



UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

MASTER DEGREE IN ICT FOR INTERNET AND MULTIMEDIA

Title

Seeing with Sound: Object Detection and Localization by
YOLOv8 and Audio Feedback for Blind Individuals.

SUPERVISOR:

PROF.SSA FEDERICA BATTISTI

CO-SUPERVISOR:

PROF.SSA YEONGMI KIM

CANDIDATE:

ALI

TAVAKOLI YARAKI

2040500

ACADEMIC YEAR 2023-2024

Abstract

Object detection is a challenging Computer Vision (CV) application, particularly for assisting blind individuals. With the rapid advancement of Deep Learning (DL), algorithms like *Convolutional Neural Network* (CNN) have significantly improved video analysis and image understanding for this purpose. Blind individuals face substantial challenges when navigating indoor or outdoor environments, underscoring the pressing need for assistive technologies. In this thesis, a system has been developed to address this need, integrating the You Only Look Once (YOLO) object detection algorithm with audio guidance to aid blind users. The solution utilizes YOLOv8's State-Of-The-Art (SOTA) deep convolutional neural network architecture to detect objects in the user's environment, providing spatial information and counting processes through audio feedback. The system, equipped with a text-to-speech engine, converts all the information into verbal instructions, in some cases acting as a virtual assistant shape program. This context-aware feedback, available in multiple languages, has been optimized for webcams as a real-time scenario, images, and videos. The system has shown promising results, enhancing the autonomy and quality of life for blind users, a significant step towards addressing the challenges they face in daily environments.

Keywords: Object detection, blind people, audio feedback, spatial location, YOLO, computer vision, bounding box.

Acknowledgements

I express my deepest gratitude to my supervisors at the University of Padova and the Management Center Innsbruck (MCI).

I thank Professor. Federica Battisti for her kind and helpful support and feedback throughout this journey. I am equally grateful to Professor Yeongmi Kim for assigning me a title that perfectly fits my interests and guiding me toward this research direction.

I am incredibly thankful to my family, partner, and friends, whose unwavering support and encouragement have been my endless source of strength.

I also extend my sincere appreciation to the University of Padova for the comprehensive education and knowledge imparted to me and MCI for providing an impactful internship experience that demanded hard work and dedication.

Lastly, I would like to thank Cybathlon for inspiring me with the idea for this thesis and their motivating influence throughout the process.

Contents

Abstract	III
Acknowledgements	V
1 Introduction	1
1.1 Background and Motivation	1
1.2 Problem Statement	2
1.3 Objective of The Study	3
1.4 Deep learning for computer vision application	3
1.4.1 Convolutional Layer	4
1.4.2 Pooling Layer	4
1.4.3 Network training	4
1.4.4 Backpropagation	4
1.4.5 Fully Connected Layers	5
1.5 Object Detection	5
1.5.1 Single stage detection	5
1.5.2 Two stage detection	6
1.6 Object detection models performance evaluation metrics	6
1.6.1 Precision	7
1.6.2 Recall	7
1.6.3 F1-Score	7
1.6.4 Average Precision	7
1.7 YOLO (You Only Look Once)	8
1.7.1 Step-by-Step Insights into YOLO	9
1.7.2 Residual blocks	9
1.7.3 Bounding box regression	9
1.7.4 Intersection over Union	10

1.7.5	Non-Maximum suppression	11
2	Related Works	13
2.1	A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS	13
2.2	Real-Time Object Detection Using Yolo Algorithm for Blind People	14
2.3	An Real Time Object Detection Method for Visually Impaired Using Machine Learning	15
2.4	Object Detection for Blind People Using Yolov3	15
2.5	Understanding of Object Detection Based on CNN Family and YOLO	16
2.6	Real-Time Object Detection with YOLO	16
2.7	An Assistive Model for Visually Impaired People using YOLO and MTCNN	17
2.8	Object Detection System For The Blind With Voice guidance	17
2.9	Real Time Object Detection with Audio Feedback using Yolo vs. Yolov3	18
2.10	Real Time Object Detection	18
2.11	Object Detection Using Machine Learning for Visually Impaired People	19
2.12	Object Detection Featuring 3D Audio Localization for Microsoft HoloLens: A Deep Learning Based Sensor Substitution Approach for the Blind	19
2.13	Deep learning based object detection and surrounding environment description for visually impaired people	20
2.14	Object Detection with Voice Guidance to Assist Visually Impaired Using YOLOv7	20
2.15	Android Based Object Recognition for Visually Impaired	21
2.16	Object Detection and Recognition for a Pick and Place Robot	21
2.17	Assistive Technologies for Obstacle Detection and Identification	22
2.18	Absolute Distance Prediction Based on Deep Learning Object Detection and Monocular Depth Estimation Models	22
2.19	Existing Systems	23
2.19.1	Wearable Devices	23
2.19.2	Smartphone Applications	24
2.19.3	Robotics	24
2.19.4	Websites	24
3	Proposed Method	25
3.1	Tools and Libraries	25

3.2	Dataset and Data Augmentation	25
3.3	YOLOv8	26
3.4	Training	30
3.5	Phase 1	33
3.5.1	Object Detection	33
3.6	Phase 2	35
3.6.1	Object Counting	35
3.7	Phase 3	35
3.7.1	Object Distance	35
3.7.2	Object Spatial Location	36
3.8	Phase 4	38
3.8.1	Text to Speech	38
3.8.2	Generating the Audio Feedback	40
3.8.3	Delivering the Audio Feedback	40
3.8.4	Language Modifying	40
3.9	Speech Recognition	41
4	Results and Discussion	45
4.1	System Implementation	45
4.2	Experimental Setup	46
4.2.1	Framework	46
4.2.2	Dataset limitation	46
4.3	Performance Metrics and Evaluation Criteria	47
4.3.1	Accuracy	47
4.3.2	The Loss	49
4.3.3	Speed	50
4.4	Analyzing of Model Performance	50
4.4.1	Object Detection	50
4.4.2	Object's Spatial Location	54
4.5	Audio feedback analysing	55
4.6	A Prototype of Virtual Assistant	56
5	Limitations and Future Work	59
6	Conclusions	61

Acronyms	63
Bibliography	63

List of Figures

1.1	Object detection example	5
1.2	Object detection stages	6
1.3	YOLO architecture[1]	8
1.4	Grid cells	9
1.5	Bounding box regression	10
1.6	Intersection Over Union (IoU)	10
1.7	Non Maximum Suppression (NMS)	11
2.1	A Timeline of YOLO versions[2]	14
2.2	YOLO versions architecture[2]	14
2.3	Accuracy comparison[3]	15
2.4	User interaction Through system[4]	19
2.5	Design diagram[5]	21
2.6	Visual process of whole network[6]	23
2.7	Estimated distance vs. absolute distance[6]	23
2.8	Orcam MyEye device	24
3.1	Sample of data augmentation	26
3.2	Kernel operation	28
3.3	YOLOv8 Architecture	29
3.4	Accuracy and speed of YOLOv8[7]	30
3.5	Training configurations.	31
3.6	Training batch with bounding boxes and labels	32
3.7	System design	33
3.8	Object detection script	34
3.9	Code snippets for counting objects	35
3.10	Image dimensions.	36

3.11	Location category based on coordinates	37
3.12	initializing text-to-speech engine	38
3.13	Code Snippets Through a Loop	39
3.14	Feedback Generation	40
3.15	Delivering the Audio Feedback	40
3.16	Language Modifying Script	41
3.17	Speech recognition code snippets	42
4.1	Class labels	46
4.2	Training log	47
4.3	Confusion matrix	48
4.4	F1-Score and PR-Curve	48
4.5	Training result	49
4.6	Validation label and Validation Prediction	50
4.7	Detection result	51
4.8	Model first performance	52
4.9	Adjusting parameters	52
4.10	Evaluation detection accuracy	53
4.11	Real time example	54
4.12	Practical result	54
4.13	Printing audio on console	55
4.14	Italian Language	56
4.15	Speech recognition practical result	57
4.16	Speech recognition another practical result	57

List of Tables

3.1	Categorizing the location.	38
3.2	Visualization of the 3x3 grid	38

Chapter 1

Introduction

Living a healthy life can seem simple, but it's often profoundly challenging for those with disabilities. Genuinely understanding the daily struggles and limitations faced by individuals who can't see, hear, or touch as we do requires deep empathy.

1.1 Background and Motivation

My passion for technology led me to explore ways to help blind individuals interact more fully with their surroundings, see the world, or at least perceive it in a manner.

This passion came to life during my master's internship at the MCI in Austria, where I discovered CYBATHLON. This company organizes competitions for people with disabilities, focusing on daily living tasks. These events pushed participants to use advanced technology to overcome obstacles, showcasing how innovation can improve lives. My time with CYBATHLON sparked a strong interest in developing assistive technologies for the visually impaired. I saw the potential to create tools that empower blind individuals to handle everyday activities independently, such as shopping, walking, dressing, using public transport, or navigating their homes. Traditionally, these tasks often require help from guide dogs, canes, or other people, which, while beneficial, can limit autonomy.

I envisioned a solution that would let blind individuals manage these activities independently without constant reliance on external aids. This vision launched my journey to develop a technological platform that offers real-time, practical assistance, enabling blind individuals to "see" the world in new ways through innovative technology.

1.2 Problem Statement

According to World Health Organization (WHO) survey, almost 286 million people are visually impaired, which is separated from 39 million being blind and 246 million having low vision[8][9]. The ability to navigate and interact with the environment is essential to independent living, but it remains one of the most significant challenges these sorts of people face. Blind individuals severely restrict access to visual cues necessary for safe and efficient navigation, often making even familiar environments challenging to negotiate.

The development of assistive technologies has played a vital role in improving the quality of life for blind individuals, enabling better mobility, safety, and independence. Computer Vision, a field within Artificial Intelligence (AI) , is considerably looking forward to these challenges by allowing machines to analyze and understand visual information from the real world, such as images or videos. Employing progress in deep learning and neural networks, computer vision has significantly improved in recent years, leading to its general application across various domains. For instance, in healthcare, computer vision aids doctors in diagnosing diseases early by analyzing magnetic resonance images[10]. In agriculture, it assists farmers by tracking animals[11], and in transportation, it enables autonomous vehicles to recognize pedestrians[12], among other applications.

It's important to note that computer vision has also paved the way for improving real-time object detection systems. Among the various algorithms designed for object detection, YOLO has stood out as a strong algorithm because of its speed and accuracy, which are crucial for applications needing instant feedback. YOLO's ability to detect objects in real time provides a solid basis for creating tools that could assist visually impaired individuals by giving them immediate auditory feedback about their surroundings.

1.3 Objective of The Study

This thesis proceeds into integrating the latest version of YOLO, YOLOv8, with audio guidance systems to develop an assistive device specifically designed for blind individuals. The system aims to identify objects in the surroundings and deliver real-time audio descriptions, effectively converting visual information into auditory feedback. By using YOLO's enhanced precision and processing speed, this project aims to improve real-time applications in various and rapidly varying circumstances.

The implementation involves setting up a framework where a webcam captures scenes, which is then processed by the YOLO algorithm to detect and identify objects. The implementation of the same procedure on images and videos was also examined. The detected objects are subsequently relayed to a text-to-speech system that converts the information into audio signals, guiding the user about the location and counting of nearby objects. This setup not only enhances spatial awareness for blind users but also promotes their independence by reducing reliance on human assistance.

1.4 Deep learning for computer vision application

Computational models inspired by the composition and operations of the human brain are known as neural networks. They are made up of layer-organized, networked nodes known as neurons. Every neuron generates an output signal after applying an activation function (such as Relu, Tanh) to receive input signals. By varying the weights of the connections between neurons, neural networks can learn from data.

DL allows computational models composed of multiple processing layers to learn data representations with numerous levels of abstraction. These methods have dramatically improved the SOTA in speech recognition, visual object recognition, object detection, and many other domains, such as drug discovery and genomics. Deep learning discovers complex structures in large data sets using the backpropagation algorithm to indicate how a machine should change its internal parameters used to compute each layer's representation from the previous layer's. Deep convolutional nets have brought about breakthroughs in processing images, video, speech, and audio, whereas recurrent convolutional networks nets have shone light on sequential data such as text and speech[13]

In exploring DL architectures, particularly CNN , several critical components and processes define the effectiveness and efficiency of model training and operation. Among the

foundational elements of CNNs are the convolutional layer, pooling layer, fully connected layer, network training, and the backpropagation algorithm. Each of these aspects are essential for building robust and efficient neural network models.

1.4.1 Convolutional Layer

This is the first significant layer in CNNs. It involves filters that move across the input image and capture important features like edges, colors, patterns and textures. These filters help the network focus on specific parts of the image.

1.4.2 Pooling Layer

There's often a pooling layer after the convolutional layer. Pooling helps reduce the size of the data the network needs to process. It does this by summarizing the features captured in small areas of the image. For example, it might take the most significant number (max pooling) or the average (average pooling) from a small square in the picture. This makes the network faster and more efficient.

1.4.3 Network training

involves multiple steps: Initialization: Set initial weights and biases. These could be random or follow a specific distribution. Forward Propagation: Input data passes through the network, layer by layer, until the output layer is reached. Loss Calculation: Use a loss function to compute the difference between the network output and the actual values. This measures the network's performance.

1.4.4 Backpropagation

To optimize the neural network, it is essential to calculate the gradient of the loss function concerning each weight. This involves tracing the impact of the loss function through each layer of the network and then adjusting the weights to minimize the loss. This process, known as backpropagation, allows for the iterative refinement of the weights, ultimately leading to improved network performance.

1.4.5 Fully Connected Layers

Often follow convolutional and pooling layers in CNN architectures and play a crucial role in decision-making processes. These layers integrate learned features into final outcomes such as class scores, Where every neuron in one layer is connected to every neuron in the subsequent layer. They are critical in synthesizing the features extracted by convolutional and pooling layers into outputs that pertain to the task at hand.

