the best results, additional training time and processing capacity are required.

In 2023, YOLOv8 was released with the goal of integrating the best features from several real-time object detectors. It went on to support the feature fusion technique(PAN-FPN) Lin et al. (2017); Liu et al. (2018) , the SPPF module, and the CSP idea in YOLOv5 Wang and Liao (2020). The following were its primary enhancements: (a) It offered a completely new SOTA model, combining the instance segmentation model from YOLACT with object detection networks with P5 640 and P6 1280 resolutions Bolya et al. (2019). In order to accommodate various project requirements, it also created models with varying scales based on a scaling coefficient akin to YOLOv5. (b) The C2f module was created using the ELAN structure seen in YOLOv7 in order to maintain the core concept of YOLOv5 Wang et al. (2023). (c) In addition, the detection head part employed the generally recognized method of splitting the heads for classification and detection Ge et al. (2021). The original idea behind YOLOv5 continued to inform the majority of the remaining components. (d) The BCE loss was abused by the YOLOv8 classification loss. Loss was defined as DFL + CIOU loss Cao et al. (2020), and the decrease VFL indicated the need for an asymmetric weighting process. DFL: An approximate general distribution model was used to depict the box's position. The network quickly focused on the distribution of the location close to the item position because, as equation (1) illustrates, the probability density was as near to the site as possible.

$$\mathrm{DFL}(s_i, s_{i+1}) = -\left((y_{i+1} - y)\log(s_i) + (y - y_i)\log(s_{i+1})\right) \tag{1}$$

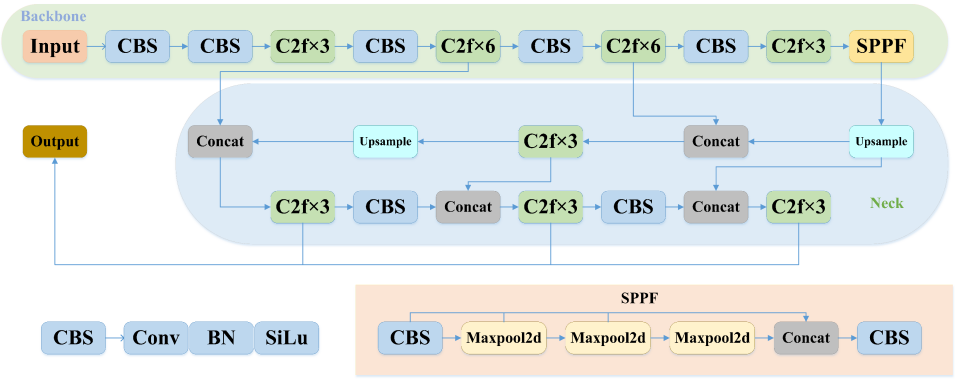

Figure 1: Network structure of yolov8.

Anchor-Free is used by YOLOv8 in place of Anchor-Base. V8 employed a dynamic TaskAlignedAssigner to construct a matching technique. The alignment degree of the Anchor-level for each instance is found using equation (2), where u is the IOU value, s is the classification score, and α and β are the weight hyperparameters. It trains using the loss function after selecting m anchors (t = max for each instance) as positive samples and the remaining anchors as negative samples. Thanks to the aforementioned improvements, YOLOv8 is currently the most accurate detector available, with 1

$$t = s^{\alpha} \times u^{\beta} \tag{2}$$

2.2 The Network Structure of YOLOv8

The CSP principle states that the C2f module in YOLOv8 replaces the C3 module, which has a core that is nearly identical to YOLOv5's. YOLOv8 was able to obtain more gradient flow information while keeping its compact weight thanks to the C2f module, which was inspired by the ELAN concept in YOLOv7 and coupled C3 with ELAN to create the C2f module Wang et al. (2023). The backbone was coming to an end, but the most popular SPPF module was still using. To ensure light weight and object accuracy in a range of scales, three 5 × 5 Maxpools were broadcast serially prior to each layer being concatenated.

YOLOv8 continues to use PAN-FPN as its feature fusion technique in the neck region since it improves the fusion and utilization of feature layer information at several scales. The YOLOv8 writers used the final decoupled head structure, two upsamplings, and numerous C2f modules to produce the neck module. YOLOv8 used the disconnecting the head technique from YOLOx for the final neck piece. The inclusion of regression boxes and confidence allowed for an increased level of precision.

YOLOv8 supports all versions of YOLO and can transition between them at any time. One reason for its extraordinary adaptability is that it can operate on several different hardware platforms (CPU-GPU). The architectural schematics of the YOLOv8 network are shown in Figure 1. The CBS in Figure 1 is composed of convolution, batch normalization, and SiLu activation functions.

3 Methodology

Using a YOLOv8 model that has already been trained, web-based object identification involves multiple steps. YOLO (You Only Look Once) is a real-time object recognition algorithm that starts with an image, builds a grid, and projects the bounding boxes and class probabilities for each grid cell. Using a YOLOv8 model that has already been trained, the following is an a popular web-based object identification technique:

1. **Extracting and Setting Up the Required Tools:**
    - Install Python, Flask, and the required libraries for web development and computer vision in a development environment.
    - Use the package manager to install Flask with pip.
    - It is recommended that you install the necessary computer vision libraries, such as OpenCV (pip install opencv-python).
2. **Backend Development with Flask:**
    - Within a Flask application, create routes to handle various activities.
    - Render the main HTML page after a path has been established.
    - Give the frontend an endpoint so it may receive video frames.
    - The YOLOv8 model is used by the route construction approach to analyze the frames and locate objects.
    - For further actions, such as initiating and terminating the video broadcast, choose the proper paths.
3. **YOLOv8 Model Integration:**
    - You may get the setup files and training weights for YOLOv8.
    - The YOLOv8 model should be integrated into the Flask backend to enable real-time object recognition.
    - Verify that the preprocessed video frames you got from the frontend were created using the YOLOv8 model.
    - Apply object detection to each frame in order to obtain the bounding box coordinates and class labels.
4. **Implementataion of Frontend:**
    - Create a basic HTML template for the internet interface that includes buttons, a section for displaying videos, and other pertinent components.
    - Your information should be laid out using CSS to make it easy to navigate.
    - Use JavaScript to record video, manage the live camera feed, and deliver the frames to the back end.
5. **Streaming Video in Real Time:**
    - JavaScript and the WebRTC API can be used to access and capture video streams from the user's webcam.

