

الجمهورية الجزائرية الديمقراطية الشعبية
PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY OF MAY 8 TH , 1945 - GUELMA -
FACULTY OF MATHEMATICS, COMPUTER SCIENCE AND MATERIAL SCIENCES

Computer Science Department



Master's Thesis

Sector: IT

Option : STIC

Theme

**Object Detection for Impaired Visual Assistance Using Transfer Learning
and IoT-Raspberry Pi**

presented by:

ADEM HAMICI

N	Full Name	Quality
1	Dr. Halimi Khaled	Chairman
2	Dr. Hallaci Samir	Supervisor
3	Dr. Benhamida Nadjette	Examiner

Acknowledgment

Thank you, Allah, for providing me the capacity to think and write, the courage to hold into my convictions, and the perseverance to see my dreams through to completion.

I would like to express my profound gratitude to **Dr. Samir Hallaci**, my supervisor, for overseeing the completion of this task and for providing me with the best of his knowledge and assistance.

I also want to express my gratitude to the jury members, **Dr. Halimi Khaled** and **Dr. Benhamida Nadjette**, for giving me the privilege of judging this work.

I want to express my gratitude to all of the IT department teachers especially **Abderrahmane Kefali** , **Seridi Hamid** , **Kouahla Zineddine** , **Seridi ali** and **Hiba Abdelmoumène** and everyone else who helped to create this work.

Finally, I would like to thank my family , my Teachers and my Friends.

Dedication

We express our deepest gratitude to **Almighty Allah** for granting us the strength and determination to complete this work.

To my beloved mother, **Fadia Zalani** No words can truly capture the depth of my respect, my endless love, and my gratitude for the countless sacrifices you've made for my education and well-being. Your unwavering support and boundless love have been with me every step of the way since my childhood. I pray that your blessings continue to guide me throughout my life. May this humble work be a realization of your hopes and dreams, the result of your endless sacrifices. May God bless you with good health, happiness, and a long life

To my dearest father, **Mourad** You have been more than just a father to me—you have been a guiding light, a true friend, and a constant source of wisdom throughout my life. Your prayers have been my strength during this long and challenging journey. Words cannot express how much your unwavering support and love mean to me. I pray that God, Almighty, keeps you in good health and blesses you with happiness and a long life, so that you may continue to be the light that illuminates my path.

To my dear sister **Lina**, and to my cherished family members **Mariam, Rabah, Ines, Sirin, Wassim, Samer, Yamina, Karim, Rafif, Mouhamed, Samir, Ilham, Fehd, Ania, and Miral** Words are scarcely enough to express the deep connection, love, and affection I have for each of you. Your presence in my life has been a source of immense strength and joy. I wish you all a life filled with happiness, success, and fulfillment.

A special tribute goes to my **grandfather**, who was like a father to me and a pillar of support in my life. May God have mercy on his soul and grant him eternal peace. His memory will always remain a guiding light in my heart.

Special thanks to my friend **Ahmed Rami Bouguettoucha**, whose constant encouragement and support have been invaluable to me. I wish him even greater success in all his endeavors. To my dear friends **Ahmed, minou, djihed , mouhamed, and Abd El Basset** the memories of our laughter, shared moments, and the experiences we've had together are priceless. I sincerely hope that our friendship endures and continues to bring joy and warmth to our lives for years to come.

I would like to extend my heartfelt gratitude to all the members of the Consulate General of Algeria in Naples. Your unwavering support, camaraderie, and collaboration have made our time together both productive and memorable. Working alongside each of you has been an honor, and I am deeply thankful for the shared experiences, professional growth, and mutual respect that have marked our journey. I dedicate this work to all of you, with the hope that our paths continue to cross and that our collective efforts bring continued success to our mission.

Abstract :

Navigating outdoor environments poses substantial challenges for blind and visually impaired people, limiting their ability to move independently and safely. This thesis presents a novel AI-based system designed to enhance mobility for visually impaired users by providing real-time object detection and depth sensing. Utilizing deep learning techniques and the YOLOv8 object detection algorithm, the system is implemented on embedded systems with the Raspberry Pi 4 and integrated with a 3D camera to assess the spatial proximity of detected objects.

The custom WOTR (walk on the road) dataset developed for this project, tailored to the needs of visually impaired individuals, ensures high accuracy in object detection and depth estimation. The system delivers real-time audio feedback, offering practical guidance for non-controlled outdoor assistive navigation.

Comprehensive testing in various outdoor settings demonstrates the system's effectiveness in detecting objects, estimating their depth, and providing timely feedback. The portability and cost-effectiveness of the Raspberry Pi 4 make this solution accessible to a wide audience, potentially improving the quality of life for visually impaired individuals by enabling safer and more confident navigation. This work advances the field of assistive technologies, offering a practical tool that empowers blind and visually impaired individuals to navigate outdoor spaces with greater ease and independence.

Keywords : Deep learning, object detection, YOLOv8, embedded systems, 3D Camera, blind and visually impaired people, Non Controlled outdoor assistive navigation.

Résumé :

La navigation dans les environnements extérieurs pose des défis considérables aux personnes aveugles et malvoyantes, limitant leur capacité à se déplacer de manière indépendante et en toute sécurité. Cette thèse présente un nouveau système basé sur l'IA conçu pour améliorer la mobilité des utilisateurs malvoyants en fournissant une détection d'objet et une détection de profondeur en temps réel. Utilisant des techniques d'apprentissage profond et l'algorithme de détection d'objets YOLOv8, le système est implémenté sur des systèmes embarqués avec le Raspberry Pi 4 et intégré avec une caméra 3D pour évaluer la proximité spatiale des objets détectés.

La base de données personnalisé "WOTR (walk on the road)" développé pour ce projet, adapté aux besoins des personnes malvoyantes, garantit une grande précision dans la détection des objets et l'estimation de la profondeur. Le système fournit un retour audio en temps réel, offrant des conseils pratiques pour la navigation assistée non contrôlée en extérieur.

Des essais approfondis dans divers environnements extérieurs démontrent l'efficacité du système dans la détection d'objets, l'estimation de leur profondeur et la fourniture d'un retour d'information en temps utile. La portabilité et la rentabilité du Raspberry Pi 4 rendent cette solution accessible à un large public, ce qui pourrait améliorer la qualité de vie des personnes malvoyantes en leur permettant de naviguer de manière plus sûre et plus confiante. Ce travail fait progresser le domaine des technologies d'assistance, en offrant un outil pratique qui permet aux personnes aveugles et malvoyantes de naviguer dans les espaces extérieurs avec plus de facilité et d'indépendance.

Mots clés : Apprentissage profond, détection d'objets, YOLOv8, systèmes embarqués, caméra 3D, personnes aveugles et malvoyantes, navigation extérieure assistée non contrôlée

الخلاصة

يفرض التنقل في البيئات الخارجية تحديات كبيرة على المكفوفين وضعاف البصر، مما يحد من قدرتهم على التنقل بشكل مستقل وأمن. تقدم هذه الأطروحة نظاماً جديداً قائماً على الذكاء الاصطناعي مصمماً لتعزيز تنقل المستخدمين المكفوفين وضعاف البصر من خلال توفير اكتشاف الأجسام واستشعار العمق في الوقت الفعلي. وباستخدام تقنيات التعلم العميق وخوارزمية YOLOv8 للكشف عن الأجسام، يتم تنفيذ النظام على أنظمة مدججة مع Raspberry Pi 4 ومتكملاً مع كاميرا ثلاثية الأبعاد لتقييم القرب المكاني للأجسام المكتشفة.