1.5 Object Detection

Object detection is a CV task that involves identifying and locating objects in images or videos. It is essential in many applications, such as surveillance, self-driving cars, or robotics. The process will combine elements of an image by classification, which will represent what object categories are present in an image, and localization, which brings out the location of the objects in an image by drawing some bounding boxes around them. These features will typically be done by using either two-stage detectors, which first propose regions and then classify them, or one-stage detectors like, which simplify this process by doing both tasks in one shot



Figure 1.1: Object detection example

1.5.1 Single stage detection

Single-shot object detection uses a single pass of the input image to make predictions about the presence and location of objects in the image. It processes an entire image in a single pass, making them computationally effective. However, single-shot object detection is generally less

accurate than other methods, and it's less effective in detecting small objects. Such algorithms can be used to detect objects in real time in resource-constrained environments. YOLO is a single-shot detector that uses a fully CNN to process an image.

1.5.2 Two stage detection

Two-shot object detection uses two passes of the input image to make predictions about the presence and location of objects. The first pass is used to generate a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than single-shot object detection but is also more computationally expensive. In this thesis, single-shot object detection was deployed to better suit real-time applications.

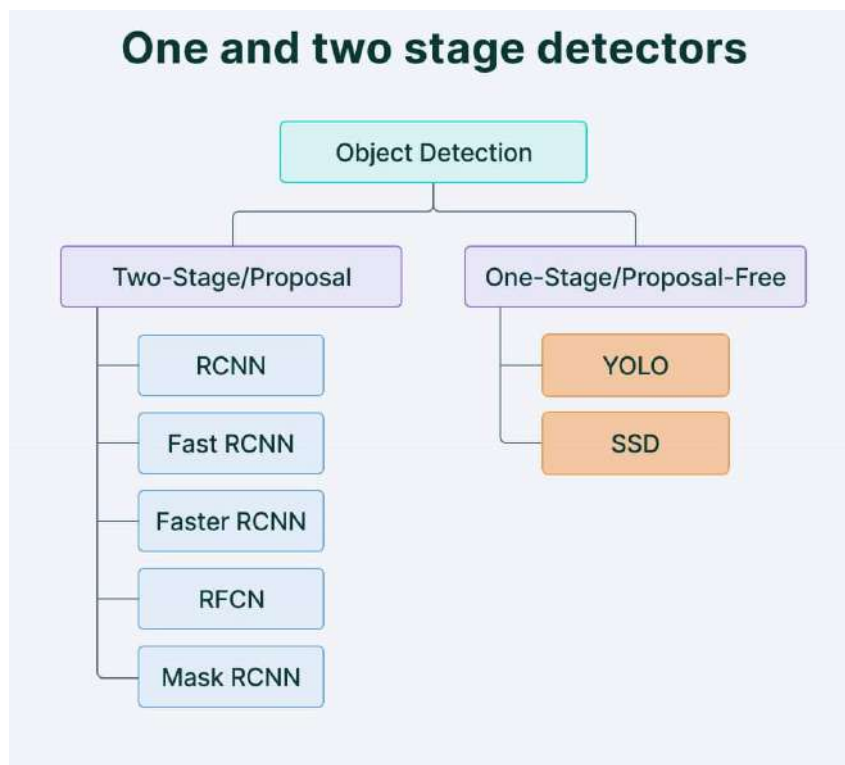


Figure 1.2: Object detection stages

1.6 Object detection models performance evaluation metrics

To determine and compare the predictive performance of different object detection models, we need standard quantitative metrics. Below the three most common evaluation metrics are represented.

1.6.1 Precision

It is a necessary metric in model evaluation as it serves to quantify the accuracy of the **positive predictions** made by the model. It represented how well the model distinguishes true objects from false positives. Precision provides insight into the model's ability to make positive predictions that are indeed accurate. A high precision score indicates that the model is skilled at avoiding false positives and provides reliable positive predictions.

$$\text{Precision} = \frac{TP}{TP + FP}$$

1.6.2 Recall

Recall, also known as perceptiveness or true positive rate, is another essential metric to evaluate model performance. It measures the model's capability to capture all appropriate objects in the image and evaluates the model's completeness in identifying objects of interest. A high recall score suggests that the model effectively identifies most of the relevant objects in the data.

$$\text{Recall} = \frac{TP}{TP + FN}$$

1.6.3 F1-Score

Another metric related to the trade-off mean of precision and recall, the F1-score, provides a measure of the model's performance, considering both false positives and false negatives. This metric is particularly useful when there is an imbalance between positive and negative classes in the dataset.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

1.6.4 Average Precision

Average Precision (AP) is calculated as the area under a precision vs. recall curve for a set of predictions. Recall and precision offer a trade-off that is graphically represented into a curve by varying the classification threshold. The area under this precision vs. recall curve gives us the Average Precision per class for the model. The average of this value, taken over all classes, is called Mean Average Precision (mAP). In object detection, precision and recall are not used for class predictions. Instead, they serve as predictions of boundary boxes for

1.7.1 Step-by-Step Insights into YOLO

1.7.2 Residual blocks

By dividing the original image into $N*N$ grid cells of equal shape YOLO will start the processing. Each cell in the grid is responsible for localizing and predicting the class of the object that it covers, Onwards with the probability/confidence value.

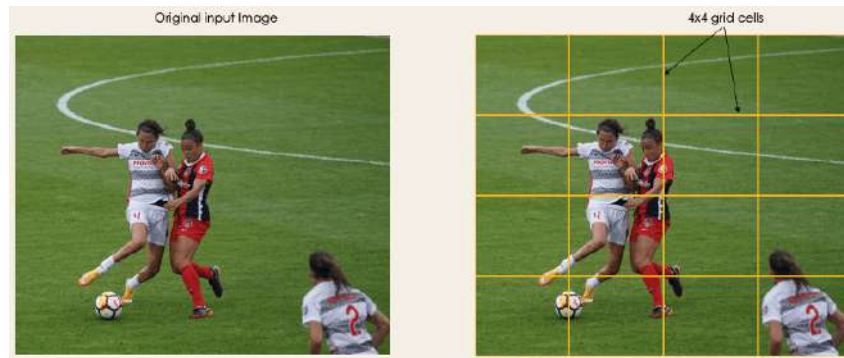


Figure 1.4: Grid cells

1.7.3 Bounding box regression

After dividing image, the algorithm will determine the bounding boxes which correspond to rectangles highlighting all the objects in the image. There is no limitations regarding the bounding boxes it can be as many object as in the image. YOLO controls the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation for each bounding box. $Y = [pc, bx, by, bh, bw, c1, c2]$ This is especially important during the training phase of the model. **pc** corresponds to the probability score of the grid containing an object. For instance, all the grids in red will have a probability score higher than zero. The image on the right is the simplified version since the probability of each yellow cell is zero (insignificant). **bx**, **by** are the x and y coordinates of the center of the bounding box with respect to the enveloping grid cell. **bh**, **bw** correspond to the height and the width of the bounding box. $c1$ and $c2$ correspond to the two classes.

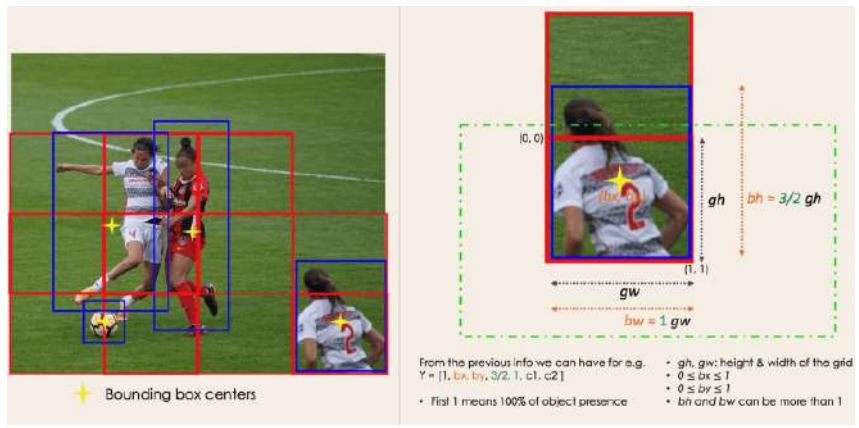


Figure 1.5: Bounding box regression

1.7.4 Intersection over Union

IoU is a metric to measure localization accuracy and calculate localization errors regarding object detection models. To calculate the IoU between the predicted and the ground truth bounding boxes, first, the intersecting area between the two corresponding bounding boxes for the same object is examined, then the calculation of the total area covered by the two bounding boxes— also known as the “Union” and the area of overlap between them called the “Intersection.” The metric mentioned above gives the ratio of the overlap to the total area, providing a good estimate of how close the prediction bounding box is to the original bounding box.

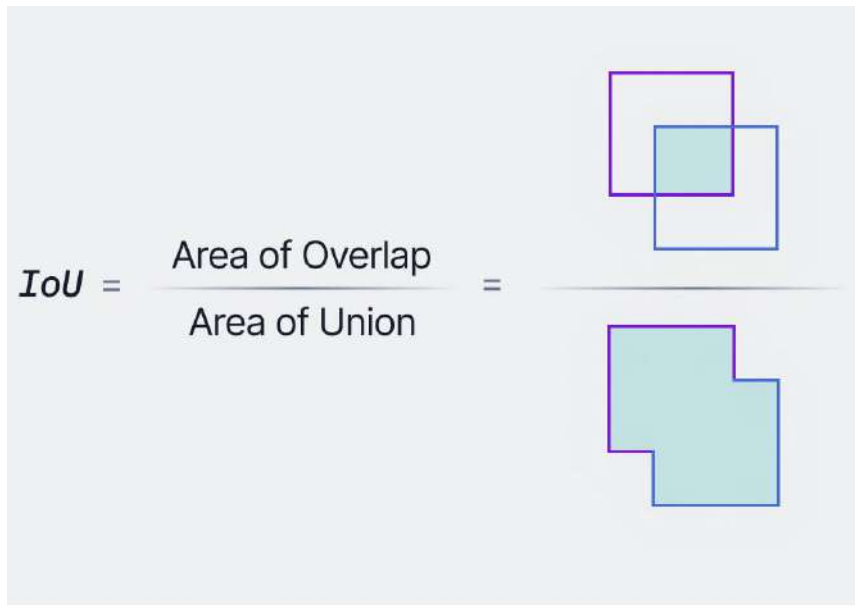


Figure 1.6: IoU

1.7.5 Non-Maximum suppression

NMS As a post-processing technique used in YOLO as the last. It is used to remove redundant detections and retain only the most confident ones. It compares the predicted bounding boxes of the detected objects and discards the ones that have a high overlap or intersection.

NMS involves two main steps, **Thresholding** and **Suppression**. The thresholding step filters out detections with low confidence scores. The suppression step removes redundant detections that have a high overlap with each other, retaining only the most confident ones. By performing NMS, YOLO can reduce the number of positives and improve the accuracy of object detection.

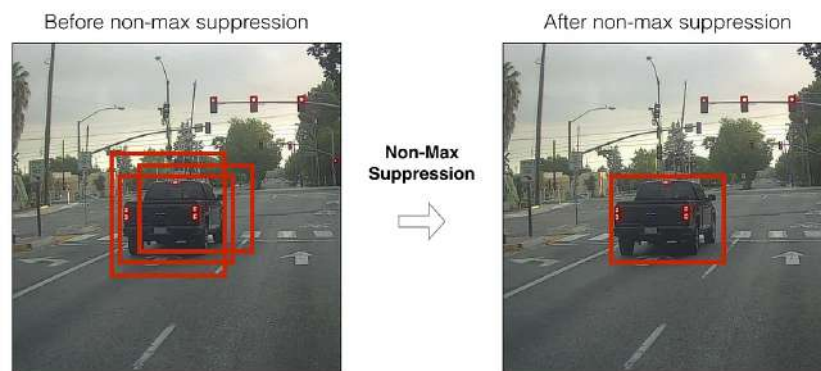


Figure 1.7: NMS

Chapter 2

Related Works

The Related Works chapter will extensively analyze cutting-edge research papers on object detection, centering on different versions of YOLO models. The examination will provide detailed insights into the precision and efficiency of each model, offering thorough explorations of how audio feedback is integrated into object detection, object distance to the user and location, and the hardware and tools used in each paper's approach.

2.1 A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS

This comprehensive review outlines the evolution of YOLO (You Only Look Once) from its initial version (YOLOv1) to the latest iterations, including YOLOv8 and variations integrating Neural Architecture Search (YOLO-NAS) and transformers. It highlights key advancements in network architecture, training methodologies, and post-processing techniques across versions. YOLO has become a central real-time object detection system for robotics, driverless cars, and video monitoring applications. It describes the standard metrics and post-processing and discusses the significant changes in network architecture and training methods for each model. The YOLO (You Only Look Once) family of object detection models has significantly impacted computer vision, offering real-time object detection by directly predicting classes and bounding boxes. YOLOv1 introduced the concept of single-stage object detection but needed help with small and overlapping objects due to its grid system. YOLOv2 improved upon these limitations with batch normalization, anchor boxes, and high-resolution classifica-

tion. YOLOv3 further enhanced accuracy through multi-scale predictions and the Darknet-53 backbone. Subsequent iterations, like YOLOv4 and YOLOv5, optimized training and feature extraction with methods such as spatial pyramid pooling, mosaic augmentation, and improved feature pyramid networks. Recent versions, YOLOv8 and YOLO-NAS, have leveraged neural architecture search and quantization techniques for better performance. Due to their speed and accuracy, these models have been widely adopted in various fields, including autonomous vehicles, precision agriculture, and medical imaging[2].