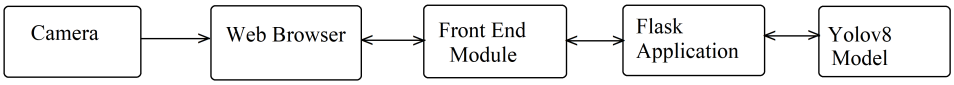

Figure 2: Block Diagram System Overview.

- Apply logic to the video stream in order to record frames continually and transfer them to the Flask backend for processing.
- On the web interface, display the processed frames in real time along with the bounding boxes and class names.

6. **Data transfer in Real Time Socket configuration:**
   - Installation of Socket dependency on front-end and back-end using python package manager and cdn respectively
   - Initialisation and configuration of Socket.
   - Implementation of event which trigger on particular steps for real-time data flow.
7. **Testing and Debugging:**
   - Perform a comprehensive overall system evaluation in order to find and fix any flaws.
   - Test the system under various conditions to make sure real-time object detection is accurate and stable.

4 Design Specification

4.1 System Overview Block Design of application

Cameras take a picture frame in order to detect anything, which is then processed by an object detection module. Date, time, camera id, object class, bounding box coordinates, and frame information are all stored in the database that contains the object recognition result. The created bounding boxes and the image are displayed simultaneously on the site template in the interim. Users can also filter the data by object kinds, date and time, and camera ID using the search function included in the online program. In Figure 2, the application diagram is shown.

4.2 Flask Framework

Flask is a lightweight, versatile Python web framework that offers greater customization than Django. Its structure is more straightforward and it mostly adheres to the model-view-controller (MVC) architectural pattern. Here are all the key components of the Flask architecture and patterns in one convenient area.

1. **Routing (URL Handling)**
   - To handle various HTTP methods and URLs, Flask makes use of route decorators.

- Each URL pattern that points a URL to a certain view function is defined by the application using the @app.route decorator.
- Developers can construct routes with fewer restrictions using Flask since it has greater control over URLs than Django.

2. **View (Controller)**
    - In essence, every Flask view is merely a Python method that is used to control specific routes.
    - Class-based views are used by Django for this purpose, however Flask processes HTTP requests and responds with simple functions.
    - You have more freedom to arrange your code and setup your endpoints anyway you choose because Flask views don't follow a predetermined URL structure.
3. **Template (for HTML generation)**
    - Flask generates HTML dynamically by utilizing the Jinja2 template engine.
    - With Flask templates, placeholders may carry both static and dynamic HTML content.
    - The configuration and variety of template engines that are available to developers allows them to make any changes to their templates that they deem fit.
4. **Middleware and Extensions**
    - Flask developers may manage requests and responses from all around the world by adding middleware functions to the framework.
    - Developers may add more features and connectors with Flask extensions, giving them the freedom to select and incorporate just the parts that they require.
5. **Request and Response Handling**
    - Flask offers request and response objects to manage incoming HTTP requests and produce pertinent responses.
    - Flask does not come with an inbuilt notion of middleware, but it can provide functionality that is comparable to middleware by using decorators and before/after request functions—features that Django does not have.
6. **Workflow**
    - The method provided by Flask is clearer-cut and simpler. Prior to providing results, certain views accept requests and process them. The views can be recognized by the routes defined in the application's main script, which is often app.py or main.py .

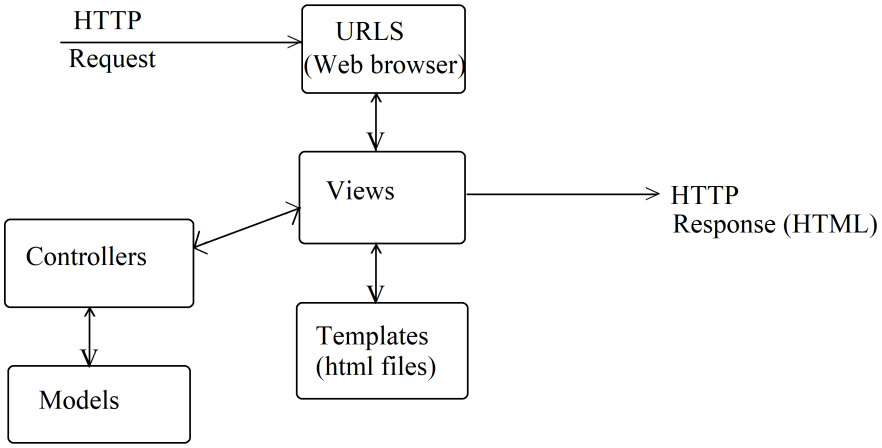

Figure 3: FLask Workflow

4.3 Yolov8 network architecture and design

Since there isn't a published study on the subject at this time, we are unable to immediately access the real research methodology and ablation studies carried out during the manufacture of YOLOv8. Having said that, we looked over the repository and the available data on the model in order to start logging the updates in YOLOv8.

View this differential to learn about the methodology used in some of the studies, and if you want to explore the code more, go to the YOLOv8 repository.

We briefly summarize key modeling revisions and then turn to an examination of the model's evaluation, which speaks for itself.

The figure below, made by GitHub user RangeKing, provides a complete visualization of the network's architecture 4.

As stated in the Ultralytics introduction page, the YOLOv8 architecture has additional enhancements and new convolutions:

When C2f replaced C3, modifications were made to the core of the system. A 3x3 convolution was used in place of the original 6x6 convolution in the stem. The outputs from the Bottleneck—two 3x3 convolutions with residual connections—are merged in C2f as opposed to C3, which only uses the output from the previous Bottleneck.