تضمن مجموعة البيانات المخصصة WOTR (walk on the road) التي تم تطويرها لهذا المشروع، والمصممة خصيصاً لتلبية احتياجات الأفراد ضعاف البصر، دقة عالية في اكتشاف الأجسام وتقدير العمق. يقدم النظام ملاحظات صوتية في الوقت الفعلي، مما يوفر إرشادات عملية للملاحة المساعدة في الهواء الطلق بدون تحكم.

يُظهر الاختبار الشامل في مختلف البيئات الخارجية فعالية النظام في اكتشاف الأجسام وتقدير عمقها وتقديم التغذية الراجعة في الوقت المناسب. إن إمكانية نقل جهاز Raspberry Pi 4 وفعاليته من حيث التكلفة يجعل هذا الحل في متناول جمهور واسع، مما قد يحسن من جودة حياة الأفراد ضعاف البصر من خلال تمكين التنقل بأمان وثقة أكبر. يعمل هذا العمل على تطوير مجال التقنيات المساعدة، حيث يقدم أداة عملية تمكّن المكفوفين وضعاف البصر من التنقل في الأماكن الخارجية بسهولة واستقلالية أكبر.

الكلمات الدالة : التعلم العميق، واكتشاف الأجسام، وـ YOLOv8 ، وأنظمة المدججة، والكاميرا ثلاثية الأبعاد، والمكفوفين وضعاف البصر، والملاحة المساعدة الخارجية غير المتحكم بها.

Contents

List of Figures	iii
List of Tables	vi
General introduction	1
1 Introduction to AI-Based Solutions for Visual Impairments	4
1.1 Introduction	4
1.2 Human Vision System vs Computer Vision systems	4
1.2.1 Anatomy and Function of the Human Eye	4
1.2.2 Computer/machine Vision	7
1.3 Challenges in Navigating Life with Visual Impairments	8
1.3.1 The different types of visual impairments	8
1.3.2 The difficulties that people with visual impairments face in their daily lives	8
1.4 Classic Solutions for Helping People with Visual Impairments	9
1.4.1 The benefits and limitations of each of these solutions	12
1.5 AI Innovations for Assisting Individuals with Visual Impairments	13
1.6 Related works	14
1.7 AI and Deep Learning	17
1.7.1 Artificial intelligence	17
1.7.2 Machine learning	18
1.7.3 Deep learning	19
1.8 Datasets(Benchmarks)	25
1.8.1 MS COCO dataset :	25
1.8.2 PASCAL VOC dataset :	26
1.8.3 ImageNet dataset	27
1.8.4 Open Images dataset	28
1.9 Conclusion	29

2 Real-Time Object Detection with Deep Learning	30
2.1 Introduction	30
2.2 Real-Time Object Detection	30
2.2.1 Accuracy	32
2.2.2 Frames per second (Fps)	32
2.3 How to choose the best model for object detection system ?	32
2.4 Object Detection Models	33
2.4.1 EfficientDet	33
2.4.2 MobileNetV2	34
2.4.3 Faster R-CNN (Region Convolutional Neural Network)	34
2.4.4 SSD (Single Shot MultiBox Detector)	34
2.4.5 RetinaNet	34
2.4.6 YOLO (You Only Look Once)	35
2.5 Conclusion	43
3 Conception	44
3.1 Introduction	44
3.2 What makes YOLO a better choice for Object Detection ?	44
3.3 Global architecture	45
3.4 Data Acquisition	46
3.5 Data preparation	47
3.5.1 Data collection	47
3.5.2 Data preprocessing	49
3.5.3 challenges faced in data preparation	50
3.6 YOLOv8 architeture	51
3.6.1 What are the main features in YOLOv8?	52
3.7 Model configuration	54
3.7.1 choose the best model	54
3.7.2 Augmentation	55
3.8 Train model on our data	56
3.8.1 Hyperparameter Choices to Train YOLOv8	56
3.8.2 Loss Function	57
3.8.3 training procedure	58
3.8.4 Challenges Faced During Training	59

3.9	metrics	60
3.10	IOT module	61
3.10.1	Sensors	61
3.11	Conclusion	62
4	Implementation and Results	63
4.1	Introduction	63
4.2	Development environment	63
4.2.1	Hardware Environment	63
4.2.2	Software environment	65
4.3	Overview of the Assistive Navigation System for the Visually Impaired	67
4.3.1	Overview	67
4.3.2	User Interface (UI)	67
4.3.3	Object Detection	68
4.3.4	Alert System	68
4.3.5	Camera and Data Handling	69
4.3.6	Detailed Object Interactions and Alerts	69
4.4	Training and validation	74
4.4.1	training results :	74
4.5	Test, Results and Discussion	78
4.5.1	Test and Results	78
4.5.2	Discussion	80
4.6	Conclusion	81
General conclusion		81
bibliography		

List of Figures

1.1	human eye anatomy [1]	5
1.2	The Pathway of Visual Processing	6
1.3	Human Vision vs Computer Vision	7
1.4	Braille Writing System	9
1.5	Navigating with a White Cane	10
1.6	guide dog	10
1.7	Magnifiers aids [2]	10
1.8	assistive technology aids [3]	11
1.9	Human assistance[3]	11
1.10	Different subdomain AI, ML, ANN and DL [4]	17
1.11	machine learning models and their training algorithms [5]	18
1.12	Human learning VS Machine learning [6]	19
1.13	The architecture of ANN.[7]	20
1.14	The architecture of DNN.[8]	20
1.15	The architecture of CNN.[9]	21
1.16	The architecture of RNN.[10]	21
1.17	The architecture of GAN.	22
1.18	The architecture of GAN.[11]	22
1.19	The architecture of RL.[12]	23
1.20	The architecture of Transfer Learning.[13]	23
1.21	The architecture of Transfer Learning.	24
1.22	The architecture of Transfer Learning.[13]	24
1.23	Difference between classification, detection and segmentation .[14]	25
1.24	Example of images of MS-COCO [15]	26
1.25	Example of annotation of MS-COCO	26
1.26	The most popular models used for various tasks on the MSCOCO	26

1.27	Example of annotation of PASCAL VOC	27
1.28	The most popular models used for various tasks on the PASCAL VOC	27
1.29	Example of images of ImageNet	28
1.30	The most popular models used for various tasks on the open Image	28
1.31	Example of images of Open Image	28
1.32	The most popular models used for various tasks on the Open Image	29
2.1	Difference Between 60FPS And 24FPS[16]	32
2.2	YOLO model architecture [17]	35
2.3	Divide the image into (S^*S) grid.	35
2.4	The predicted vector in the case of a single box.	36
2.5	Intersection over union.	36
2.6	Examples of IoU.	37
2.7	The predicted vector in the case of multiple boxes in the cell.	37
2.8	A tensor that specifies the bounding box	37
2.9	The output after different steps of NMS [18]	38
2.10	YOLOv2 architecture[19]	39
2.11	YOLOv3 Architecture	39
2.12	YOLOv4 Architecture [20]	40
2.13	YOLOv5 Architecture [21]	40
2.14	YOLOv6 Architecture [22]	41
2.15	YOLOv7 Architecture [23]	42
2.16	YOLOv8 Architecture [24]	42
2.17	YOLO NAS Architecture [25]	43
3.1	Comparing YOLOv8 to Other YOLO Models: A Comparative Analysis.[26]	45
3.2	Global architecture	46
3.3	RGB Image VS Depth Image	47
3.4	Object categories in the WOTR dataset	48
3.5	Some Pictures form Pothole dataset	49
3.6	Data Preprocessing	49
3.7	WOTR format conversion to WOTR-Yolo Format	50
3.8	YOLOv8 Architecture.[26]	52
3.9	Visualization of an anchor box	53