Figure 2.1: A Timeline of YOLO versions[2]

Version	Date	Anchor	Framework	Backbone	AP (%)
YOLO	2015	No	Darknet	Darknet24	63.4
YOLOv2	2016	Yes	Darknet	Darknet24	78.6
YOLOv3	2018	Yes	Darknet	Darknet53	33.0
YOLOv4	2020	Yes	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Yes	Pytorch	YOLOv5CSPDarknet	55.8
PP-YOLO	2020	Yes	PaddlePaddle	ResNet50-vd	45.9
Scaled-YOLOv4	2021	Yes	Pytorch	CSPDarknet	56.0
PP-YOLOv2	2021	Yes	PaddlePaddle	ResNet101-vd	50.3
YOLOR	2021	Yes	Pytorch	CSPDarknet	55.4
YOLOX	2021	No	Pytorch	YOLOXCSPDarknet	51.2
PP-YOLOE	2022	No	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	No	Pytorch	EfficientRep	52.5
YOLOv7	2022	No	Pytorch	YOLOv7Backbone	56.8
DAMO-YOLO	2022	No	Pytorch	MAE-NAS	50.0
YOLOv8	2023	No	Pytorch	YOLOv8CSPDarknet	53.9
YOLO-NAS	2023	No	Pytorch	NAS	52.2

Figure 2.2: YOLO versions architecture[2]

2.2 Real-Time Object Detection Using Yolo Algorithm for Blind People

Real-time object detection systems are essential for enhancing independence and mobility in the developing area of assistive technologies for visually impaired individuals. Different object detection algorithms, including the CNN family, YOLO, and SSD, have been studied for their effectiveness in this domain. Comparative analyses of speed and accuracy show that YOLO

stands out as one of the best algorithms due to its excellent performance metrics. Kasture[15] leveraged the YOLO V3 algorithm within a web application to provide auditory feedback on detected objects, thereby assisting blind users in real-time navigation. Their system identifies a scope of objects and communicates this information through an innovative audio feedback mechanism, making it highly accessible for visually impaired users. However, challenges such as dependency on internet connectivity and environmental variability remain.

2.3 An Real Time Object Detection Method for Visually Impaired Using Machine Learning

Recent work has concentrated on real-time object detection systems combined with auditory feedback to enhance spatial awareness and independence. The paper developed a system integrating YOLO V3 and R-CNN with audio feedback to identify objects and communicate their locations verbally, enhancing the ability of the visually impaired to navigate both familiar and unfamiliar environments safely and enabling greater independence, offering a more effective and accurate solution compared to other algorithms even though it faces challenges regarding computational demands on mobile devices and varying environmental conditions[3].

S. No	State of art methods	MAP in %
1	R-CNN+Yolo	96.48
2	CNN	90.23
3	Yolo	89.36
4	RCNN	91.26
5	Hypernet	84.45
6	Edge based detection	88.78

Figure 2.3: Accuracy comparison[3]

2.4 Object Detection for Blind People Using Yolov3

In this project, the authors have designed a website that assists blind persons in recognizing diverse things in their surroundings using the YOLO Version 3 algorithm. This combines many technologies to develop a prosperous website that helps individuals with vision impairments recognize different objects in their surroundings in real-time and directs them through an auditory output. The COCO (Common Objects in Context) Dataset, which contains 81 distinct objects, was used, since datasets and Neural Networks are necessary for this sort of

web application to recognize objects. Regarding detection, The Haar cascade algorithm was presented in combination with YOLO as a pre-processing technique to detect some specific objects. However, the authors mentioned limitations, such as the lack of scene perception, YOLO concentrates on item detection but may not offer a comprehensive understanding of the scene since efficient navigation for blind people requires comprehending the context and the scene. It can also detect a mixture of items but may have trouble correctly identifying particular objects, especially in complicated or blocked settings. Difficulty in identifying small or complex items has been reported as well.[16].

2.5 Understanding of Object Detection Based on CNN Family and YOLO

Juan Dual's paper provides a comprehensive overview of the YOLO (You Only Look Once) algorithm within the context of various CNN-based object detection models. It details the growth of YOLO from YOLOv1 to YOLOv2 and describes its architecture and characteristics. The paper contrasts YOLO with Faster R-CNN by comparing factors such as speed, cost, accuracy, and complexity, highlighting the advantages and disadvantages of each. Lastly, it concludes the summary of the YOLO performance and Faster R-CNN, positioning YOLO as a more efficient alternative for real-time object detection. The result underscores YOLO's advancements in balancing detection speed and accuracy, making it a key algorithm in object detection[17].

2.6 Real-Time Object Detection with YOLO

In real-time object detection, YOLO algorithm represents a significant improvement, combining high speed with robust accuracy in detecting multiple objects simultaneously. The study by Geethapriya[18] emphasizes YOLO's capability to perform at 45 frames per second while maintaining lower false positives than traditional methods like R-CNN. The result makes YOLO particularly suitable for applications requiring rapid and reliable detection, such as autonomous driving and real-time surveillance. However, despite its strengths, the algorithm faces challenges such as localization errors and sensitivity to object scales, pointing to areas for further research and development. Enhancing YOLO's grid strategies and bounding box predictions could address these issues, making it even more effective for wide-ranging real-

time applications.

2.7 An Assistive Model for Visually Impaired People using YOLO and MTCNN

In recent outcomes within assistive technologies for the visually impaired, integrating artificial intelligence for real-time object and facial recognition presents a promising advancement. In this area, Rahman introduced an untapped application of YOLO and MTCNN (Multi-Task Cascaded Convolutional Networks) algorithms within a Raspberry Pi-based system designed to enhance environmental perception for visually impaired users via audio feedback. This system supports real-time interaction and achieves high accuracy, with object detection accuracy ranging from 63 to 80 and facial recognition accuracy from 80 to 100. Such systems underscore the potential of integrating cost-effective computing platforms like Raspberry Pi with advanced neural network models to deliver practical and accessible solutions for the visually impaired. However, the variability in accuracy and the challenges posed by environmental factors suggest further refinement and testing under diverse conditions[19].

2.8 Object Detection System For The Blind With Voice guidance

In their 2021 publication, Karmarkar and Honmane introduced an innovative system on Android that merges the You Only Look Once (YOLO) algorithm with text-to-speech technology to provide real-time guidance to blind users. This system detects objects and verbally communicates their details, improving spatial awareness and independence for visually impaired individuals. While the system shows promising results in object detection that enables practical use, its reliance on specific environmental conditions indicates the need for further accuracy and compatibility improvement on each device. These enhancements are essential in offering more robust solutions that can significantly enhance the quality of life for visually impaired individuals[20].

2.9 Real Time Object Detection with Audio Feedback using Yolo vs. Yolov3

The comparative study of object detection algorithms YOLO and YOLOv3 by Mahendru and Dubey highlights YOLOv3's enhanced abilities in detecting small and distant objects with great accuracy, leveraging a multi-scale detection architecture. The system's integration of audio feedback is particularly notable, as it translates the detection results into auditory signals, thereby aiding visually impaired users in real-time navigation. While the study shows YOLOv3's superior performance, it also points to the need for further algorithm responsiveness and adaptability to varied environmental conditions, suggesting a promising direction for future research in making these technologies more accessible and practical for real-life applications[21].

2.10 Real Time Object Detection

Kaur, Yadav, and Joshi, indicate YOLO's powers in processing images at high speeds while preserving accuracy, making it an ideal solution for applications requiring quick response, such as autonomous driving and advanced applications for aiding blind people. The algorithm's distinctive approach of treating object detection as a regression problem simplifies the computational process, allowing faster processing times than traditional methods like R-CNN and SSD. However, despite its advantages, the YOLO algorithm struggles with detecting small or fast-moving objects, highlighting an area for potential improvement. This limitation suggests a direction for future research, possibly integrating YOLO with other sensor technologies to create a more robust object detection system capable of operating effectively in diverse and dynamic environments. Additionally, it emphasizes the ability of the algorithm to process images at high speeds (45 frames per second) and with considerable accuracy. The authors tested the algorithm's performance in various settings, demonstrating its robustness and reliability across different scenarios[22].

2.11 Object Detection Using Machine Learning for Visually Impaired People

Integrating advanced machine learning techniques has shown promising potential to enhance object detection abilities. This paper significantly contribute to the mentioned area by implementing a system that utilizes RetinaNet and neural network technologies to facilitate accurate navigation aids for indoor and outdoor settings. The system looked at different environments, crowded and empty, and obtained great accuracy during the challenging area by looking at different algorithms and exploring them deeply theoretically, It identifies objects with high accuracy and communicates this information through audio outputs, thus aiding visually impaired individuals in understanding their surroundings effectively. While the system shows potential, its performance across diverse environmental conditions and its dependence on specific hardware configurations highlight areas for future improvement. These findings underscore the need for ongoing research to refine and adapt machine learning applications in assistive technologies, ensuring they are robust, versatile, and widely accessible[23].

2.12 Object Detection Featuring 3D Audio Localization for Microsoft HoloLens: A Deep Learning Based Sensor Substitution Approach for the Blind

The integration of augmented reality (AR) and artificial intelligence (AI) offers novel tracks to enhance navigation and interaction within the environment. An innovative application of this integration in their development of a system that combines YOLOv2-based object detection with 3D audio localization via Microsoft HoloLens was studied.

User	Client	Server
"Scan"	<ul style="list-style-type: none"> records image sends it to server reads list of objects aloud 	<ul style="list-style-type: none"> receives picture start object detection return list of recognized objects
"Objects"	<ul style="list-style-type: none"> rereads list of objects aloud 	
"Cup"	<ul style="list-style-type: none"> highlights cup with pink cube 	
"Distance"	<ul style="list-style-type: none"> announces distance to selected object start 3D audio ping 	
"All"	<ul style="list-style-type: none"> selects or deselects all found objects 	

Figure 2.4: User interaction Through system[4]

This system detects objects in real-time and also provides spatial audio feedback,a speech recognition program was setup to interact with the user. In order to test how well voice com-

mands are recognized and processed, the debug output for each recognized voice command has been used. The voice commands that should be recognized were limited to “Scan”, “Start”, “Stop” and “Chair”. Such technological advancements underscore the potential of AR and AI to transform everyday experiences for individuals with visual impairments, providing them with greater independence and mobility. The reliance on continuous network connectivity and the processing demands on external servers highlight areas for further development, particularly in enhancing the system’s autonomy and operational efficiency[4].

2.13 Deep learning based object detection and surrounding environment description for visually impaired people

Deep learning for both object detection and environmental interaction represents a forward leap in functionality and user experience. regarding this paper , It introduces a comprehensive system that detects objects using a TensorFlow-based SSDLite MobileNetV2 model and then describes the surrounding environment through an innovative ‘ambiance mode.’ Trained on a diverse dataset, including weather conditions, this system offers auditory feedback that enhances spatial and situational awareness for visually impaired users. the system will demonstrate robust performance metrics on standard benchmarks relying on Raspberry Pi hardware that presents scalability and environmental adaptability challenges.[24].

2.14 Object Detection with Voice Guidance to Assist Visually Impaired Using YOLOv7

Innovative applications of deep learning models in assistive technologies for the visually impaired have shown marked progress, particularly with integrating YOLOv7 in object detection systems. Authors exemplify this advancement by combining YOLOv7’s precise detection capabilities with voice guidance technology, offering a practical solution for enhancing the independence of visually impaired individuals. This system identifies objects with high accuracy and communicates this information through voice, thus making spatial navigation more accessible.

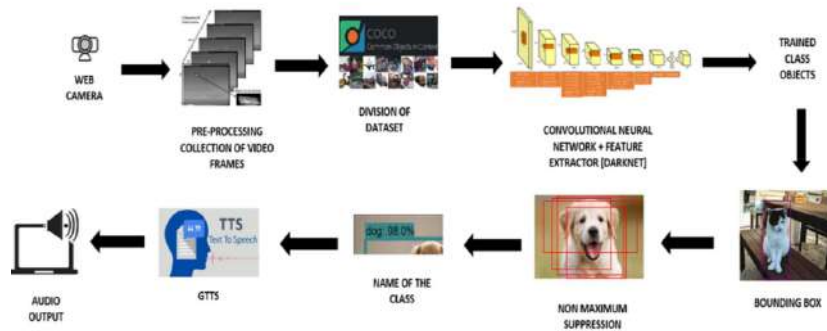


Figure 2.5: Design diagram[5]

Future developments could focus on reducing the system’s computational demands and extending its applicability to more complex environments[5].

2.15 Android Based Object Recognition for Visually Impaired

The research by Saeed presents a feasible solution for real-time, offline object recognition on Android devices. This solution significantly benefits visually impaired users by enhancing their environmental awareness. The system was evaluated using a dataset of 600 images across various conditions. It achieved the highest accuracy, at 95.36%. The primary challenge was ensuring accurate object recognition under varying conditions without excessive computational load. Optimization of classifier parameters and feature selection were critical to balancing performance and resource usage. Future improvements include expanding the object database, incorporating more complex recognition tasks, and further optimizing the system for better performance and lower energy consumption. The proposed Android-based object recognition system will be used to implement a money reader for the Egyptian currency. No such money reader is known among the blind and visually impaired[25].

2.16 Object Detection and Recognition for a Pick and Place Robot

Recent studies have demonstrated that YOLO has an impressive processing speed of 155 frames per second while achieving double the mean Average Precision (mAP) compared to other real-time detectors. Despite this notable efficiency, the methodology has its limitations. For exam-

ple, while contemporary detection techniques have effectively reduced the occurrence of false positives, YOLO is inclined to exhibit more localization errors. Anyhow, YOLO demonstrates exceptional generalization capabilities across diverse domains. For instance, it outperforms approaches such as the Deformable Parts Model (DPM) and R-CNN in transferring knowledge from natural images to other domains, including art, exhibiting adaptability across various applications.[26].

2.17 Assistive Technologies for Obstacle Detection and Identification

Applications centered on sensing obstacles near the user and alerting them through alarms or beeping sounds implemented on sensory devices such as ultrasonic, smart sticks with obstacle detectors, mobile phones, and navigators. These devices are expensive and could sometimes be a barrier to the user. In some situations, tactile signs and Braille texts are labeled at the top of the items for identification. Moreover, High-tech systems such as Radio Frequency Identification Devices (RFID), barcodes, or talking labels can be used to find objects in near distance[27][28].

2.18 Absolute Distance Prediction Based on Deep Learning Object Detection and Monocular Depth Estimation Models

The You Only Look Once (YOLOv5) model, which is renowned for its ability to detect objects in real time with high accuracy, specifically enhances speed and performance over its predecessors, making it an effective tool for detecting and localizing objects within an image. The challenge of depth estimation using a single (monocular) camera is established since monocular cameras capture 2D images, making it challenging to intimate depth directly. To address this, the authors use a deep autoencoder network, trained in a self-supervised form, to estimate depth from the image, providing relative distances within the scene. By integrating these two techniques, the framework effectively predicts the absolute distance to objects, achieving an accuracy of 96% with a Root Mean Square Error (RMSE) both the method and result out of this study is presented in Figure [6].