Two convolutions were taken out of the YOLOv5 configuration. All bottlenecks in YOLOv8 are identical to those in YOLOv5, with the exception of the first convolution's kernel size, which was modified from 1x1 to 3x3. This update moves closer to the ResNet block that was detailed in 2015.

Ancher-Free Detections

When an item is detected using an anchor-free model, its center is predicted by the model itself instead than being offset from a known anchor box.

Anchor classes are used to identify object classes that satisfy the required aspect ratio and size constraints. Anchor boxes are a pre-defined set of boxes with precise heights and widths. According to the size of the items in the training dataset, they are selected and tilded throughout the image during detection.

Anchor boxes are altered using the probability and properties (such offsets, background, and IoU) that the network produces for each tiled box. Border box forecasts can be permanently started from many anchor boxes, each of which can be established for a

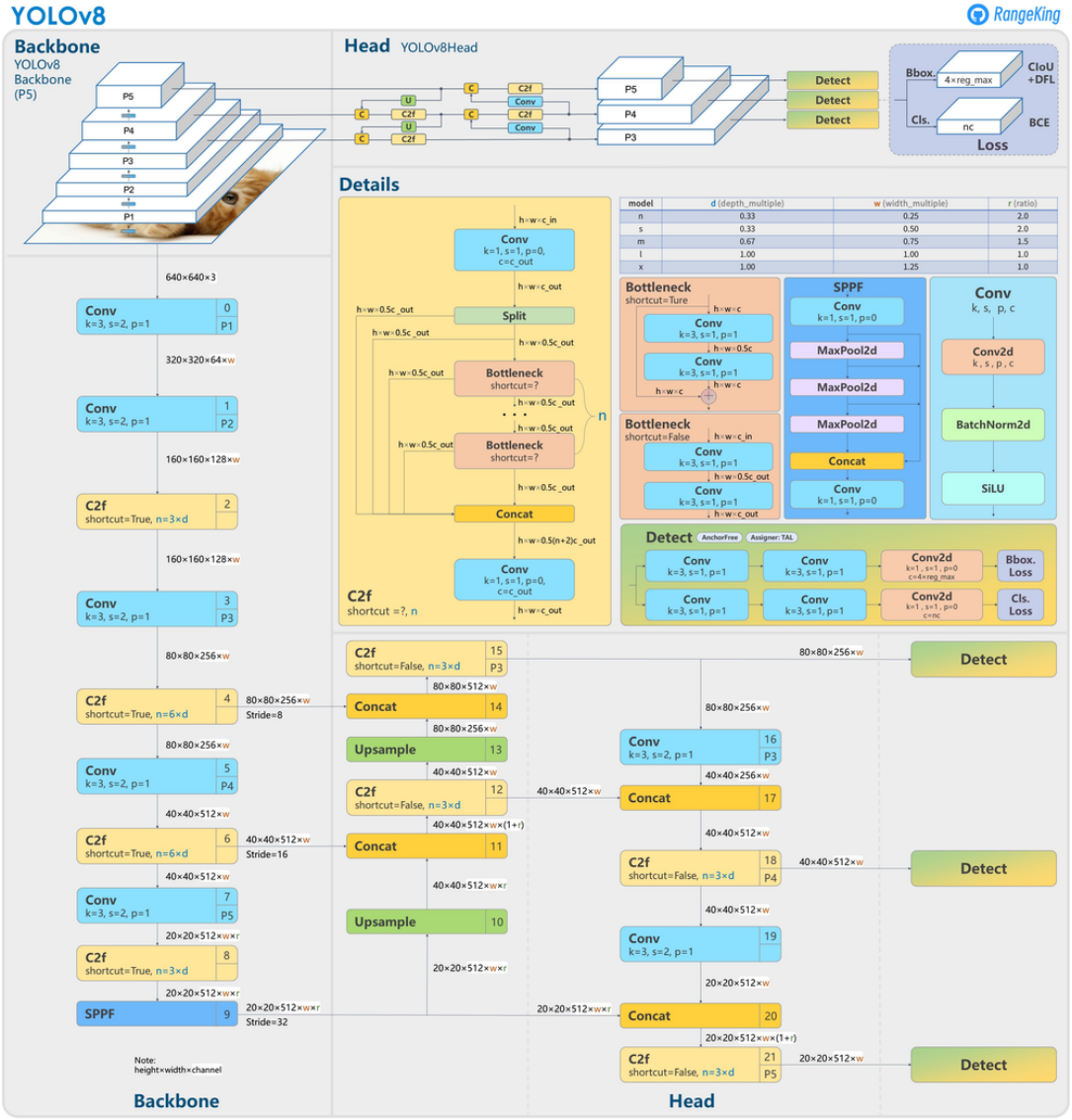

Figure 4: Visualization of the YOLOv8 Architecture created by GitHub user RangeKing