3.10	New YOLOv8 C2f module [27]	53
3.11	YOLOv8 COCO evaluation[28]	54
3.12	training procedure	59
3.13	System architecture (IOT)	61
4.1	Google Colab Logo	64
4.2	RPI4 logo	65
4.3	Python logo	65
4.4	PyTorch logo	66
4.5	OpenCv logo	66
4.6	PyCharm logo	66
4.7	Case one	72
4.8	Case tree	73
4.9	Case four	73
4.10	Depth and RGB	74
4.11	Train batch	74
4.12	validation batches	75
4.13	Results	76
4.14	Confusion matrix	77
4.15	Confusion Matrix Normalized	78

List of Tables

1.1	Comparison of Solutions for Visually Impaired Individuals	12
1.2	Comparison of different computer vision systems for assisting visually impaired individuals	17
1.3	Differences between Artificial Intelligence and Machine Learning	19
3.1	WOTR dataset statistics	48
3.2	YOLOv8 Training Configuration.[29]	57
3.3	Sensors for the IoT System	61
4.1	Case two	72
4.2	Results of training	77
4.3	Inference times and FLOPs for different model weights on CPU and GPU	79

General Introduction

Navigating outdoor environments poses significant challenges for visually impaired individuals, impacting their ability to interact with and perceive their surroundings safely. These challenges necessitate the development of innovative solutions that can enhance the independence and mobility of those affected. Recent advancements in deep learning, particularly in real-time object detection, offer promising avenues for addressing these challenges. In this thesis, we present a solution that leverages the YOLOv8 object detection algorithm and the Raspberry Pi 4, enhanced by depth-sensing capabilities, to provide real-time audio feedback to visually impaired users.

Our system is designed not only to detect objects and obstacles but also to accurately assess their depth, offering a critical advantage in guiding users through complex environments. By integrating depth information, the system provides more nuanced feedback, allowing users to understand the relative distance of objects, which is vital for safe navigation. This feedback is delivered in real-time through audio cues.

A significant contribution of this thesis is the development of a custom dataset tailored to the specific needs of visually impaired individuals navigating outdoor spaces. This dataset, which incorporates subclasses from various benchmarks, was used to train and fine-tune the YOLOv8 model, ensuring high accuracy and reliability in object detection and depth estimation. The system is further enhanced by the integration of Depth sensors, which collectively contribute to its functionality and usability.

The system has been rigorously tested across different outdoor environments to evaluate its performance. The results demonstrate that the system can reliably detect objects, estimate their depth, and provide timely audio feedback, making it a practical tool for enhancing the mobility of visually impaired individuals. Additionally, the system's portability, enabled by the Raspberry Pi 4 platform, ensure that it can be widely accessible to those in need.

This thesis offers a comprehensive and innovative solution to the challenge of assisting visually impaired individuals in navigating outdoor environments. By combining advanced object detection with depth sensing and real-time feedback, the system significantly improves the user's ability to move independently and confidently. The successful integration of these technologies into an accessible and portable system underscores the potential to positively impact the lives of visually impaired individuals, empowering them to navigate outdoor spaces with greater ease and safety.

our thesis is decomposed to four chapters:

- **Chapter 1: Introduction to AI-Based Solutions for Visual Impairments**

introduces AI-based solutions for visual impairments by comparing the human vision system with AI-

driven computer vision. It discusses the challenges faced by visually impaired individuals and reviews traditional aids, highlighting their limitations. The chapter then focuses on how AI, particularly deep learning, offers more effective solutions. It also explains key AI concepts and reviews relevant datasets, setting up the groundwork for developing a custom dataset for YOLOv8 model training.

- **Chapter 2: Real-Time Object Detection with Deep Learning**

This section delves into real-time object detection, underscoring its importance in computer vision and its role in AI-based solutions for visual impairments. It reviews various object detection models, including EfficientDet, MobileNetV2, Faster R-CNN, SSD, RetinaNet, and YOLO, focusing on their strengths and suitability for real-time use. The chapter highlights how deep learning has improved object detection accuracy and speed, discusses criteria for model selection, and emphasizes the need for real-time performance in assistive technologies. This analysis leads to a detailed examination of YOLOv8, the key algorithm in the solution.

- **Chapter 3: YOLOv8 Model Architecture and Training**

This section provides an in-depth overview of the YOLOv8 model, focusing on its architecture, configuration, and training process for the specific application. It details model parameters, loss functions, and the integration of sensors like depth sensors to enhance system functionality. The chapter discusses the challenges encountered during training, emphasizing the importance of custom datasets and how different model configurations affect performance. The training process and results are thoroughly analyzed to demonstrate YOLOv8's effectiveness in real-time object detection and its contribution to the overall system.

- **Chapter 4: Development Environment and Implementation**

This section explores the development environment and implementation of the project, detailing the hardware and software tools used, such as the Raspberry Pi 4 and various sensors. It outlines the design and implementation steps, addressing development challenges and the solutions applied. The chapter emphasizes the integration of hardware and software components, focusing on creating a functional, user-friendly system. It highlights the project's outcomes, demonstrating the system's ability to deliver accurate object detection and real-time feedback in outdoor environments.

We conclude with a general summary and discussion of future perspectives.

Chapter 1

Introduction to AI-Based Solutions for Visual Impairments

1.1 Introduction

Visual impairments affect millions of people worldwide creating difficulties, in recognizing faces , understanding text and navigating. While traditional solutions like aids and specialized software exist to address these challenges their effectiveness is limited. As a result the rise of AI powered tools has attracted attention for their potential to better support individuals with impairments.

This section aims to give an overview of these solutions. It starts by comparing vision with machine vision capabilities and discussing the obstacles faced by those with impairments. It then explores interventions and highlights the benefits offered by AI technologies. Furthermore it delves into how 3D technology combined with AI can expand the support for people with impairments. The focus is on explaining the basics of AI learning and its crucial role in improving accessibility solutions. Additionally it examines known datasets used for model training and assessment well, as popular deep learning frameworks. Finally this section concludes by summarizing points and providing a glimpse into the topics that future chapters will address.

1.2 Human Vision System vs Computer Vision systems

1.2.1 Anatomy and Function of the Human Eye

Understanding the anatomy and functioning of the human eye is crucial to creating effective treatments for visual impairments, as it plays a major role in our capacity to comprehend the world around us. This section will provide an overview of the eye's anatomy and function, detailing its various components and how they work together to process light and transmit nerve impulses to the brain.

The structure of the eye

The eye is a sensitive and intricate organ with a unique anatomical and physiological structure [30], which can be divided into three main layers:

- **Outer Layer:**

- The **cornea** is a transparent, approximately spherical structure with an outer radius of curvature of about 8 mm.
- The **sclera** is a dense, white, opaque fibrous tissue, also nearly spherical, with a radius of curvature of about 12 mm.[31]

- **Middle Layer (Uveal Tract):**

- The **iris** plays a crucial optical role by controlling the size of its aperture.
- The **ciliary body** is essential for the process of accommodation, enabling the eye to change focus.
- The **choroid** is located at the back of the eye and plays a role in providing the eye's blood supply. [31]

- **Inner Layer:**

- The **retina** is an extension of the central nervous system, connected to the brain via the optic nerve.[31]

Lens and Chambers: The lens, located about 3 mm inside the eye, is attached to the ciliary body by suspensory ligaments known as zonules.