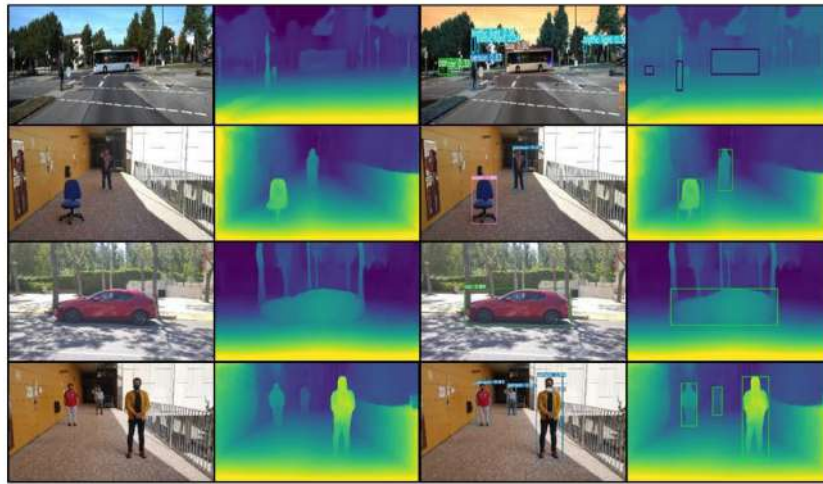


Figure 2.6: Visual process of whole network[6]

Figure 5	Object	Absolute distance (m)	Predicted distance (m)	Error (m)
Row 1	Car	53.9	53.21	0.69
	Person	21.5	21.35	0.15
	Bus	48.7	48.13	0.57
Row 2	Chair	3.5	3.45	0.05
	Person	8.0	8.09	0.09
Row 3	Car	10.1	9.83	0.27
Row 4	Person 1	8.0	8.13	0.13
	Person 2	12.0	11.69	0.31
	Person 3	4.0	3.88	0.12

Figure 2.7: Estimated distance vs. absolute distance[6]

2.19 Existing Systems

2.19.1 Wearable Devices

Wearable devices are widely used in object detection and recognition for visually impaired individuals. These devices typically incorporate sensors and cameras to detect objects in the user's surroundings, providing audio or tactile feedback. An example is the **Orcam MyEye**, which utilizes a camera and Optical character recognition (OCR) technology to read text and recognize faces, products, and more.



Figure 2.8: Orcam MyEye device

2.19.2 Smartphone Applications

Numerous smartphone applications assist visually impaired individuals in object recognition by leveraging the smartphone's camera and computer vision algorithms for real-time detection and classification. Examples include the **Be My Eyes app**, which connects blind users with sighted volunteers for object identification, and Seeing AI by Microsoft, which can recognize faces, read text, and identify objects.

2.19.3 Robotics

Robotics represents another approach to object detection and recognition for visually impaired users. Researchers have developed robotic systems that can detect and recognize objects and navigate environments to provide assistance. An example is the Blind Explorer, a robotic platform capable of detecting obstacles and offering audio feedback to help users navigate through complicated environments.

2.19.4 Websites

Considerable websites have been developed to assist visually impaired individuals with object detection and other tasks by employing web-based technologies and computer vision algorithms for real-time object detection and classification. These websites typically utilize user-uploaded images and live streams to analyze objects, text, and scenes on the website or to provide remote individual assistance. One example is a **web captioner**, which offers real-time captioning and text recognition.

Chapter 3

Proposed Method

This chapter provides a comprehensive overview of the methodology employed in each part of the project, outlining the precise procedures and approaches utilized to ensure successful execution.

3.1 Tools and Libraries

Various libraries, such as CV Numpy and the others, Torch as the framework, are installed and used to succeed in different tasks. The YOLO model imported from ultralytics libraries who has produced the intended YOLO version[7]. The Pygame Python module is used to play the audio. Pyttsx3 and Google Text-To-Speech (gTTS) are libraries that provide a text-to-speech conversion engine in Python offline and online, respectively. The second one, gTTS, was presented only to support a wide range of languages in the algorithm to tackle concerning distinct languages. The main algorithm will work on the Pyttsx3 to provide more accessible feedback without worrying about an internet connection. It is also faster than gTTS providing both male and female voices.

3.2 Dataset and Data Augmentation

The dataset included 2282 images and 3523 annotations around the specific objects, which were completely manually and carefully selected to achieve the best result. Only square shape box was determined for the labeling object that can be used related to bounding boxes. While there were two other options, such as smart polygon and smart labeling, that had a misunderstanding between segmentation and detection tasks for the algorithm.

The detection, labeling, and annotating data was used on RoboFlow[29], a web-based platform.

The dataset covers both indoor and outdoor navigation objects, with 17 classes related to supermarkets, shopping, walking tours, and home surfing. After labeling and annotating objects, data augmentation was considered for almost all variations, such as flipping horizontal and vertical, to create a robust model for the training process and tackle the algorithm's limitation of not detecting objects in diverse environments through bad lightening areas, noisy data, and disparities in color. Hence, data transformations such as Rotation about 90°, Clockwise, Counter-Clockwise, Upside Down, Grayscaleing was considered for 17% of images, Hue and saturation Between -19° and +19°, Between -29% and +29% respectively, Brightness Between -17% and +17%, Exposure Between -13% and +13% , Blur Up to 2.5 pixels and Noise Up to 1.29% of pixels were adjusted and added to the training data. Furthermore, The Mosaic augmentation was automatically applied during training by YOLOv8, but it is disabled before the last 10 epochs. Data augmentation has been accomplished using the RoboFlow platform as well[29].

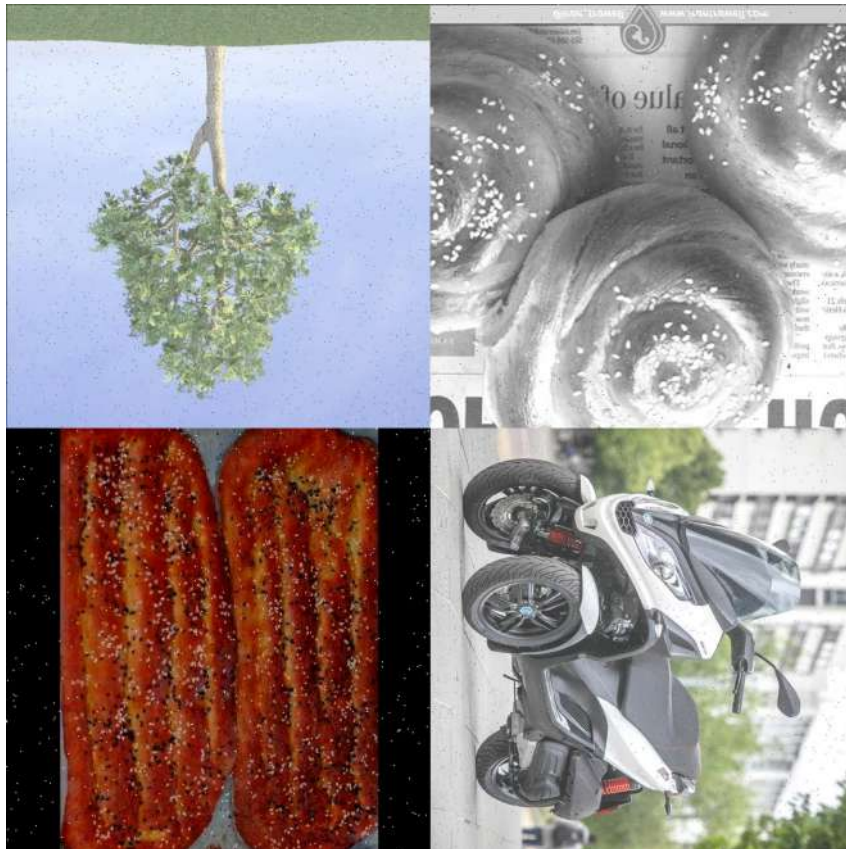


Figure 3.1: Sample of data augmentation

3.3 YOLOv8

YOLOv8 architecture is divided into three essential parts: **Backbone, Neck, and Head**.

The backbone acts as a feature extractor, capturing patterns in the first layers, such as edges

and textures. The *neck* part will perform as a bridge between the head and the backbone. It gathers feature maps from different stages of the backbone and provides information related to the accuracy and speed of the model. Another functionality of this part is Performing concatenation or fusion of features of different scales to ensure the network can detect objects of different sizes. The head is the final part of the network and is responsible for generating the network's outcomes, such as bounding boxes and confidence scores for object detection. The YOLO architecture utilizes a local feature analysis approach instead of processing the entire image at once, aiming to reduce computational efforts and facilitate real-time detection. The algorithm employs convolutions multiple times to extract feature maps. Convolution, a mathematical operation that combines two functions to produce a third one, is commonly used to apply filters to images or signals in computer vision and signal processing, thereby emphasizing specific patterns. Additionally, it extracts features from inputs such as images in CNN. Convolutions are characterized by Kernels, Strides, and Paddings.

The *kernel*, also known as the filter, is a small array of numbers that moves slowly across the input (image or signal) during the convolution operation. The goal of Kernels is to apply local operations to the input to detect specific characteristics. Each element in the kernel represents a weight multiplied by the corresponding value in the input during convolution.






Name	Kernel	Image Result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Mean Blur	$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$	
Laplacian	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
Gaussian Blur	$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$	

Figure 3.2: Kernel operation

Another factor related to Convolution is *stride*, which is the amount of displacement the kernel considers as it moves across the input. A stride of 1 means the kernel moves one position at a time. Stride directly influences the spatial dimensions of the convolution output. Larger of them can decrease the dimensionality of the output and reduce computational effort, thereby increasing the speed of the operation, which can directly impact quality. while smaller strides retain more spatial information.

Padding refers to adding extra pixels around the edges of the input image, also known as zero padding. It occurs before applying convolution operations. This ensures that information at the image's edges is treated the same way as information in the center during convolution

operations. When a filter (kernel) is applied to an image, it typically goes through the image pixel by pixel.

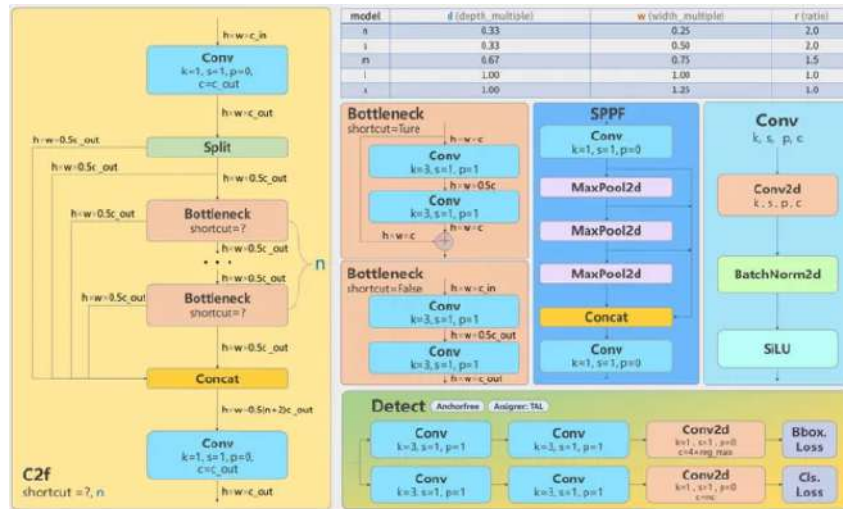


Figure 3.3: YOLOv8 Architecture

YOLOv8[7] deep CNN architecture is renowned for its high-speed and accurate detection capabilities. YOLOv8 utilizes a similar backbone as YOLOv5, making changes to a specific layer that combines advanced features with relative information to improve detection accuracy. It incorporates advanced backbone and neck architectures to enhance feature extraction and object detection performance. In addition, YOLOv8 utilizes an anchor-free split Ultralytics head, resulting in improved accuracy and a more efficient detection process compared to anchor-based approaches. YOLOv8 is designed to manage diverse modes such as training, predicting, and validating, as well as different tasks including detection, segmentation, pose estimation, and classification. In the output layer, YOLOv8 uses the sigmoid function as the activation function for the objectness score, indicating the probability that the bounding box contains an object. It utilizes the softmax function for the class probabilities, representing the probabilities of objects belonging to each possible class. It is utilized for its SOTA performance in distinct tasks such as Detection, Pose, Classification, Oriented Detection and Instance Segmentation. YOLOv8 can be executed from the command line interface (CLI) or installed as a PIP package. Moreover, it comes with multiple integrations for labeling, training, and deployment. This version of YOLO focuses on achieving an optimal balance between accuracy and speed, making it suitable for real-time object detection tasks in various application areas. It is utilized for its SOTA performance in distinct tasks such as Detection, Pose, Classification, Oriented Detection and Instance Segmentation. YOLOv8 is also presented in various sizes, parameters, speed, and accuracy. The medium size was deployed to have a trade-off for accuracy, speed, and suitable parameters. Therefore, the intended model has been derived by using

YOLOv8 transfer learning (pre-trained model) techniques and fine-tuning hyperparameters during training to enhance detection rates. Moreover, in Figure 3.4, a comparison took place related to previous versions of the YOLO and YOLOv8 that shows great mark in both speed and accuracy of this version.