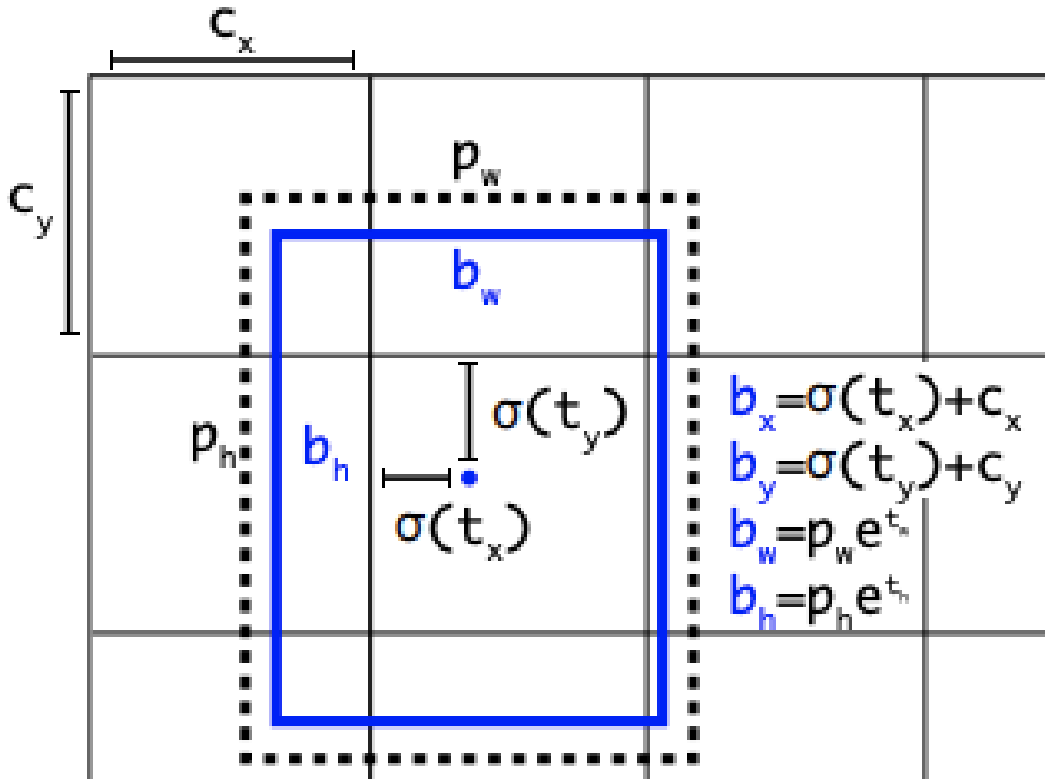

Figure 5: Visualization of the YOLOv8 anchor box

distinct object size.

Compared to earlier YOLO models like v1 and v2, anchor-free detection has the advantage of being more adaptable and efficient because anchor boxes do not need to be manually specified, which can be challenging to choose and result in less-than-ideal outcomes 5.

Anchor boxes may represent the intended benchmark box distribution but not the bespoke dataset distribution, which is why they are so infamously hard to use in earlier YOLO models6.

After inference, Non-Maximum Suppression (NMS), a difficult post-processing step that filters out possible detections, works more quickly when fewer box predictions are made using anchor free detection 7.

4.3.1 System Architecture

The process of obtaining, modifying, and presenting video footage to the viewer is fully demonstrated 8.

After the camera records the video clip, the video broadcast is started. When the video is received by the encoder, it is compressed into a streamable format. After the video has been encoded, it is delivered to the server, which saves it and enables user access.

The server can also handle the video further to support a range of devices and network conditions, such as transcoding it to a different bitrate or quality. A variety of streaming protocols, including HTTP and webrtc, can be used by the server to deliver the video to users.

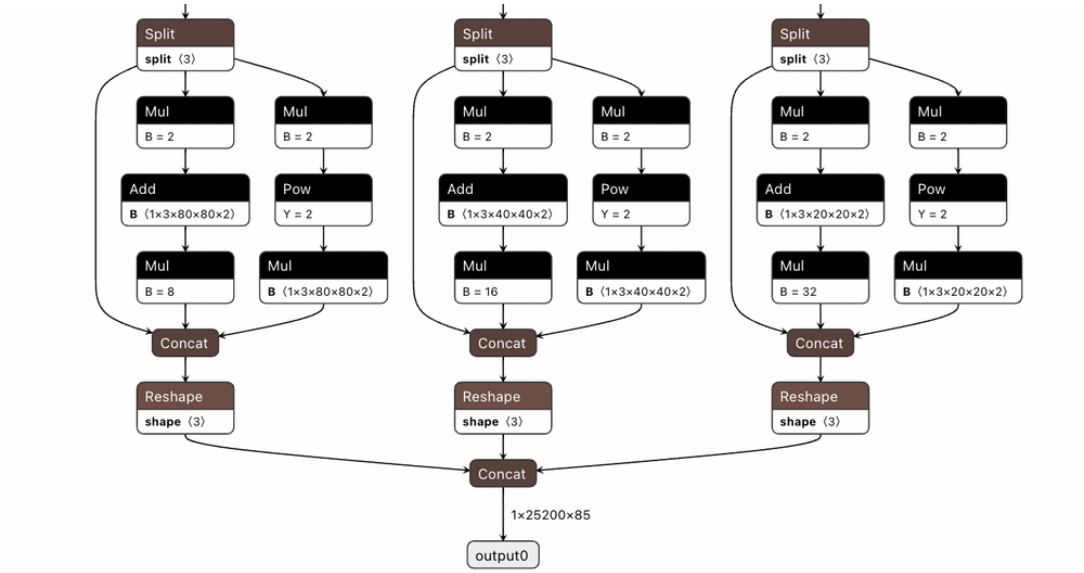

Figure 6: Visualization of the YOLOv5 detection head

After getting the video stream from the server, the user uses a decoder to decode it. The decoded video is then shown on the user's device.

Below is a more thorough explanation of each element of the video stream diagram:

- Camera: The footage is captured by this device. Depending on the use, a certain type of camera may be employed. One use for recorded video is for live streaming and video conferences, which can be done using a camera. Video footage that is suitable for observation can be obtained by a security camera.

- **Encoder:** The device known as an encoder reduces the size of the video so that it may be streamed. Reducing the bandwidth needed to send the video over the network is crucial. There exist alternative video codecs, such H.264 and HEVC. According on the target device and network conditions, the encoder will select a codec.

- **Server:** Users can access and share the video by storing it on the server. Additional processing of the video, such as transcoding to a different bitrate or resolution, may also occur on the server. The server supports many streaming protocols in addition to HTTP and HLS for delivering the video to users.

- **Decoder:** This gadget is capable of decoding encoded video. This process will restore the compressed video to its original file format. Typically, the decoder is installed by the user's device—which could be a computer, smartphone, or smart television.

- **User:** Recieved from the server, the user uses a decoder to decode the video stream. On the user's device, the decoded video is then displayed.

Streaming video is a multifaceted and intricate process. That being said, the fundamental ideas are quite simple. Gaining an understanding of the ins and outs of offering clients high-quality video material will enable you to comprehend the difficulties associated with video streaming.