The inside of the eye is divided into three compartments:

- The **anterior chamber**, located between the cornea and iris, contains aqueous humor.
- The **posterior chamber**, situated between the iris, ciliary body, and lens, also contains aqueous humor.
- The **vitreous chamber**, found between the lens and retina, is filled with a transparent gel called vitreous humor.[31]

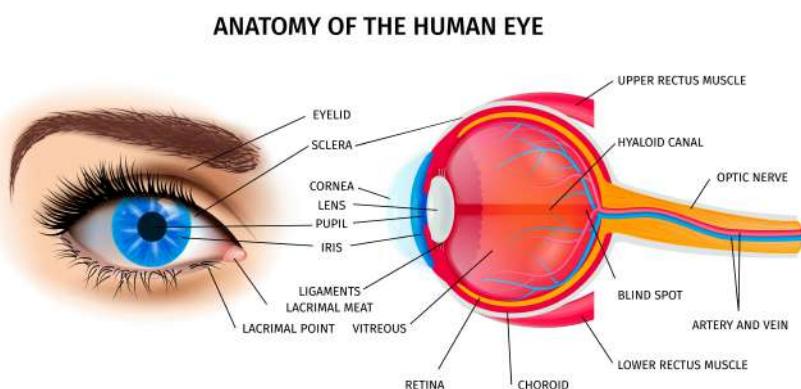


Figure 1.1: human eye anatomy [1]

The pathway of vision

When light enters the eye, it triggers chemical changes in certain pigments within the photoreceptor cells of the retina, leading to the production of electrical impulses . Before passing onto ganglion cells, these impulses, which contain visual information, pass through a network of linked neurons in the retina. The optic nerve, which is made up of these ganglion cells, sends visual information to the occipital lobe of the brain's visual cortex. These impulses are processed by complex neuronal circuits in the visual cortex, which combine different visual clues to create a coherent image of the object being watched. The ability to see and interpret visual inputs is made possible by this intricate processing, which offers a thorough awareness of the world [32].

Because of the intricate biological design of human vision , the two eyes each have a separate field of view. Since each eye records data from half of the visual field, the brain integrates the pictures seamlessly. When this process is completed, one develops binocular vision, which enables depth perception and enhances spatial awareness by overlapping the central regions of the visual field. However, the marvels of human vision extend even further . Humans are able to see the world in three dimensions due in large part to differences in the periphery of each eye's visual field. This skill is necessary for navigating our surroundings, determining distances properly, and engaging with the things in our immediate environment. Our ability to perceive depth allows us to approximate sizes , which facilitates the execution of simple activities like picking up a cup from a table or more complicated ones like maneuvering a car safely through traffic. Humans sense depth through the intricate interaction of several visual signals, which adds levels of complexity and richness to our perception of the environment [33] (figure 1.2).

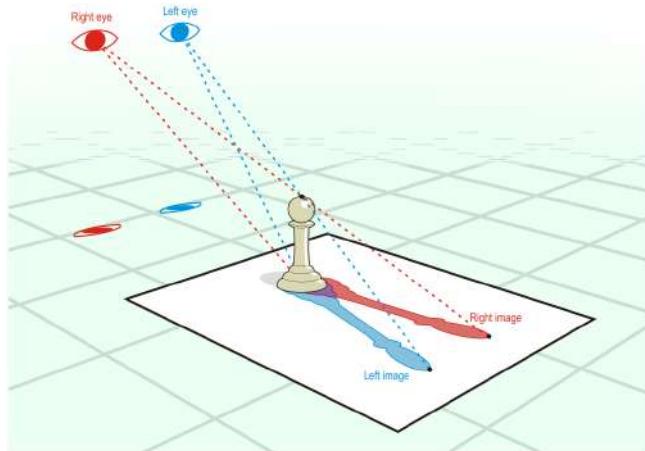


Figure 1.2: The Pathway of Visual Processing

Capabilities of the human visual system

- Interprets and understands complex visual scenes in real-time.
- Swiftly processes and comprehends intricate visual environments.
- Identifies a wide range of visual attributes (hues, forms, patterns, motions) with precision.
- Adapts to varying lighting, distances, and angles for consistent perception.
- Recognizes familiar objects and faces, even with partial obstruction or altered viewpoints.

- Handles demanding tasks like reading and symbol recognition with ease.
- Perceives depth, enabling 3D understanding of objects and environments.

1.2.2 Computer/machine Vision

Computer Vision, also known as Machine Vision, is a discipline that lets machines learn how to "see" and is an important application field of deep learning technology, which is widely used in security, industrial quality inspection and automatic driving scenarios. Specifically, the goal is to enable the machine to identify objects in images or videos captured by the camera, detect the object's location, and track the target object , so as to understand and describe the scene and story in the picture or video, in order to simulate the human brain visual system. Therefore, computer vision is also commonly referred to as machine vision, and the goal is to build artificial systems that can "sense" information from images or videos.[34] The development of computer vision begins with biological vision.

Limitations of the visual machine

In contrast to the human visual system (Figure 1.3), a visual machine operates within predefined limits based on the data it has been trained on . Should this data be skewed or incomplete, the machine's performance in real-world circumstances may be damaged. Additionally, the machine's effectiveness is bound by the intricacy of the jobs it has been taught to accomplish. For instance, if the computer is simply taught to detect specified things in preset circumstances, it may fail to adapt to fresh situations or identify unknown objects.

Another restriction of the visual machine is its inability to participate in reasoning or contextual knowledge equivalent to human cognition. Despite its potential to distinguish things within a picture, the computer may stumble in recognizing the connections between these objects or interpreting the image's underlying meaning. Consequently, this constraint might induce mistakes or misinterpretations in specific settings.

Moreover, the visual machine is bound by the processing resources at its disposal. Tasks of increasing complexity or datasets of bigger size may need more powerful technology or lengthier training durations, consequently incurring heightened expenditures or logistical obstacles.

Despite its impressive powers, the visual machine is restricted by variables such as training settings, datasets, contextual understanding restrictions and computing constraints.

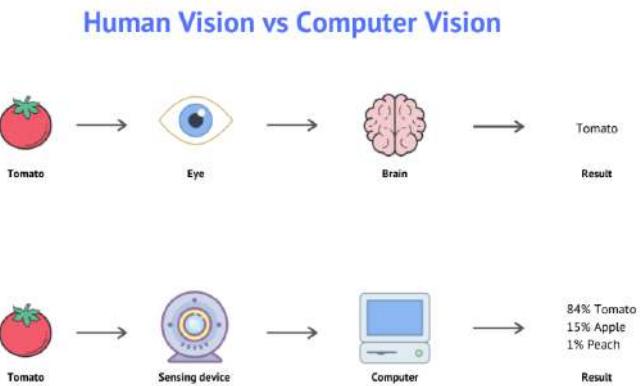


Figure 1.3: Human Vision vs Computer Vision

1.3 Challenges in Navigating Life with Visual Impairments

1.3.1 The different types of visual impairments

Visual impairment comprises different disorders affecting vision, ranging from minor to severe. Among the sorts of visual impairments [35] are:

- **Color Vision Deficiency:** commonly known as color blindness, hampers an individual's ability to accurately perceive or distinguish certain colors. The most prevalent types are red-green and blue-yellow color blindness.
- **Blindness:** The most severe kind, defined by full loss of vision in both eyes, due to reasons such as hereditary diseases, traumas, infections, or medical problems.
- **Low vision:** Significantly diminished vision without full blindness, resulting in difficulty with skills like reading or recognizing faces, commonly linked with illnesses such as cataracts, glaucoma, or age-related macular degeneration.
- **Night blindness (nyctalopia):** Difficulty seeing in low-light situations owing to variables like genetic abnormalities, hunger, or medical illnesses.
- **Visual processing disorders:** Conditions affecting the brain's interpretation of visual information, resulting in difficulty with identifying forms, letters, numbers, or depth perception.
- **Photophobia:** Sensitivity to strong light producing discomfort or agony.