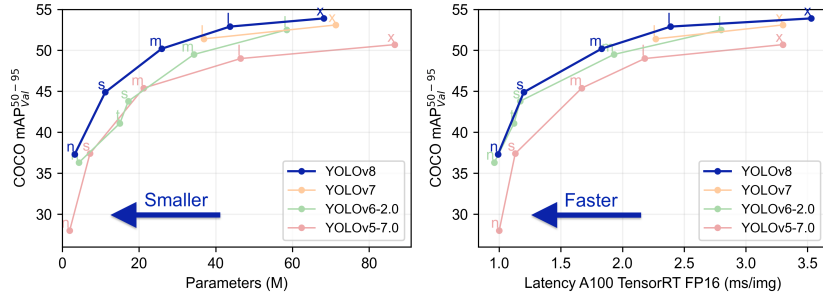


Figure 3.4: Accuracy and speed of YOLOv8[7]

3.4 Training

High-quality and diverse images with accurate bounding boxes and labels are essential. The input images contain objects in almost every category, such as automobiles, animals, fruits, dairy product, furniture, obstacles, and groceries, at different positions, angles, and lighting conditions, Regarding the necessary pre-processing. Yet Another Markup Language (YAML) is a human-readable data serialization language. It is commonly used for configuration files and in applications storing the number of the classes , annotations and labeling train test, and valid sets mostly in YOLO object detection. By reading the specific YAML file related to the dataset and considered configurations, The model will step forward for each corresponding task to improve the training process.

```

1 results = model.train(
2     data="/content/data.yaml",
3     epochs=20,
4     imgsz=640,
5     lr0=0.001,
6     lrf=0.01,
7     cos_lr=True,
8     save_period=10,
9     batch=16,
10    workers = 8,
11    momentum=0.937,
12    optimizer = 'AdamW',
13    weight_decay=0.0005,
14    dropout=0.5,
15    verbose=True,
16    name='exp',
17    val=True,
18    save=True
19 )

```

Figure 3.5: Training configurations.

Feature Extraction: The first step is to use convolutional layers to extract image features. These features capture details such as edges, textures, shapes, and more complex patterns as they pass through successive layers.

Forward Propagation: The input image is passed through the network, and the model produces predictions, including bounding boxes, class probabilities, and confidence scores.

Loss Calculation: The loss is calculated by comparing the model's predictions to the ground truth (actual bounding boxes and class labels).

Backpropagation: The loss is propagated backward through the network to adjust the weights. This process involves calculating gradients and updating the weights using an optimization algorithm like Stochastic Gradient Descent (SGD) or Adam, Which in our case the AdamW was added since it would work better with larger dataset.

Iteration: The forward and backward processes repeat for many iterations (epochs), with the model gradually improving its predictions by minimizing the loss.

Role of Bounding Boxes: Bounding boxes are critical for training object detection models since they provide spatial information about objects locations within the images. They will show the impact of learning through:

Localization Information: Bounding boxes provide the coordinates for the *top-left* and *bottom-right* corners of objects in the images. This information helps the model learn to localize objects accurately.

Regression Targets: The model learns to predict bounding boxes by regressing to the coordinates provided in the training data. The quality of these predictions is crucial for the model's

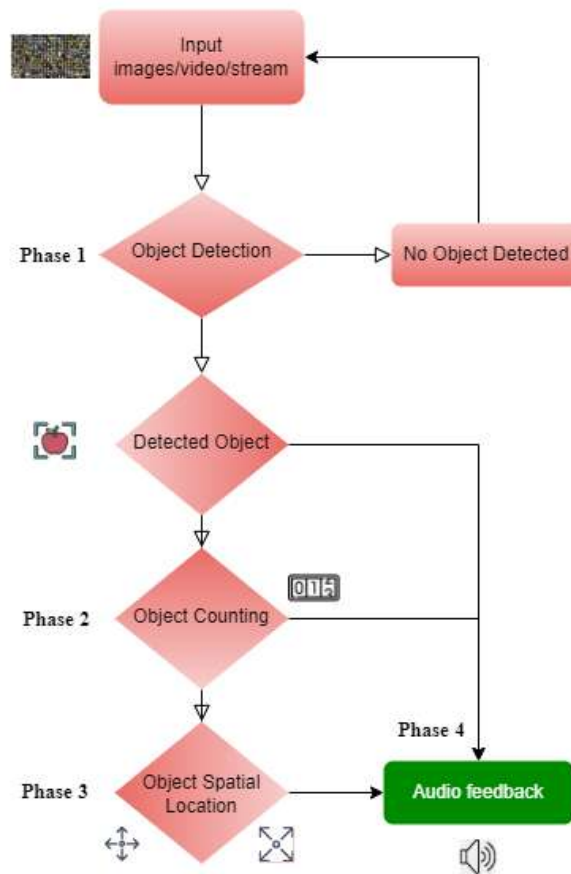


Figure 3.7: System design

3.5 Phase 1

3.5.1 Object Detection

Regarding testing the model on images, They will converted to RGB format and scaled appropriately. The corresponding YOLO model processes this image to detect objects, returning the results which include bounding boxes, confidence scores, and class labels. Bounding boxes are drawn on the image to visualize detected objects. For each detected object, a rectangle is plotted around it, and a label with the class name and confidence score is added. The result is displayed as the **detected objects** and saved in the output path.

```

1 # Initialize custom YOLO model
2 model = YOLO('feedback.pt')
3
4 def load_image(image_path):
5     # Load image
6     img = cv2.imread(image_path)
7     return img
8
9 def detect_objects(img):
10    # Convert image to RGB
11    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
12    # Perform detection
13    results = model(img_rgb)
14    return results
15
16 def draw_bounding_boxes(img, results):
17    # Extracting bounding boxes, labels, and confidence scores
18    img_height, img_width = img.shape[0], img.shape[1]
19    labels = results[0].boxes.cls
20    coords = results[0].boxes.xyxy
21    confs = results[0].boxes.conf
22
23    for i in range(len(labels)):
24        x1, y1, x2, y2 = int(coords[i][0]), int(coords[i][1]), int(coords[i]
25        ][2]), int(coords[i][3])
26        label = model.names[int(labels[i])]
27        conf = confs[i]
28
29        # Draw bounding box
30        cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 0), 2)
31        # Add label and confidence score
32        text = f"-label -conf: .2f "
33        cv2.putText(img, text, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX,
34        0.9, (255, 0, 0), 2)
35
36    return img
37
38 # Example usage
39 img_path = 'Car.jpg'
40 img = load_image(img_path)
41
42 results = detect_objects(img)
43 img_with_boxes = draw_bounding_boxes(img, results)
44
45 # Display the image with bounding boxes
46 cv2.imshow('Detected Objects', img_with_boxes)
47 cv2.waitKey(0)
48 cv2.destroyAllWindows()

```

Figure 3.8: Object detection script

The "for loop" iterates over the number of detected objects corresponding to the label of them. Finally, Image with bounding box , class , and confidence threshold will display by the algorithm.

3.6 Phase 2

3.6.1 Object Counting

The number of the detected objects is followed based on their labels and spatial locations. A **defaultdict** named *detected_objects* is initialized to store the counts and locations of each object type. Each object's count is incremented in *detected_objects [label]['count']*. The feedback is created by repeating through the detected objects and generating illustrative sentences based on their numbers and locations.

```
1 # Initialize defaultdict to store counts and locations
2 detected_objects = defaultdict(lambda: {'count': 0, 'locations':
3     defaultdict(int) })
4 for i in range(len(labels)):
5     label = model.names[int(labels[i])]
6     x1, y1, x2, y2 = int(coords[i][0]), int(coords[i][1]), int(coords[i]
7         ][2]), int(coords[i][3])
8     # Access count for a specific label
9     detected_objects[label]['count']
10
11 # Adding count
12 detected_objects[label]['count'] += 1
```

Figure 3.9: Code snippets for counting objects

3.7 Phase 3

3.7.1 Object Distance

Three distinct methods were studied to detect the object spatial location. The simplest one was measuring the distance of an object from the camera by knowing its actual width/height and the focal length of the camera, in addition to the sensor size either determining an estimated width/height for each class and measuring the distance with the help of a third-party app to join the mobile phone's camera to the laptop. This strategy was not as accurate as needed since It is not easy to assign only one size for the images in a class, for example, one size for all images related to class CAR, since the size of the classes is different image by image.

$$\text{Distance} = \frac{\text{Actual W/H of the Object} \times \text{Focal Length}}{\text{W/H of the Object in Image}}$$

Another experienced method was looking at the size of the bounding boxes and setting

a threshold for them to determine three stages such as **Close, Far, Very Far**. However, the algorithm had some errors regarding this technique; for example, it looked at the bounding boxes of the objects and estimated the ones with low Confidence thresholds with the small size of the bounding boxes around them as a "far" distance since it was not always true.

3.7.2 Object Spatial Location

The mentioned methods were not available to completely accomplish through user experience, Therefore, best approach was instead of information related to distance of the object , dividing the images into a 3x3 grid to categorize the location of each detected objects. It will use **the center** point of the object's bounding box to determine its position.

x1: Left boundary of the bounding box.

y1: Top boundary of the bounding box.

x2: Right boundary of the bounding box.

y2: Bottom boundary of the bounding box.

img_width, img_height represented the image dimensions, These provide the width and height of the image, necessary for converting normalized coordinates into pixel values and for location categorization.

```
1 img_width = img.shape[1]
2 img_height = img.shape[0]
```

Figure 3.10: Image dimensions.

The center of the bounding box is calculated by averaging the x-coordinates and y-coordinates of the bounding box:

$$\text{center_x} = (x1 + x2) / 2$$

$$\text{center_y} = (y1 + y2) / 2$$

This center point is used to determine the location category. The image is conceptually divided into a grid with nine regions.

The divisions are:

Horizontal: Divided into three equal columns: **left, center, right**.

Vertical: Divided into three equal rows: **top, middle, bottom**.

The boundaries of these divisions are:

Horizontally:

$\frac{\text{img_width}}{3}$: First column boundary.

$\frac{2 \cdot \text{img_width}}{3}$: Second column boundary.

Vertically:

$\frac{\text{img_height}}{3}$: First row boundary.

$\frac{2 \cdot \text{img_height}}{3}$: Second row boundary.

```
1 def get_location_category(x1, y1, x2, y2, img_width, img_height):
2     center_x = (x1 + x2) / 2
3     center_y = (y1 + y2) / 2
4     if center_x < img_width / 3:
5         if center_y < img_height / 3:
6             return "top-left"
7         elif center_y < 2 * img_height / 3:
8             return "left"
9         else:
10            return "bottom-left"
11    elif center_x < 2 * img_width / 3:
12        if center_y < img_height / 3:
13            return "up"
14        elif center_y < 2 * img_height / 3:
15            return "center"
16        else:
17            return "down"
18    else:
19        if center_y < img_height / 3:
20            return "top-right"
21        elif center_y < 2 * img_height / 3:
22            return "right"
23        else:
24            return "bottom-right"
```

Figure 3.11: Location category based on coordinates

The function *get_location_category* takes the bounding box coordinates and the image dimensions as inputs. It uses the following logic to categorize the location in nine equal sections.

Column	Formula	Spatial location
Left Column	$center_x < \frac{img_width}{3}$	
Top	$center_y < \frac{img_height}{3}$	top-left
Middle	$\frac{img_height}{3} \leq center_y < \frac{2 \cdot img_height}{3}$	left
Bottom	$center_y \geq \frac{2 \cdot img_height}{3}$	bottom-left
Middle Column	$\frac{img_width}{3} \leq center_x < \frac{2 \cdot img_width}{3}$	
Top	$center_y < \frac{img_height}{3}$	up
Middle	$\frac{img_height}{3} \leq center_y < \frac{2 \cdot img_height}{3}$	center
Bottom	$center_y \geq \frac{2 \cdot img_height}{3}$	down
Right Column	$center_x \geq \frac{2 \cdot img_width}{3}$	
Top	$center_y < \frac{img_height}{3}$	top-right
Middle	$\frac{img_height}{3} \leq center_y < \frac{2 \cdot img_height}{3}$	right
Bottom	$center_y \geq \frac{2 \cdot img_height}{3}$	bottom-right

Table 3.1: Categorizing the location.

top-left	up	top-right
left	center	right
bottom-left	down	bottom-right

Table 3.2: Visualization of the 3x3 grid

3.8 Phase 4

3.8.1 Text to Speech

The process of providing audio feedback for detected objects involves aggregating information such as their names, numbers and locations, converting this data into spoken feedback using a text-to-speech engine. This step enhances user interaction by allowing them to receive immediate, audible updates about the objects detected in a scene.

```

1 # initializing the text-to-speech engine
2 engine = pyttsx3.init()

```

Figure 3.12: initializing text-to-speech engine

The results contains the detection results from the YOLO model, *labels* contain the detected class indices and *coords* contains the normalized coordinates (x1, y1, x2, y2) and confidence

scores for the bounding boxes. *Defaultdict* is a specialized dictionary from the collections module. It initializes entries with a default structure with keys **count** and **locations**. **count** keeps track of the total number of each type of detected object. **locations** uses another *defaultdict* to count occurrences of each object in specific spatial categories. The function *provide_audio_feedback* encapsulates the entire process in order to generating and delivering audio feedback.

```
1 def provide_audio_feedback(results, img_width, img_height):
2     labels = results[0].boxes.cls
3     coords = results[0].boxes.xyxy
4     detected_objects = defaultdict(lambda: -'count': 0, 'locations':
5     defaultdict(int) )
6
7     for i in range(len(labels)):
8         label = model.names[int(labels[i])]
9         x1, y1, x2, y2 = int(coords[i][0]), int(coords[i][1]), int(coords[i]
10        ][2]), int(coords[i][3])
11         location_category = get_location_category(x1, y1, x2, y2, img_width
12        , img_height)
13         detected_objects[label]['count'] += 1
14         detected_objects[label]['locations'][location_category] += 1
```

Figure 3.13: Code Snippets Through a Loop

Regarding the function "def provide_audio_feedback (results)" a looping procedure implemented for each detected object updating it's counts and spatial location by extract its label and bounding box coordinates and converts the normalized coordinates to pixel values by multiplying by the image dimensions. It also employs the label index to get the actual class name from the YOLO model names array.

get_location_category calls the function to determine the spatial category of the object. Increment the count for the object's label and the count for its location in the *detected_objects dictionary*.