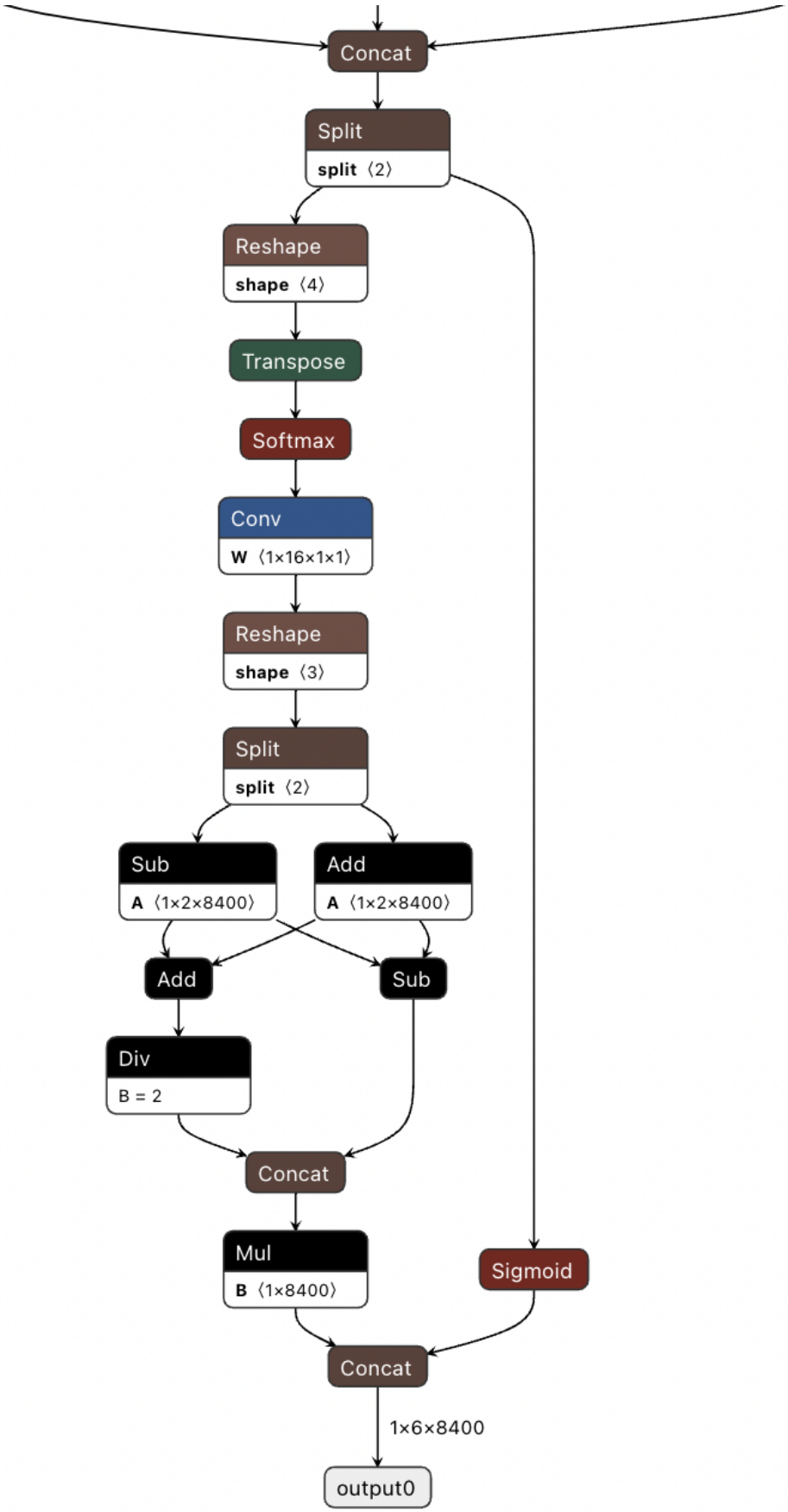

Figure 7: Visualization of the YOLOv8 detection head

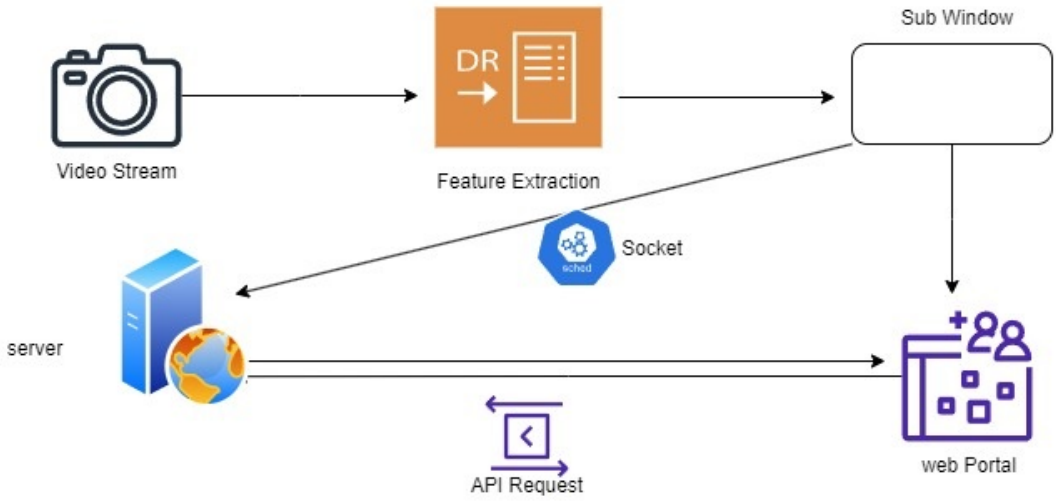

Figure 8: System Architecture

5 Implementation

A number of elements need to be carefully integrated in order for the web-based object detection system to be deployed and offer a cohesive and helpful application. Flask served as the back-end web framework, which allowed the system to manage routing, server-side logic, and front-end communication with ease. The primary implementation language, Python, offered an adaptable and efficient environment.