These classes reflect a portion of the varied range of visual impairments, each having distinct manifestations and implications on people. It's interesting that experiences with visual impairment might vary greatly dependent on individual disorders and their severity.

1.3.2 The difficulties that people with visual impairments face in their daily lives

Individuals with visual impairments encounter a range of daily challenges, including :

1. **Navigational challenges :** Limited vision impairs people's ability to safely navigate their surroundings, making it difficult to go freely through streets, public areas, and new locations.
2. **Social interactions :** Visual limitations may make it difficult to perceive nonverbal communication signs including facial emotions and body language during social interactions. As a result, people may have difficulty sustaining social bonds and participating in social activities.
3. **Access to information :** Individuals with visual impairments have difficulties obtaining information via printed materials, computer displays, and other visual media.
4. **Employment opportunities :** Visual impairments may restrict employment chances by creating challenges to obtaining job-related documents, navigating working surroundings, and executing activities that need visual acuity.

5. **instructional barriers** : Visual impairments may make it difficult to access instructional materials, participate in classroom activities, and get equal educational opportunities.
6. **everyday living tasks** : Individuals with vision impairments may have substantial difficulty while doing basic everyday activities such as cooking, cleaning, and personal hygiene. To do these things safely and efficiently, you may need to use assistive technologies or ask for help from others.
7. **Emotional and psychological impact** : Dealing with the limits imposed by visual impairments may result in emotions of frustration, alienation, and poor self-esteem.
8. **Accessibility barriers** : Inaccessible infrastructure, transit systems, and public facilities make it more difficult for people with visual impairments to move about and participate in community life.

Despite the numerous challenges that individuals with visual impairments face in their daily lives, various solutions and support systems are available to help them overcome these obstacles and lead independent, fulfilling lives.

1.4 Classic Solutions for Helping People with Visual Impairments

Several traditional solutions have been developed to assist individuals with visual impairments in their daily lives

- **Braille** : is a tactile writing method used by the blind and visually handicapped. Braille characters represent letters, numerals, punctuation marks, and other symbols, enabling people to read by touch.

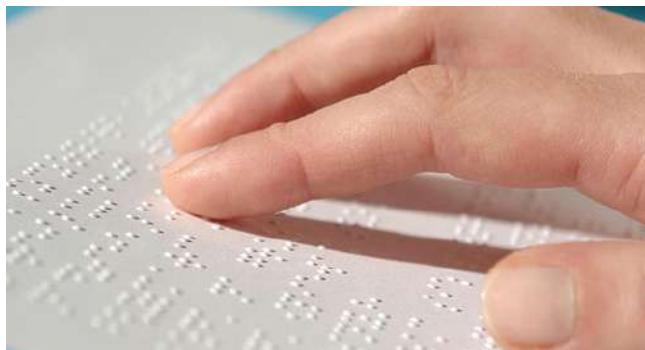


Figure 1.4: Braille Writing System

- **White Canes** : White canes help people with vision impairments discover hazards and navigate their environment safely. They give tactile feedback and indicate to others that bearer is visually impaired.



Figure 1.5: Navigating with a White Cane

- **Guide dogs :** Specially trained guide dogs assist individuals with visual impairments by guiding them around obstacles, stopping at barriers, and navigating crowded areas. These guide dogs enhance mobility and independence for their users .



Figure 1.6: guide dog

- **Magnifiers :** Magnifying devices such as handheld magnifiers, magnifying glasses, or electronic magnifiers enlarge text and images, making them easier to see for individuals with low vision.



Figure 1.7: Magnifiers aids [2]

- **assistive technology :** Assistive technology: Various assistive technologies, including screen magnification software, speech recognition software, Braille displays and more, help individuals with visual impairments perform tasks on computers, smartphones and other electronic devices.



Figure 1.8: assistive technology aids [3]

- **Family and friends:** friends and family may be a valuable source of support for those who are visually impaired. They may aid with everyday chores, transportation assistance, and emotional support. To support their loved ones who have vision impairments in leading independent lives, family and friends may also educate themselves on assistive technology and adaptive technologies.



Figure 1.9: Human assistance[3]

It is essential to acknowledge that these solutions include some limits and may not be universally appropriate. Furthermore, they may not cover every issue that affects those who are visually impaired. To create more sophisticated and individualized solutions, researchers have looked into emerging technologies like artificial intelligence (AI) and Computer Vision.

1.4.1 The benefits and limitations of each of these solutions

Solution	Benefits	Limitations
Braille	<ul style="list-style-type: none"> Enables independent reading and writing via touch. Facilitates access to written information. Provides standardized communication. 	<ul style="list-style-type: none"> Can be difficult and time-consuming to learn. Resources may not be widely available. May not suit those with cognitive or tactile limitations.
White Canes	<ul style="list-style-type: none"> Enhances safety by identifying obstacles. Provides haptic feedback for navigation. Raises public awareness of impairment. 	<ul style="list-style-type: none"> May miss small or overhead obstacles. Can cause discomfort due to stigma. Requires training and practice.
Guide Dogs	<ul style="list-style-type: none"> Increases mobility and freedom. Offers emotional support and companionship. Boosts confidence and social interaction. 	<ul style="list-style-type: none"> High cost of training and care. Not suitable for all individuals. Requires ongoing care and training.
Magnifiers	<ul style="list-style-type: none"> Enhances text and image readability. Portable and easy to use. Available in various formats. 	<ul style="list-style-type: none"> Limited field of view. May distort images when highly magnified. Less effective for severe impairments.
Assistive Technology	<ul style="list-style-type: none"> Improves access to digital devices. Increases independence in various tasks. Adaptable solutions for diverse needs. 	<ul style="list-style-type: none"> Can be costly. Requires technical skills and support. May not be compatible with all platforms.
Family and Friends	<ul style="list-style-type: none"> Provides emotional support. Assists with daily tasks and mobility. Raises awareness of individual's needs. 	<ul style="list-style-type: none"> May lead to dependency or strain. Support availability may vary. Lack of training may limit effectiveness.

Table 1.1: Comparison of Solutions for Visually Impaired Individuals

It is crucial to remember that not everyone with visual impairments would benefit from these traditional solutions, and that the advantages and disadvantages may change based on the particular solution and the requirements of the person.