3.8.2 Generating the Audio Feedback

```
1 if detected_objects:
2     feedback_parts = []
3     for label, info in detected_objects.items():
4         count = info['count']
5         if count == 1:
6             location = next(iter(info['locations']))
7             feedback_parts.append(f"one {label} at the {location} ")
8         else:
9             locations = [f"{loc_count} {label} at the {loc} " for loc, loc_count in
10 info['locations'].items()]
11             feedback_parts.append(f"{count} {label} s: " + ", ".join(
12 locations))
13             feedback = "I see: " + ", ".join(feedback_parts)
14 else:
15     feedback = "No objects detected."
```

Figure 3.14: Feedback Generation

By inspecting the *detected_objects dictionary*, algorithm checks if any objects were detected, Creating a list to store parts of the feedback message.

finally it Joins all parts into a single feedback message for example : "I see one object at the location".

If no objects are detected, the feedback message set to "No objects detected".

3.8.3 Delivering the Audio Feedback

```
1 engine.say(feedback)
2 engine.runAndWait()
```

Figure 3.15: Delivering the Audio Feedback

Audio feedback will be provided to the user immediately after processing. The output will be represented on the console as well. The engine will activate audio feedback to announce the result, "first specifying the total number of classes, the ones with more detection, separated each with a matching location, and then the objects with less detection". The feedback will be stored in MP3 or any other supported format.

3.8.4 Language Modifying

gTTS will allow the Algorithm to perform on additional languages to have compatibility with various users. This process impacted future work focusing on the virtual assistant shape pro-

gram.

```
1 # Translating audio to Italian
2 translated_feedback = translator.translate(feedback, src='en', dest='it').
   text
3
4 # Print translated feedback
5 print(translated_feedback)
6
7 # modify language to supported ones in google translate : de,fr etc
8 tts = gTTS(translated_feedback, lang='it')
9 tts.save('audio_feedback.mp3')
10 audio = AudioSegment.from_mp3('audio_feedback.mp3')
11 play(audio)
12 os.remove('audio_feedback.mp3')
```

Figure 3.16: Language Modifying Script

By Modifying the *dest* string to accessible languages from Google Translate, the feedback will also provide the represented language. Figure 3.16 demonstrated feedback in console in Italian language.

3.9 Speech Recognition

A semi-virtual assistant program has been incorporated into the system to improve user experience and engagement. This speech recognition algorithm will specifically address the streaming functionality, particularly the webcam feature. Upon activation, the algorithm will initiate by asking, "How can I help you?" and then remain in a listening mood for the user's command. Once the user issues the command, "What do you see?" the webcam will be activated, allowing it to provide feedback on any detected objects.

```

1
2 # Recognizing voice commands
3 def recognize_command():
4     recognizer = sr.Recognizer()
5     with sr.Microphone() as source:
6         print("Listening...")
7         audio = recognizer.listen(source)
8
9     try:
10        command = recognizer.recognize_google(audio)
11        print(f"User said: -command ")
12        return command.lower()
13    except sr.UnknownValueError:
14        engine.say("Sorry, I did not understand that.")
15        engine.runAndWait()
16        return None
17
18 # manage the interaction
19 def main():
20     cap = cv2.VideoCapture(0) # Open the webcam
21     if not cap.isOpened():
22         print("Error: Could not open video stream.")
23         return
24
25     while True:
26         ret, frame = cap.read()
27         if not ret:
28             break
29
30         # Recognize voice commands
31         command = recognize_command()
32         if command:
33             if "what do you see" in command:
34                 detect_and_announce(frame)
35             elif "exit" in command or "quit" in command:
36                 engine.say("Exiting the program.")
37                 engine.runAndWait()
38                 break
39             else:
40                 engine.say("Command not recognized. Please try again.")
41                 engine.runAndWait()
42
43     cap.release()
44
45 if __name__ == "__main__":
46     main()

```

Figure 3.17: Speech recognition code snippets

On the other hand, the user can input the command "capture image", and the algorithm will utilize the webcam to capture an image, process it, and generate audio feedback. This functionality is currently limited to these features. However, plans are in place to expand its capabilities by integrating two buttons into the hardware. The first button would work when the user holds it and request the algorithm to describe the environment in real-time. The other button would act as a second choice, allowing user to press it, capture the environment, and

then process it by the algorithm to provide feedback. This approach, which involves integrating hardware and camera sensors, aims to offer extensive assistance for blind individuals.

Chapter 4

Results and Discussion

Current chapter comprehensively overviews the findings obtained through the employed methodologies. Subsequent sections outline the fundamental aspects of system development and performance evaluation regarding all the phases undertaken during the project.

4.1 System Implementation

Processing was carried out on a Dell Inspiron 14 laptop with an Iris Xe onboard graphic, a Core i7 CPU, and 16 GB RAM. For the training, Google Colab Pro was used with various epochs and an image size of 640 pixels, which is needed for YOLOv8. It took around 4 hours, and the runtime L4 GPU with 22.5 GB was used. Different hyperparameters and YOLOv8 sizes (nano, small, medium, large, and extra large) were tested to achieve the best balance between speed and accuracy. Ultimately, the YOLOv8 (Medium) model was chosen to balance accuracy and speed. This version had a normal number of parameters, and the Floating Point Operations Per Second (FLOPS) to be supported by the system. The system was tested on webcam as a stream scenario, images and videos through different formats and various states. To test the algorithm, the webcam of the laptop was employed. Google Colab does not easily support the webcam due to privacy concerns. Therefore, the model was analyzed on JupyterLab regarding all the steps.

4.2 Experimental Setup

4.2.1 Framework

After training, the model was exported to PyTorch framework to test the weight on unseen data. PyTorch framework has been Utilized to load and interact with the YOLO. It is worth mentioning that after YOLOv5, the framework has changed from Darknet to PyTorch. PyTorch, developed by Ultralytics[30], a popular DL framework. This version was not an official continuation by the original YOLO authors but became widely adopted due to its ease of use, extensive documentation, and active community support. The PyTorch implementation facilitated integration, training, and deployment in various Machine Learning (ML) and DL environments.

4.2.2 Dataset limitation

The total number of datasets after augmentation was almost 13000, split to train 80%, test 10%, and validation sets 10%.

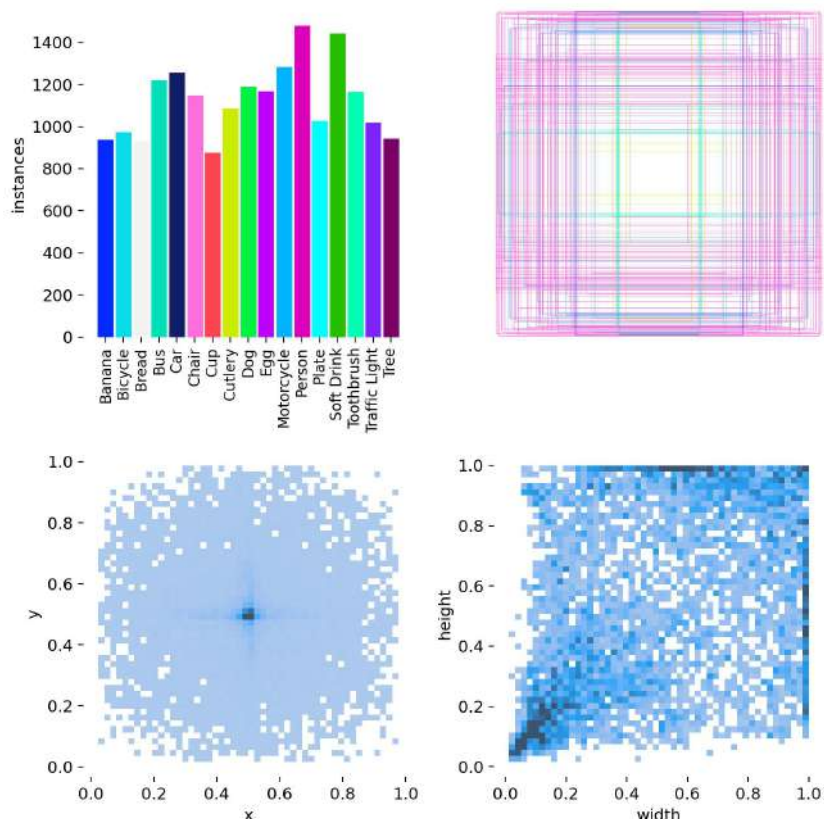


Figure 4.1: Class labels

Regarding the resource datasets and their limitations to the specific type, was due to having

both results on *indoor and outdoor* navigation such as stepping around at home and going out for shopping and taking care of the obstacles. The system’s performance in diverse scenarios demonstrates its potential to enhance autonomy and quality of life for blind users significantly. This practical algorithm offers safer and more efficient navigation capabilities.

4.3 Performance Metrics and Evaluation Criteria

During the training process, YOLOv8 allows the model to be validated by monitoring the result to see overfitting and underfitting signs. The Training process was developed with outstanding metrics. Regarding mAP 89% Precision and recall 84%, 86% respectively.

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
all	348	547	0.839	0.715	0.791	0.613
Banana	27	35	0.968	0.876	0.975	0.781
Bicycle	32	33	0.908	0.898	0.925	0.857
Bread	18	21	0.826	0.714	0.735	0.517
Bus	37	41	1	0.869	0.958	0.788
Car	20	43	0.435	0.581	0.525	0.347
Chair	26	26	0.949	0.923	0.98	0.83
Cup	16	20	0.867	0.8	0.891	0.696
Cutlery	12	27	0.955	0.815	0.931	0.66
Dog	18	29	0.7	0.483	0.644	0.474
Egg	17	32	0.894	0.719	0.803	0.711
Motorcycle	43	52	0.856	0.769	0.839	0.682
Person	29	43	0.611	0.302	0.377	0.211
Plate	15	27	0.707	0.593	0.524	0.377
Soft Drink	25	47	0.975	0.681	0.88	0.639
Toothbrush	14	23	0.915	0.783	0.843	0.61
Traffic Light	18	27	0.815	0.593	0.759	0.498
Tree	21	21	0.889	0.762	0.857	0.75

Speed: 0.1ms preprocess, 5.6ms inference, 0.0ms loss, 2.9ms postprocess per image
Results saved to runs/detect/exp

Figure 4.2: Training log

The system’s performance was evaluated using particular metrics.

4.3.1 Accuracy

Detection Accuracy: To evaluate detection accuracy, two main plots were used: the Confusion Matrix and the F1-score. The confusion matrix helps assess object prediction performance by comparing true (actual) and predicted classes. The X-axis represents the true object classes, while the Y-axis represents the predicted object classes by the model.

Upon reviewing the plot, it is evident that the most confident detections are for the classes Tree, Soft drink, and Bus, the value is 1.00 (very high). This suggests the model correctly classified an object labeled with high confidence while the least confident detection is related to the class Person with value 0.50. This difference is influenced by the diversity in the dataset related to class person, varying from small to large sizes and close to far distance. During detection, most instances relate to the class cutlery, which had numerous labels and annotations during the preprocessing step.

Overall, all objects demonstrate satisfactory performance, with only a few background detections for each. This false detection can be attributed to not annotating all objects in an image and not specifically training the model on background characteristics.

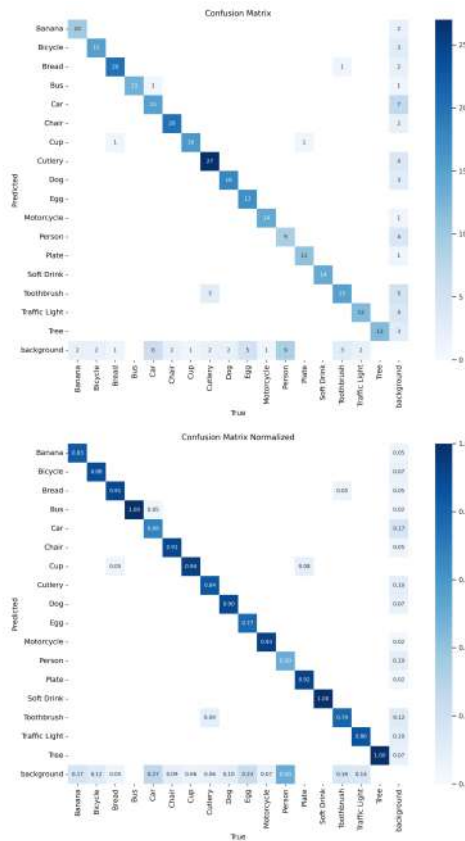


Figure 4.3: Confusion matrix

The F1-Curve which is the mean of precision and recall defining by two axes: The x-axis represents the confidence threshold from 0 to 1. The y-axis represents the F1 score, in a same range. The mentioned curve obtained the result at 0.85 regarding confidence in accuracy that all objects meet. Moreover, a combination of precision and recall is presented in detail to analyze them separately.

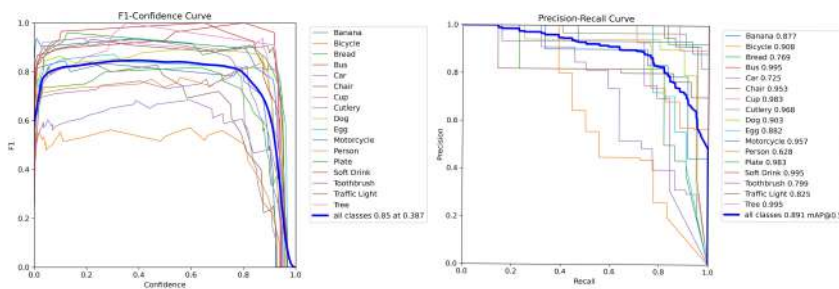


Figure 4.4: F1-Score and PR-Curve

Localization Accuracy: Refers to the precision of spatial position descriptions, which was quantitatively measured to assess the correctness of the described spatial positions. Lo-

calization accuracy is a vital measure of an object detection model performance. It specifies how accurately the detected bounding box corresponds to the ground truth box. IoU is the main metric that will delve into this accuracy by measuring the overlap between the predicted bounding box and the actual bounding box.

4.3.2 The Loss

The loss function for object detection typically considers:

Localization Loss: refers to a metric that quantifies the disparity between the predicted bounding boxes and the actual ones in object detection and localization tasks. By comparing the predicted and actual bounding boxes, the localization loss provides crucial feedback for the model to improve its ability to precisely locate objects.

Classification Loss: It calculates the difference between the predicted probabilities of each class and the true class labels. It quantifies how well the predicted probabilities align with the actual class labels, measuring the model’s performance in classifying the input data.