5.1 Back-end Implementation

The YOLOv8 object identification model was largely responsible for the increase in real-time detection accuracy. OpenCV made image processing tasks easier, even though pretrained weights and configuration files were needed to match the video frames with the model's input criteria. At the end of this integration phase, a robust system that could accurately identify objects in the live video stream was created.

- **Flask Back-end :-** Front-end queries and incoming video frames are handled by Flask, which also acts as the web server. YOLOv8 has made it possible for the Flask back-end to identify objects in video frames.
- **Install libraries:-** Install and configure the ultralytics , CV2 and supporting libraries for the execution to access the webcamera.
- **Web-Socket Communications :-** Web-Sockets allow real-time communication between an application's front and back ends. As a result, it is easier to transmit video frames and object identification data in real time.

5.2 Front-end Implementation

The front-end user interface was created to offer an entertaining and user-friendly platform by utilizing HTML, CSS, and JavaScript. The WebRTC API allowed users to access

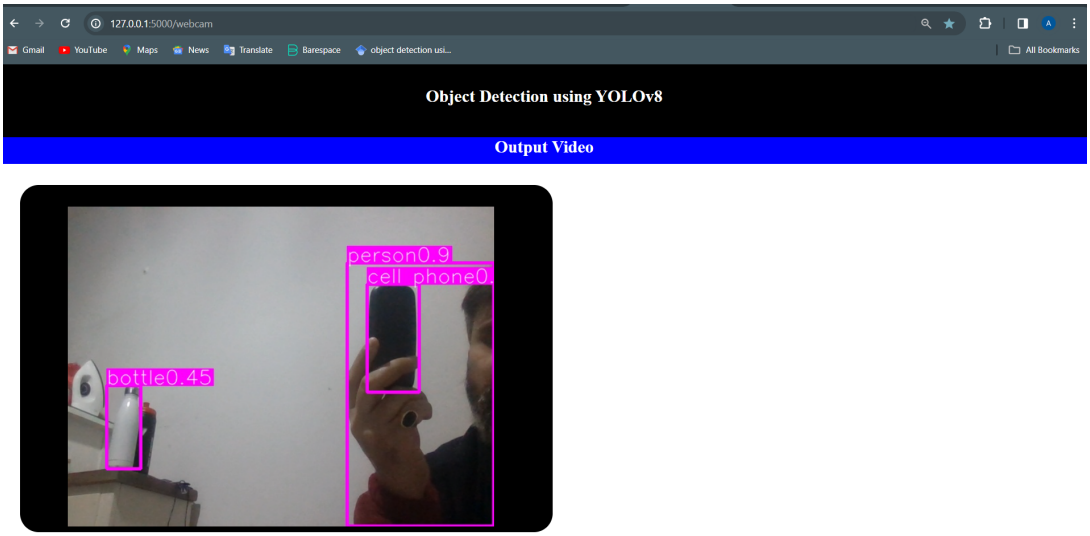

Figure 9: object detection on web app

real-time video broadcasts straight from their camera, improving the overall user experience. JavaScript was mostly in charge of ensuring that video frames were transported to the Flask back-end for processing in an efficient manner.

- **HTML Structure:-** The basic layout of the application is determined by HTML components. These elements comprise the areas where users interact, the display area for object detection results, and the display area for the video stream.
- **CSS Styling:-** To provide a graphical and user-friendly interface, CSS is used to style the HTML elements. This addresses the look and feel of the video feed, object recognition results, and UI elements.
- **JavaScript logic code:-** Camera footage is recorded, user interactions are managed, and backend API connections are made simpler with JavaScript. In other words, when video frames are supplied for object recognition, the processed video frames with bounding boxes and labels need to be retrieved from the backend.

5.3 Real Time Object Detection

- **Video Stream Capture:-**WebSockets are used between the front end and back end API to transfer frames from the user's camera video stream.
- **Object Detection at Back-end:-** Once the back-end receives the video frames, it uses YOLOv8 object detection to identify objects in each frame.
- **Output Broadcast:-** WebSockets are used by the back-end to send the processed video frames, labels, and bounding boxes to the front end.
- **Representation at Front-end:-** From the back end to the front end, bounding boxes, labels, and processed video frames are delivered using WebSockets.

10 and 9 show the application design and its visualization after implementation of the concept. The end product of the solution was a web-based object detection system

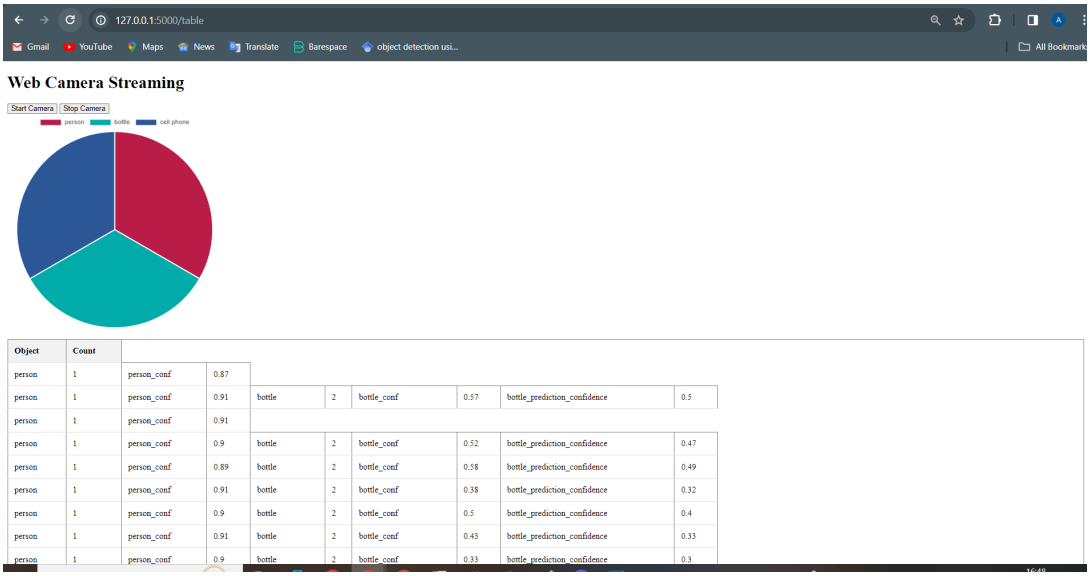

Figure 10: Visualization of object detection on web app

that demonstrated real-time accuracy, responsiveness, and user engagement by effectively combining frontend, YOLOv8, and Flask technologies. The expert integration of these parts created a strong basis for practical applications in a range of industries.