1.5 AI Innovations for Assisting Individuals with Visual Impairments

A plethora of artificial intelligence (AI) solutions have surfaced to assist those who are visually impaired. These solutions use deep learning algorithms to evaluate visual input and provide customized feedback. These inventions consist of:

1. **Object Recognition Systems :** These systems employ deep learning and computer vision techniques to rapidly and accurately identify objects in real time. When combined with wearable technology, such as glasses or smartphones, they offer users audio cues about their surroundings. For instance, Microsoft's Seeing AI app utilizes object recognition to identify items, transcribe text, and even recognize faces .
2. **Navigation Systems :** AI-driven navigation systems help people who are blind or visually impaired navigate both indoor and outdoor environments. These systems deliver users aural cues when they discover impediments using computer vision and deep learning techniques. Most notably, the Aira app pairs smart glasses with trained agents to provide real-time instruction to blind and visually impaired users.
3. **Reading Assistance Systems :** These programs translate written text into audio or braille formats using optical character recognition (OCR) technology and deep learning algorithms. The KNFB Reader app is an example of this kind of technology; it can extract text from a variety of sources, such as books, menus, and signs.
4. **Social Interaction Systems :** Artificial intelligence (AI)-powered solutions help those who are visually impaired during social interactions. The Seeing AI app, for example, is very good at identifying faces and provides information about emotional cues based on facial expressions.
5. **Assistive Learning Systems :** By providing customized instructional material and adaptive learning opportunities, AI-driven learning systems meet the educational demands of those who are visually impaired. These systems use machine learning algorithms and natural language processing (NLP) to tailor online courses and textbooks to the unique requirements and learning preferences of visually impaired students. These technologies enable people with visual impairments to more successfully pursue chances for academic and professional growth by offering interactive and accessible learning materials. A system like this is the Bookshare platform, which uses AI algorithms to transform digital books into braille, big print, audio, and other accessible forms to meet the various learning needs of visually impaired users.

These artificial intelligence (AI) solutions have the potential to improve the independence and information accessibility of people who are visually impaired. They do have certain limits, however, including the need to purchase specialist technology and issues with object identification accuracy. However, it is anticipated that further research and development efforts in this field will improve existing solutions, providing the visually impaired community with more accurate and user-friendly assistance.

1.6 Related works

in table 1.2 we will present an overview of relevant research and details about each article related to assisting visually impaired individuals over the past decade :

Ref	Year	Title	Approach	Extra		Accuracy / FPS	Indoor / Out-door
				Sensors	Dataset		
[36]	2017	Computer Vision for the Visually Impaired: the Sound of Vision System	-3D reconstruction and segmentation of the environment - Detection of objects, ground plane, walls, doors, stairs, signs, text - Different processing pipelines for indoor and outdoor environments	Stereo RGB camera - Depth-of-Field camera (Structure Sensor) - Inertial Measurement Unit (IMU)	/	Ground Surface Detection: TPR 0.98, PPV 0.90, ACC 0.89 Obstacle Detection: TPR 0.97, PPV 0.78, ACC 0.76 Object width error: 0.13m Object height error: 0.17m Object center deviation: 16px	Both
[37]	2017	A Cloud and Vision-based Navigation System Used for Blind People	Vision-based SLAM for navigation	Stereo cameras Microphone Speaker	ImageNet dataset for object recognition Custom RMB (Chinese currency) dataset with 90,000 images for currency recognition Traffic light recognition dataset OCR datasets	99.9% accuracy reported for currency (RMB) recognition	Both

Ref	Year	Title	Approach	Extra		Accuracy / FPS	Indoor / Out-door
				Sensors	Dataset		
[38]	2017	Using Technology Developed for Autonomous Cars to Help Navigate Blind People	/	Lidars and cameras	/	/	Both
[39]	2019	An Object Detection Technique For Blind People in Real-Time Using Deep Neural Network	convolutional neural network with single shot multi-box detector (SSD) algorithm Combines Faster R-CNN with deep neural network and SSD algorithm with additional layers	Integrates an audio device to help blind people	Pascal VOC dataset COCO dataset (80 classes, 80,000 images and 40,000 validation images)	achieved 78.68 mAP	/

Ref	Year	Title	Approach	Extra		Accuracy / FPS	Indoor / Out-door
				Sensors	Dataset		
[40]	2021	Computer Vision-based Assistance System for the Visually Impaired Using Mobile Edge Artificial Intelligence	Neural Compute Stick-2, model optimization techniques (OpenVINO, TensorFlow Lite)	OpenCV AI Kit-Depth (OAK-D) sensor, GPS device	-Google Open Image dataset. -LISA traffic signs dataset. -German Traffic Sign Recognition Benchmark (GTSRB) dataset. -Traffic Cone dataset. -Cityscapes dataset. Custom. -dataset of images collected by the researchers	Object detection model (SSD-MobileNet): 0.62 mean average precision (mAP)	Both
[41]	2022	A Wearable Assistive Device for Blind Pedestrians Using Real-Time Object Detection and Tactile Presentation	YOLOV3 model compression for object detection	Shape-memory alloy (SMA) actuators, vibration motors	/	96% accuracy in obstacle position recognition	Both

Ref	Year	Title	Approach	Extra Sensors	Dataset	Accuracy / FPS	Indoor / Out-door
[42]	2023	Assistive Object Recognition and Obstacle Detection System for the Visually Impaired Using YOLO	YOLOv3	Raspberry Pi, Camera and HC SR-04 sensor	MSCOCO	45 fps and 28 mAP	Both

Table 1.2: Comparison of different computer vision systems for assisting visually impaired individuals

1.7 AI and Deep Learning

1.7.1 Artificial intelligence

AI is defined as the science, engineering, and technology focused on intelligent behavior, emulating human capabilities such as thinking, sensing, and reacting (Gherghina, 2015) [43]. Additionally, AI is characterized by a system's ability to accurately interpret external data, learn from it, and use these learnings to achieve specific goals and tasks through flexible adaptation (Kaplan & Haenlein, 2019) [44]. It is also viewed as a technology of machine information processing that simulates human cognitive activities (Popkova & Sergi, 2020) [45]. The ultimate goal of AI is to develop intelligent machines capable of perceiving, learning, reasoning, and interacting autonomously, driving advancements in various fields such as healthcare, finance, transportation, and more. [46] (figure 1.10).

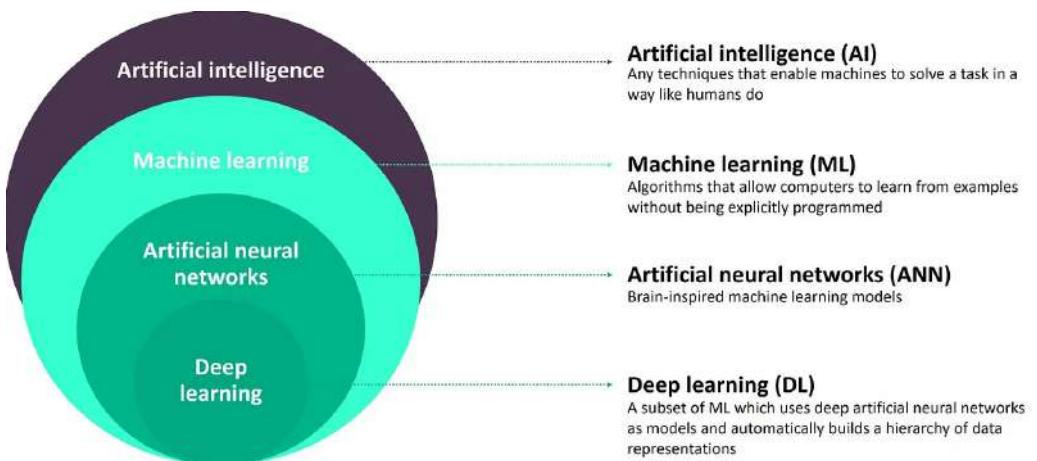


Figure 1.10: Different subdomain AI, ML, ANN and DL [4]