Confidence Loss: Measures how confident the model is about the presence of an object within the predicted bounding boxes over the epochs, All these three Losses were decreased with growing the epochs on the other hand validation Losses related to the ones mentioned for training , were decreased as well to stable the model and not showing any Overfitting sign.

Overall detection performance was quantified using mAP.

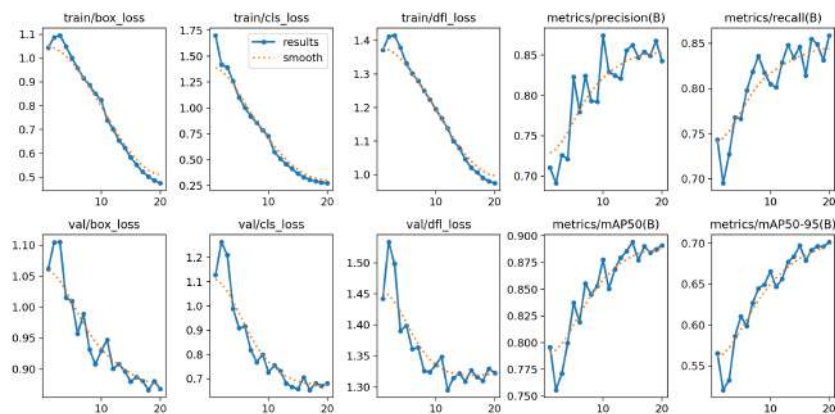


Figure 4.5: Training result

the model named *“feedback”* obtained After training and has been validated on unseen data in order to ensure that it is not only learning especial pattern and generalize it on real world data.

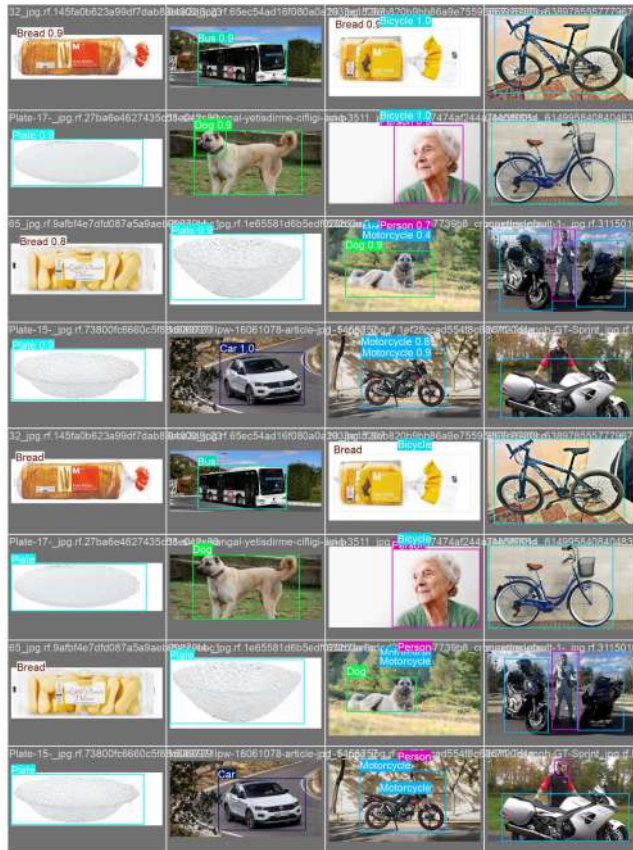


Figure 4.6: Validation label and Validation Prediction

4.3.3 Speed

Regarding the model's speed, we decided to go with the YOLOv8 medium size, which boasts an impressive speed of 2.26ms. It is essential to note that prioritizing speed over accuracy isn't always the best approach, especially when opting for larger models, as this could increase parameters. Despite this, the model delivers a strong performance with a reasonable average total processing time per image. The speed considerations include both the inference and post-processing techniques as well, techniques such as NMS

4.4 Analyzing of Model Performance

4.4.1 Object Detection

The first step regarding testing the model was implementing it on unseen images to analyze how it works.



Figure 4.7: Detection result

the model evaluates its predicted bounding boxes by comparing them with the true boxes using IoU to measure how accurately the model has located the objects. Higher IoU scores indicate more accurate localization. After IoU determined, YOLO will use NMS as a post processing technique to eliminate duplicate bounding boxes for the same object, keeping only the most confident predictions. The workload of IoU, NMS and Confidence threshold could be significant, especially in blind assistance scenarios where providing feedback could become challenging due to the presence of many objects. For this reason, an IoU of 0.7 was determined to safely detect ordinary amount of objects.



Figure 4.8: Model first performance

In Figure 4.8, there are two buses with a confidence threshold above 0.90 and two cars with different confidence thresholds, one 0.89 and another 0.39.

First, by adjusting this parameter to 0.90, the results in Figure 4.9 will demonstrate that one object in class Car and another in class Bus are wholly removed from the detection. The other parameter was set to 0.33 This will keep the ones above this ratio, removing the car with a confidence threshold of about 0.32.



Figure 4.9: Adjusting parameters

Moreover by indicating IoU very high it will detect some extra detections out of each object which NMS will discard the ones with lowest confidence.

However, the model also makes some errors regarding detection. For example, the figure 4.10 illustrates that the model incorrectly predicted a cup as a soft drink because of the

similarity between the bounding boxes.



Figure 4.10: Evaluation detection accuracy

Ignoring these few false detections about the model showed significant results in both speed and accuracy throughout the whole procedure, such as streaming, image processing, and video analysis. Regarding the video and real-time application, the challenging part is detecting frame by frame to implement it with the best impact on blind people in terms of accuracy and speed. Furthermore, it was investigated in different lighting environments.

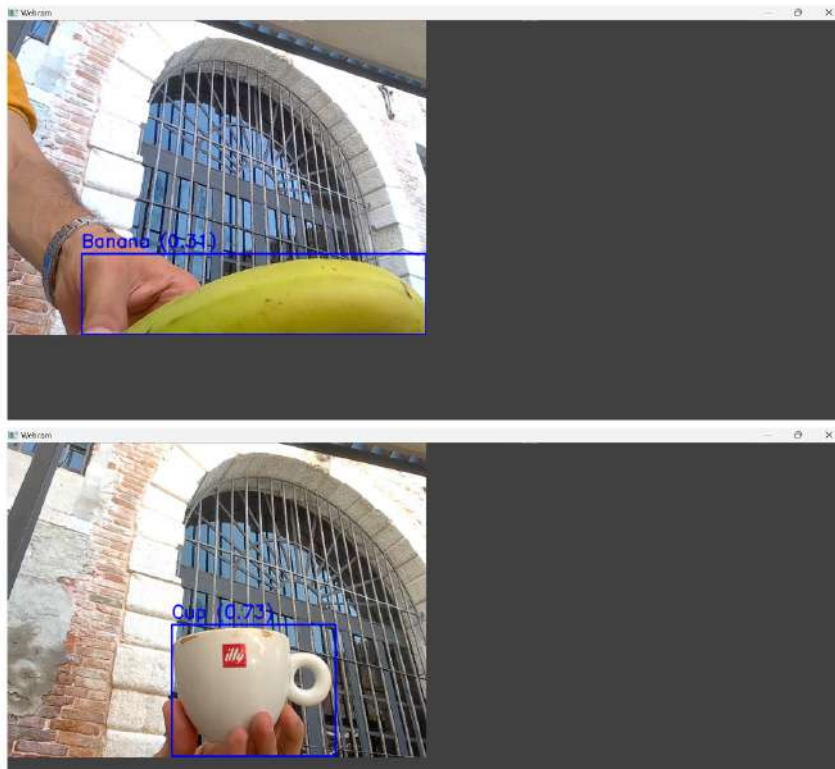


Figure 4.11: Real time example

4.4.2 Object's Spatial Location

Due to the lack of availability of the actual sizes and sensors, such as the camera, The best approach was, instead of the distance, having feedback about the spatial location of the objects. The image is divided into nine equal sections in all directions regarding spatial location. The algorithm accuracy was displaying significant results out of detection in any direction.

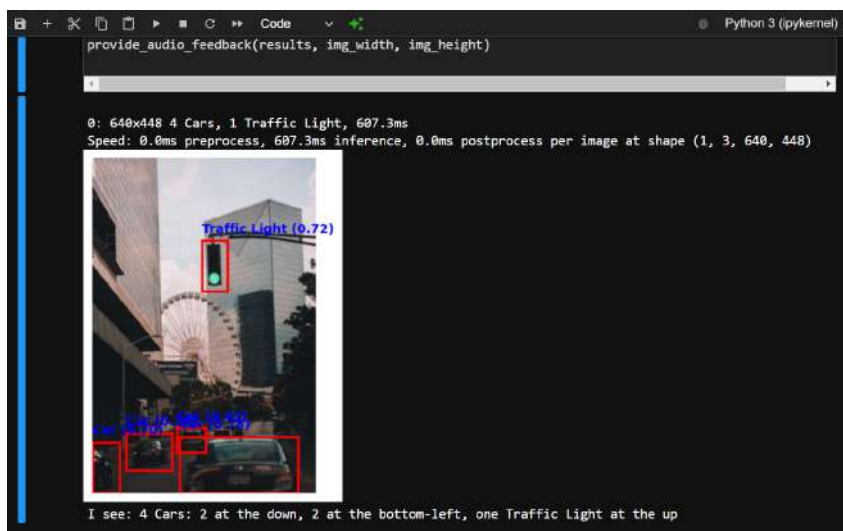


Figure 4.12: Practical result

Figure 4.12 represents a practical result of all the procedures, especially spatial location,

which is printed on the console beside the audio playback. It printed the spatial location of each detected object related to the class Car, describing them at the down, bottom left, and traffic light at the up, that are entirely correct.

4.5 Audio feedback analysing

In the case of an image, the practical result is providing feedback after capturing and processing the data. For video sources, the video is analyzed frame by frame, available for adjusting to detect objects in each frame, storing the result separately, and providing feedback during analysis. In a real-time application, the webcam processes the environment as a live stream and sends feedback frame by frame.

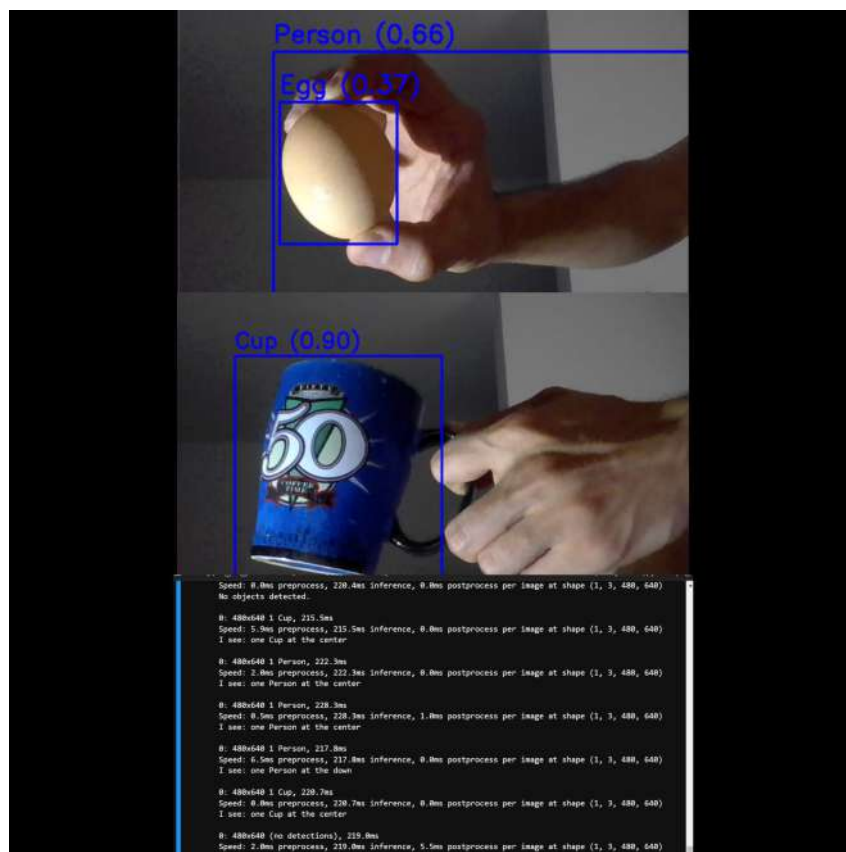


Figure 4.13: Printing audio on console

In order to have the application in different languages, implementing on gTTS was considered integrating with a Google translate library to translate simply to the other languages, enhancing user experience. Figure 4.14, corresponds to the practical result out of language modification in Italian. It is reasonable to point out that in all the processes, the response time of the audio was almost immediate except for the video part, which was slightly challenging.

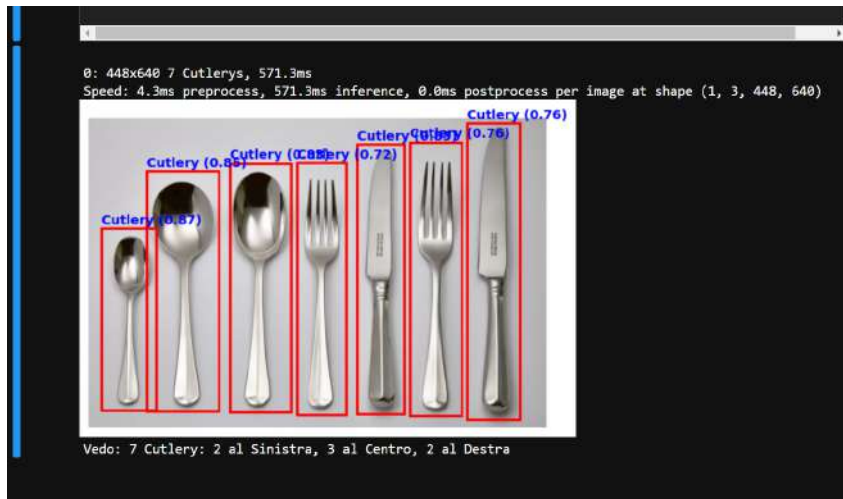


Figure 4.14: Italian Language

4.6 A Prototype of Virtual Assistant

as a prototype for virtual assistant, a speech recognition algorithm separately was studied to detect objects. This feature will allow users to interact with a semi-virtual assistant through a few talking procedures. This algorithm is focused on working only on a webcam, which can be the primary procedure and satisfactory in all tasks. However, future work would focus on an entire virtual assistance program. represents a practical result of the it. the algorithm started by saying to the user, "How can I help you?" Then, it remains in listening mode to acquire the user's command. In this example, after the user provides the command "Hello," the algorithm will deliver feedback as "Command not recognized. Please try again." Finally, the correct command was implemented, as well as the result.