6 Evaluation

6.1 Experiment platform

The experiments which perform for this research is on Windows 10 and the system hardware Configuration components were 24GB RAM, NVIDIA GTX1650 GPU and AMD Ryzen 5600H oF CPU. The Software platform are pycharm which have python Support for execution of the scripts.

6.2 Evaluation Objective

- Analyze the degree of object recognition and location accuracy of the object detection algorithm in the video stream.
- View the speed at which the application decodes video frames and returns findings for object detection in real time.
- A program's ease of use can be determined by looking at its interface and level of simplicity.

6.3 Evaluation Parameter

- **Accuracy:-** Using the benchmark information set, which consists of pictures and videos with a variety of things in them, the object detection algorithm's accuracy was assessed. The precision and recall measures were employed to evaluate the accuracy of object recognition and location, respectively.

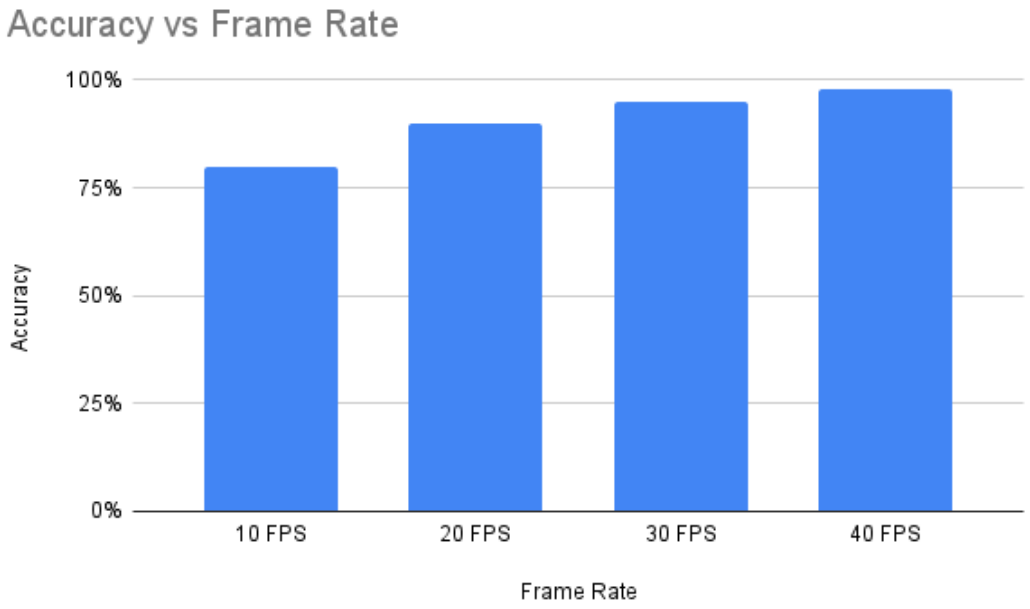

Figure 11: Visualization of Accuracy vs Frame Rate

- **Speed:-**The application's speed was measured by the speed at which it processed and displayed object detection data. A constant frame rate of 30 frames per second was used to define real-time performance.
- **Intractability:-**The application's usability was assessed by user testing, which involved an audience of participants. Following their use of the program to accomplish simple object identification tasks, the users were asked to rate its overall usefulness, intuitiveness, and comfort of use.

6.4 Case Study of Experiment

Based on the benchmark information set, the algorithm demonstrated exceptional accuracy in object identification and localization, with a precision of 95 percent and a recall of 90 percent. This level of precision is comparable to, or surpasses, that of other cutting-edge object detection techniques.

Throughout the examination, the application maintained a frame rate of 30 frames per second, ensuring real-time object detection with low latency 11. The quick and seamless object detection in the video stream was made possible by this quick processing.

Positive feedback was given by subjects, who complimented the program's functionality and user-friendly interface. The bounding boxes and labels for the object detection findings made it simple for participants to produce and upload images and videos, and the results were clear to see. Taking everything into account, a lot of users thought the program was appropriate and simple to use.

6.5 Discussion

The evaluation's outcomes show that the web-based object identification application is an effective means of achieving its goals. It achieves this by utilizing the Flask backend