1.7.2 Machine learning

As a branch of artificial intelligence (AI), machine learning (ML) uses algorithms to evaluate data, draw conclusions from it, and make choices on their own without the need for explicit programming. These algorithms fall into two categories: supervised , unsupervised , semi-supervised and Reinforcement . Supervised models use previously learned information to analyze fresh datasets, whereas unsupervised models make deductions from the data. ML algorithms use statistical techniques to categorize and predict data in order to identify both linear and non-linear correlations within datasets. Notwithstanding obstacles including the need for mathematical proficiency, high-quality data, and comprehension of intricate algorithms, machine learning (ML) presents revolutionary possibilities for work automation and enhancing corporate processes.[46, 5]

Machine learning models and their training algorithms

Supervised learning	Unsupervised learning	Semi-supervised learning	Reinforcement learning
<p>Data scientists provide input, output and feedback to build model (as the definition).</p> <p>EXAMPLE ALGORITHMS:</p> <ul style="list-style-type: none"> ■ Linear regressions ■ Sales forecasting. ■ Risk assessment. <p>Support vector machines</p> <ul style="list-style-type: none"> ■ Image classification. ■ Financial performance comparison. <p>Decision trees</p> <ul style="list-style-type: none"> ■ Predictive analytics. ■ Pricing. 	<p>Use deep learning to arrive at conclusions and patterns through unlabeled training data.</p> <p>EXAMPLE ALGORITHMS:</p> <ul style="list-style-type: none"> ■ Apriori ■ Sales functions. ■ Word associations. ■ Searcher. <p>K-means clustering</p> <ul style="list-style-type: none"> ■ Performance monitoring. ■ Searcher intent. <p>Artificial neural networks</p> <ul style="list-style-type: none"> ■ Generate new, synthetic data. ■ Data mining and pattern recognition. 	<p>Builds a model through a mix of labeled and unlabeled data, a set of categories, suggestions and exampled labels.</p> <p>EXAMPLE ALGORITHMS:</p> <ul style="list-style-type: none"> ■ Generative adversarial networks ■ Audio and video manipulation. ■ Data creation. <p>Self-trained Naïve Bayes classifier</p> <ul style="list-style-type: none"> ■ Natural language processing. 	<p>Self-interpreting but based on a system of rewards and punishments learned through trial and error, seeking maximum reward.</p> <p>EXAMPLE ALGORITHMS:</p> <ul style="list-style-type: none"> ■ Q-learning ■ Policy creation. ■ Consumption reduction. <p>Model-based value estimation</p> <ul style="list-style-type: none"> ■ Linear tasks. ■ Estimating parameters.

Figure 1.11: machine learning models and their training algorithms [5]

Differences Between Artificial Intelligence and Machine Learning :

Artificial Intelligence	Machine Learning
AI enables a computer to mimic human intellect in order to resolve issues.	ML enables a computer to independently learn from historical data.
The objective is to create a clever system that is capable of handling challenging jobs.	The objective is to create machines that can learn from data in order to improve output accuracy.
We create machines that can do intricate jobs much like a human.	We use data to educate computers to carry out certain jobs and provide precise outcomes.
There are several uses for AI.	The range of applications for machine learning is restricted.
AI replicates human decision-making in a system by using technology.	To create predictive models, machine learning (ML) employs self-learning algorithms.
AI can handle any kind of data, including unstructured, semi-structured, and structured data.	Only organized and semi-structured data may be used by ML.
Decision trees and logic are used by AI systems to learn, reason, and self-correct.	When given fresh data, machine learning (ML) systems may self-correct based on statistical models.

Table 1.3: Differences between Artificial Intelligence and Machine Learning
[47]

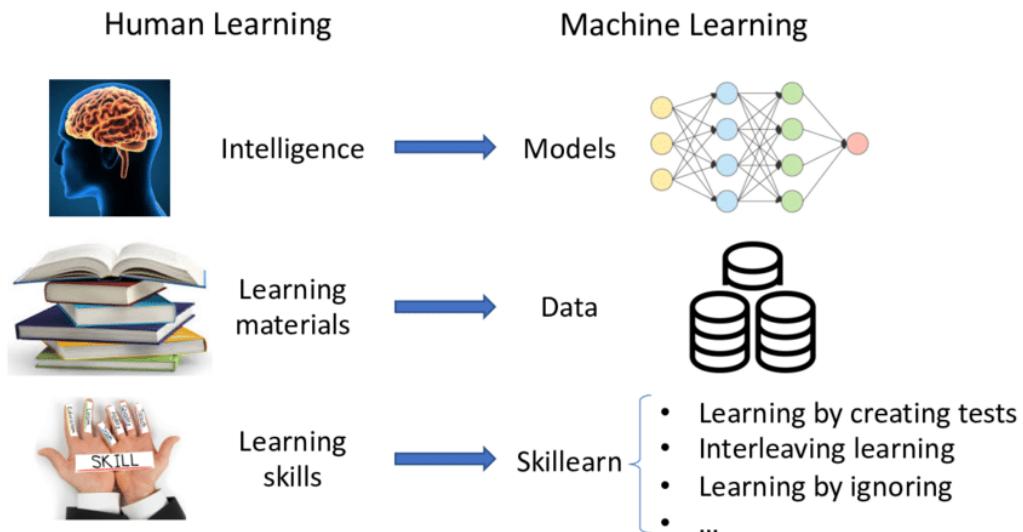


Figure 1.12: Human learning VS Machine learning [6]

1.7.3 Deep learning

Deep learning, a subset of artificial intelligence (AI), trains computers to analyze data in a way that mimics the human brain's processing . Deep learning models possess the capability to discern intricate patterns across various forms of data, including images, text, and audio, facilitating precise insights and predictions. This methodology enables the automation of tasks traditionally reliant on human intelligence, such as image description and speech-to-text transcription. [48]

Subdomain of Deep Learning

- **Artificial Neural Networks (ANNs)** : computer models that draw inspiration from the design and operation of organic neural networks seen in the human brain. Interconnected nodes, or neurons, arranged in layers—input, hidden, and output layers—make up artificial neural networks (ANNs). [9]

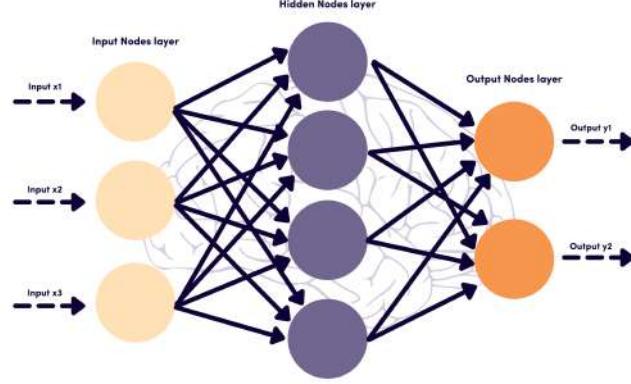


Figure 1.13: The architecture of ANN.[7]