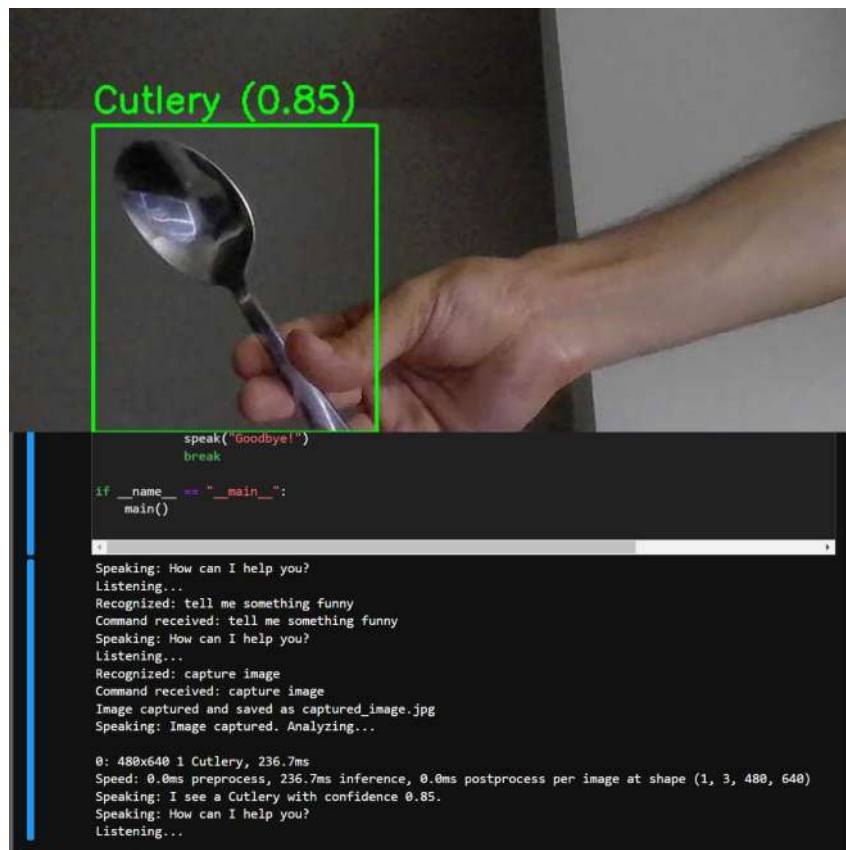


Figure 4.15: Speech recognition practical result

In the Figure 4.16, Another interaction occurred between the user and the algorithm. In this case, instead of capturing, the algorithm would describe all the objects that can be seen and provide feedback. This form of analysis will improve real-time applications.

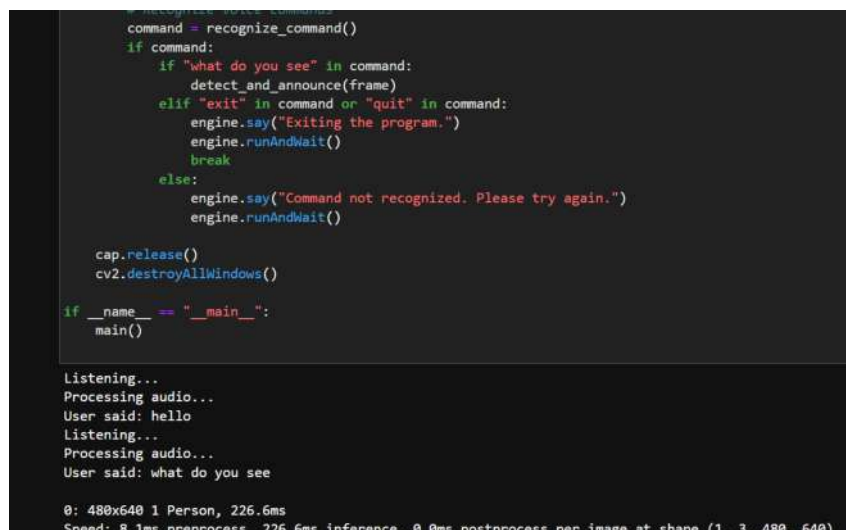


Figure 4.16: Speech recognition another practical result

Chapter 5

Limitations and Future Work

Due to the lack of accessibility to hardware such as Arduino or Raspberry Pi, the current program remained theoretical-based. On the other hand, future developments will focus on improving the detection algorithm with better metrics regarding the accuracy and speed involved in implementing it on hardware. Besides the improvements of the current algorithm, text recognition using the OCR and a color sensor can be used for a wide range of tasks. The concept for future hardware involves eyeglasses with attached hardware, including a color sensor and a microphone, to provide feedback. The proposed wearable system would have two buttons: one dedicated to capturing and sending processed information and another for real-time recording and feedback. The hardware would be compact and user-friendly, and the power supply would be a rechargeable battery. Furthermore, the audio feedback will be enhanced to provide a form of virtual assistance, offering users a more interactive, supportive experience. During the composition of this thesis, newer versions of the YOLO algorithm, YOLOv9[31] and YOLOv10[32], were announced. Regrettably, YOLOv9 was not easily accessible and did not readily accommodate audio guidance. Shortly after the completion of the thesis, YOLOv10 was released. It is anticipated that these impressive versions with outstanding transformations in YOLO object detection era, will be used in upcoming attempts.

Chapter 6

Conclusions

This research successfully implemented a real-time object detection and localization system using the YOLOv8 model, specifically tailored to assist blind individuals. The system was fine-tuned on a custom dataset using a pre-trained YOLOv8 (Medium) size, Achieving a supreme mAP of 89%. The system was tested using a standard webcam along with images and videos to facilitate practical applications, simulating real-world conditions where dedicated hardware was unavailable. The object detection component identified objects and determined their spatial positions and the number of objects, providing descriptive feedback. This positional information was converted into auditory feedback, offering users immediate and intuitive spatial awareness. Future work would focus on testing with dedicated hardware for enhanced performance and exploring additional sensory modalities to provide more comprehensive environmental awareness.

Acronym

CNN *Convolutional Neural Network*

WHO World Health Organization

CV Computer Vision

AI Artificial Intelligence

ML Machine Learning

DL Deep Learning

YOLO You Only Look Once

mAP Mean Average Precision

AP Average Precision

OCR Optical character recognition

gTTS Google Text-To-Speech

IoU Intersection Over Union

SOTA State-Of-The-Art

FLOPS Floating Point Operations Per Second

NMS Non Maximum Suppression

RCNN Region-based Convolutional Neural Network

MCI Management Center Innsbruck

RFID Radio Frequency Identification Devices

YAML Yet Another Markup Language

SGD Stochastic Gradient Descent

FLOPS Floating Point Operations Per Second

CLI command line interface

Bibliography

- [1] . Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
- [2] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716, November 2023.
- [3] Saravanan Alagarsamy, T. Dhiliphan Rajkumar, K. P. L. Syamala, Ch. Sandya Niharika, D. Usha Rani, and K. Balaji. A real-time object detection method for visually impaired using machine learning. In *Proceedings of the 2023 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6, 2023.
- [4] Martin Eckert, Matthias Blex, and Christoph Friedrich. Object detection featuring 3d audio localization for microsoft hololens: A deep learning based sensor substitution approach for the blind. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, pages 555–561, January 2018.
- [5] Dr. Boobalan, Bhuvanikha S, Sivapriya M, and Sivakumar R. Object detection with voice guidance to assist visually impaired using yolov7. *International Journal for Research in Applied Science and Engineering Technology*, 11(4):764–768, April 2023.
- [6] Armin Masoumian, David G.F. Marei, Saddam Abdulwahab, Julián Cristiano, Domenech Puig, and Hatem A. Rashwan. *Absolute Distance Prediction Based on Deep Learning Object Detection and Monocular Depth Estimation Models*. IOS Press, October 2021.
- [7] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

- [8] GBD 2019 Blindness, Vision Impairment Collaborators, and Vision Loss Expert Group of the Global Burden of Disease Study. Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to vision 2020: the right to sight: an analysis for the global burden of disease study. *Lancet Global Health*, 9(2):e144–e160, Feb 2021.
- [9] T. R. Fricke, N. Tahhan, S. Resnikoff, E. Papas, A. Burnett, M. H. Suit, T. Naduvilath, and K. Naidoo. Global prevalence of presbyopia and vision impairment from uncorrected presbyopia: Systematic review, meta-analysis, and modelling. *Ophthalmology*, May 2018.
- [10] J. Bernal, K. Kushibar, D. S. Asfaw, S. Valverde, A. Oliver, R. Martí, and X. Lladó. Deep convolutional neural networks for brain image analysis on magnetic resonance imaging: A review. *Artificial Intelligence in Medicine*, 95:64–81, 2019.
- [11] M. Kashiha, C. Bahr, S. Ott, C. P. Moons, T. A. Niewold, F. Ödberg, and D. Berckmans. Automatic identification of marked pigs in a pen using image pattern recognition. *Computers and Electronics in Agriculture*, 93:111–120, 2013.
- [12] . Heimberger, J. Horgan, C. Hughes, J. McDonald, and S. Yogamani. Computer vision in automated parking systems: Design, implementation and challenges. *Image and Vision Computing*, 68:88–101, 2017. Automotive Vision: Challenges, Trends, Technologies and Systems for Vision-Based Intelligent Vehicles.
- [13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.
- [14] S. Surendarkumar, M. Ranjithkumar, B. Deepanraj, M. Sivashankar, and M. Umaphathy. Blind people assistance for object detection using ai. *Journal of Computer Science and Engineering*, 2020. Computer Science and Engineering, The Kavery Engineering College, Mecheri, Tamilnadu, India.
- [15] Pradnya Kasture, Akshay Tangade, Aditya Pole, Aishwarya Kumkar, and Yash Jagtap. Real-time object detection using yolo algorithm for blind people. *International Journal of Advanced Research in Science, Communication and Technology*, 2021.
- [16] Pranjali Deshmukh, Ajinkya Khedkar, Shubham Kulkarni, and Shriram Morkhandikar. Object detection for blind people using yolov3. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 11(5):498–504, May 2023.

- [17] Juan Du. Understanding of object detection based on cnn family and yolo. *Journal of Physics: Conference Series*, 1004(1):012029, apr 2018.
- [18] Geethapriya. S, N. Duraimurugan, and S. P. Chokkalingam. Real-time object detection with yolo. *International Journal of Engineering and Advanced Technology (IJEAT)*, 8(3S):2249–8958, February 2019. ISSN: 2249-8958.
- [19] Ferdousi Rahman, Israt Jahan Ritun, Nafisa Farhin, and Jia Uddin. An assistive model for visually impaired people using yolo and mtcnn. In *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy, ICCSP '19*, pages 225–230, New York, NY, USA, 2019. Association for Computing Machinery.
- [20] Rajeshvaree Karmarkar and Vikas Honmane. Object detection system for the blind with voice guidance. *International Journal of Engineering Applied Sciences and Technology*, 6(2), June 2021.
- [21] Mansi Mahendru and Sanjay Kumar Dubey. Real time object detection with audio feedback using yolo vs. yolo_v3. In *Proceedings of the 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 734–740, 2021.
- [22] Simranjeet Kaur, Anup Lal Yadav, and Abhishek Joshi. Real time object detection. In *Proceedings of the 2022 International Conference on Cyber Resilience (ICCR)*, pages 1–5, 2022.
- [23] Venkata Mandhala, Debnath Bhattacharyya, Vamsi Bandi, and N. Thirupathi Rao. Object detection using machine learning for visually impaired people. *International Journal of Current Research and Review*, 12(2):157–167, January 2020.
- [24] Raihan Bin Islam, Samiha Akhter, Faria Iqbal, Md. Saif Ur Rahman, and Riasat Khan. Deep learning based object detection and surrounding environment description for visually impaired people. *Heliyon*, 9(6):e16924, 2023.
- [25] Nada N. Saeed, Mohammed A.-M. Salem, and Alaa Khamis. Android-based object recognition for the visually impaired. In *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 645–648, 2013.
- [26] Rahul Kumar, Sanjesh Kumar, Sunil Lal, and Praneel Chand. Object detection and recognition for a pick and place robot. In *Proceedings of the 2014 International Conference on Computer Vision and Robotics*, November 2014.

- [27] Samkit Shah. Cnn based auto-assistance system as a boon for directing visually impaired person. In *3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1–5. IEEE, 2019.
- [28] Jack M. Loomis. Digital map and navigation system for the visually impaired. Technical report, Department of Psychology, University of California, Santa Barbara, 1985.
- [29] B. Dwyer, J. Nelson, T. Hansen, et al. Roboflow (version 1.0) [software], 2024. Computer vision software.
- [30] Glenn Jocher. Ultralytics yolov5, 2020.
- [31] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information, 2024.
- [32] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection, 2024.
- [33] F. Liu, Z. Lu, and X. Lin. Vision-based environmental perception for autonomous driving. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 0(0), 2023.
- [34] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:886–893 vol. 1, 2005.
- [35] Matthew Blaschko and Christoph Lampert. Learning to localize objects with structured output regression. In *Proceedings of the European Conference on Computer Vision (ECCV) 2008*, volume 5302 of *Lecture Notes in Computer Science*, pages 2–15, Berlin, Heidelberg, October 2008. Springer.
- [36] Marek Vajgl, Petr Hurtik, and Tomáš Nejezchleba. Dist-yolo: Fast object detection with distance estimation. *Applied Sciences*, 12(3), 2022.