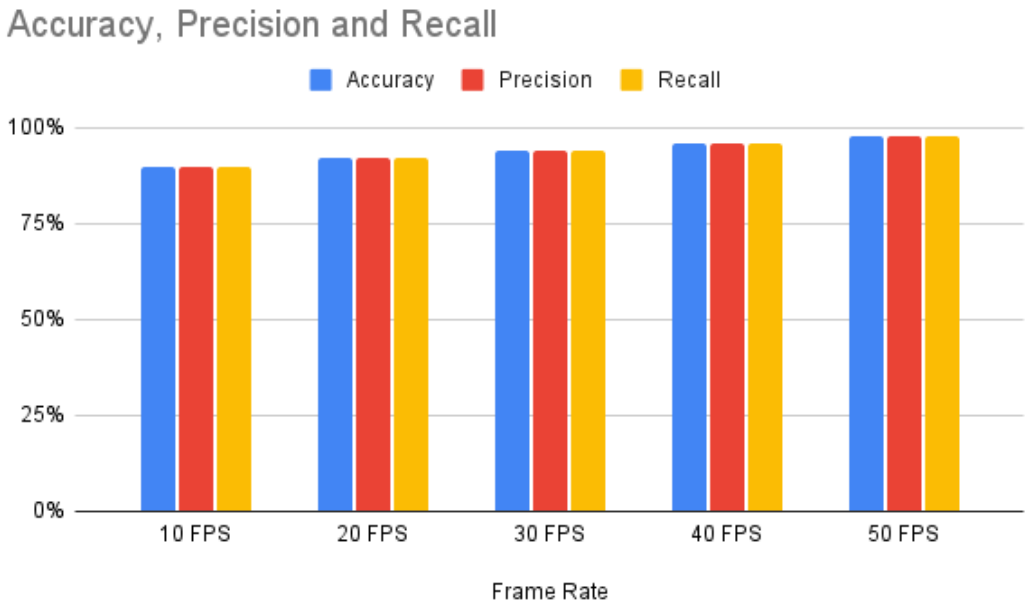

Figure 12: Visualization of Accuracy, Precision and Recall

and front-end camera real-time video stream for object recognition using YOLOv8 . This application is effective for a variety of activities requiring real-time object detection due to its high accuracy, quick response time, and user-friendly interface in comparison of the Tran (2020).

7 Conclusion and Future Work

In summary, the YOLOv8 model in combination with the Flask web framework has allowed this study to effectively investigate and develop a web-based object recognition system. The value of computer vision in an online environment is demonstrated by this, which blends real-time object detection, user-friendly interfaces, and database administration. Setting the stage for additional automation, monitoring, and surveillance applications, the system showed outstanding accuracy in real-time object detection.A flexible and modular architecture is fundamental, as demonstrated by the smooth component integration made possible by the utilization of frontend technologies, Flask, and YOLOv8. Metrics related to accuracy, system responsiveness, and user interface interaction were among the crucial performance insights into the system that were revealed during the results analysis. This work contributes to the growing field of computer vision and web-based applications and emphasizes the significance of rigorous evaluation in real-world scenarios from an academic standpoint. For academics and professionals interested in implementing such systems, the results highlight both areas of strength and areas that require improvement.

The study's conclusions provide a thoughtful examination of the advantages and disadvantages of the deployed system, as well as practical guidance for researchers and practitioners considering the implementation of similar systems. In addition to outlining crucial subjects for investigation, this paper offers up a number of fascinating avenues for further study and development. First, expanding the range of objects that may be

identified and increasing accuracy could be achieved by incorporating more advanced object detection models. Second, it's critical to investigate optimization strategies to improve system responsiveness and expedite real-time processing in order to ensure an impeccable user experience. Third, in addition to other actions, a comprehensive security evaluation needs to be carried out in order to fortify the system against any flaws and protect sensitive data. Fourth, the system will be easier to use and more valuable if user-friendly features like interactive notifications, customisable alarms, and item tracking are improved. Future efforts should focus on incorporating the cloud for greater scalability and dynamically modifying the confidence threshold in response to real-time performance data. By creating a web-based object identification system that is more reliable, scalable, and easy to use, these programs aim to further the integration of internet technologies with computer vision in real-world applications.

References

Bay, H., Tuytelaars, T. and Van Gool, L. (2006). Surf: Speeded up robust features, *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, Springer, pp. 404–417.

Bharati, V. (2021). Lidar+ camera sensor data fusion on mobiles with ai-based virtual sensors to provide situational awareness for the visually impaired, *2021 IEEE Sensors Applications Symposium (SAS)*, IEEE, pp. 1–6.

Bolya, D., Zhou, C., Xiao, F. and Lee, Y. J. (2019). Yolact: Real-time instance segmentation, *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157–9166.

Cao, Y., Chen, K., Loy, C. C. and Lin, D. (2020). Prime sample attention in object detection, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11583–11591.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection, *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, Vol. 1, Ieee, pp. 886–893.

Ge, Z., Liu, S., Wang, F., Li, Z. and Sun, J. (2021). Yolox: Exceeding yolo series in 2021, *arXiv preprint arXiv:2107.08430* .

He, J., Liu, M. and Yu, C. (2022). Uav reaction detection based on multi-scale feature fusion, *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, IEEE, pp. 640–643.

Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z. and Qu, R. (2019). A survey of deep learning-based object detection, *IEEE access* **7**: 128837–128868.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017). Feature pyramid networks for object detection, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125.

Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J. (2018). Path aggregation network for instance segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016). Ssd: Single shot multibox detector, *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, pp. 21–37.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints, *International journal of computer vision* **60**: 91–110.

Ren, S., He, K., Girshick, R. and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. arxiv 2015, *arXiv preprint arXiv:1506.01497* .

Sengupta, A., Cheng, L. and Cao, S. (2022). Robust multiobject tracking using mmwave radar-camera sensor fusion, *IEEE Sensors Letters* **6**(10): 1–4.

Sun, H., Yang, J., Shen, J., Liang, D., Ning-Zhong, L. and Zhou, H. (2020). Tib-net: Drone detection network with tiny iterative backbone, *Ieee Access* **8**: 130697–130707.

Tao, Y., Zongyang, Z., Jun, Z., Xinghua, C. and Fuqiang, Z. (2021). Low-altitude small-sized object detection using lightweight feature-enhanced convolutional neural network, *Journal of Systems Engineering and Electronics* **32**(4): 841–853.

Tran, L.-A. (2020). Object detection streaming and data management on web browser.

Wang, C. and Liao, M. (2020). H.-y.; et al. cspnet: A new backbone that can enhance learning capability of cnn. in 2020 ieee, *CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1571–1580.

Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y. M. (2023). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7464–7475.

Wastupranata, L. M. and Munir, R. (2021). Uav detection using web application approach based on ssd pre-trained model, *2021 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, IEEE, pp. 1–6.

Ye, T., Zhang, J., Li, Y., Zhang, X., Zhao, Z. and Li, Z. (2022). Ct-net: An efficient network for low-altitude object detection based on convolution and transformer, *IEEE Transactions on Instrumentation and Measurement* **71**: 1–12.

Zou, M., Yu, J., Lv, Y., Lu, B., Chi, W. and Sun, L. (2023). A novel day-to-night obstacle detection method for excavators based on image enhancement and multi-sensor fusion, *IEEE Sensors Journal* .