- **Deep Neural Networks (DNNs)** : deep neural networks (DNNs) are a type of machine learning algorithms that try to replicate the way the brain processes information. Between the input and output layers of a DNN are several hidden layers (l). A certain number of units, or neurons, are present in each layer, which perform a specific functional change on the input. The universal approximation theorem states that these kinds of models may estimate the behavior of any function. Through a set of weights ($w_{i,k}$), a bias (b), and a non-linear activation function (f), the output (y) of a unit (i) in layer (l) is connected to the output (x) of the previous layer (k) with J outputs. [49]

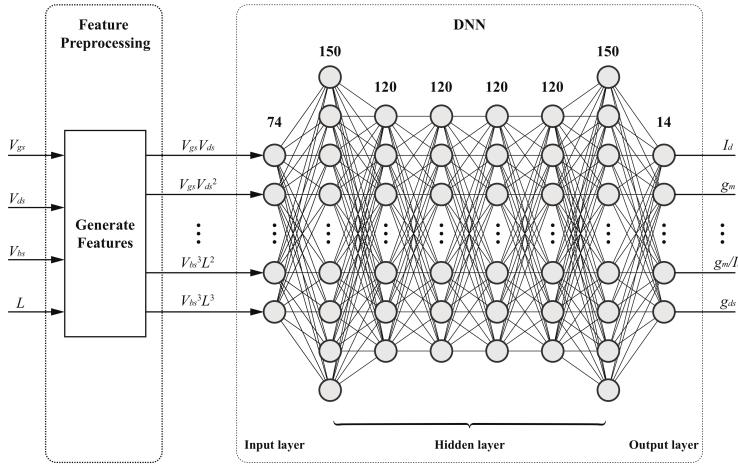


Figure 1.14: The architecture of DNN.[8]

- **Convolutional Neural Networks (CNNs)** : are a kind of artificial neural network intended to analyze time-series data, audio signals, pictures, and other data using a grid-like architecture. The network is able to extract high-level characteristics and patterns from raw inputs because of the convolutions, pooling, and non-linear transformations that are performed on the input data by its several layers of linked neurons.

Natural language processing, autonomous driving, picture and audio recognition, and other applications have all shown impressive performance from CNNs. [9]

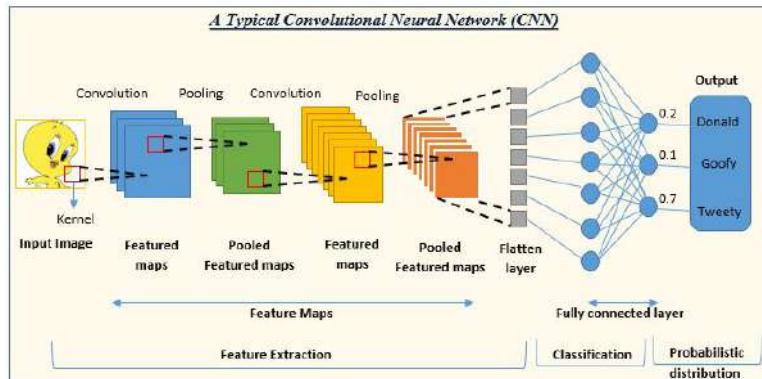


Figure 1.15: The architecture of CNN.[9]

- **Recurrent Neural Networks (RNNs)** : are a kind of artificial neural network that uses feedback loops in its architecture to process input in a sequential fashion. This enables RNNs to forecast or make judgments based on the current input while preserving a state or recollection of the prior inputs. RNNs are useful in language modeling, voice recognition, and natural language processing because they can manage variable-length sequences and capture temporal relationships. [10]

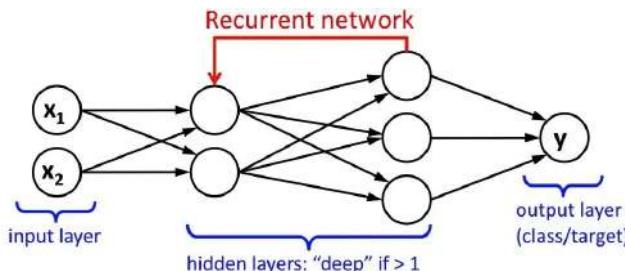


Figure 1.16: The architecture of RNN.[10]

- **Long Short-Term Memory Networks (LSTM)** : Long Short-Term Memory Networks are sequential neural networks with deep learning capabilities that preserve information. It is a unique kind of recurrent neural network that can solve the RNN's vanishing gradient issue. Hochreiter and Schmidhuber created LSTM to address the issue brought about by conventional RNNs and machine learning methods. [50]

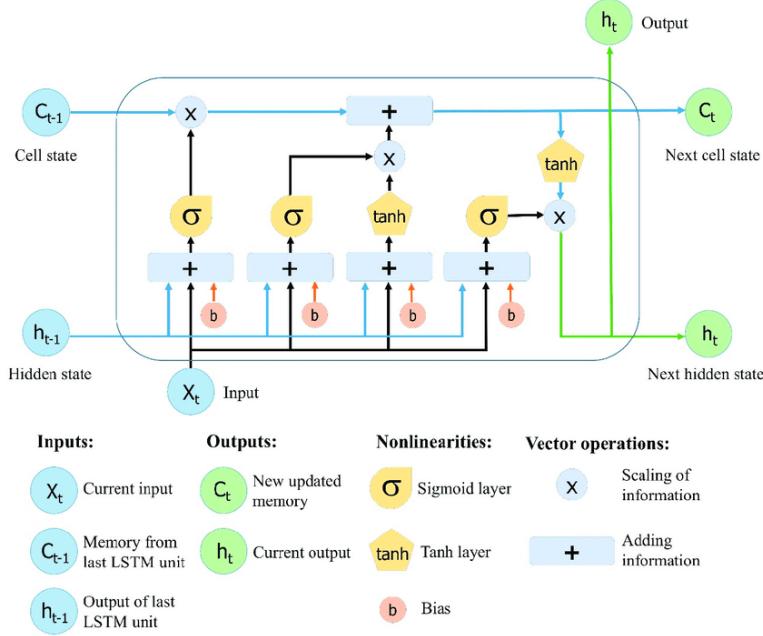


Figure 1.17: The architecture of GAN.

- **Generative Adversarial Networks (GANs)** : are a kind of deep learning models made up of a discriminator and a generator neural network. While the discriminator network learns to distinguish between the created data and the genuine data, the generator network learns to create synthetic data samples that mimic real data from a training set. GANs employ adversarial training to generate high-quality synthetic data that may be used for a variety of tasks, including creating images and videos, and that can be distinguished from actual data. [51]

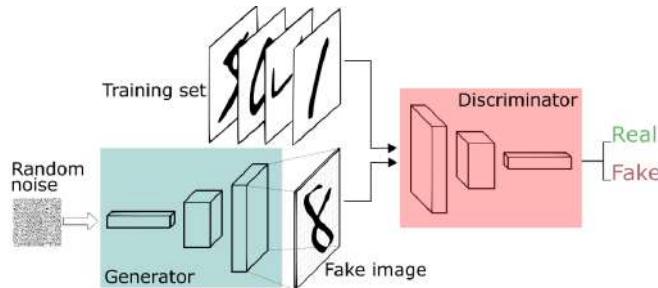


Figure 1.18: The architecture of GAN.[11]

- **Reinforcement Learning (RL)** : is a type of machine learning It involves figuring out how to behave in a situation to maximize benefits. Similar to how toddlers explore their environment and pick up skills that help them accomplish goals, this ideal conduct is acquired through interactions with the environment and observations of how it responds.[52]

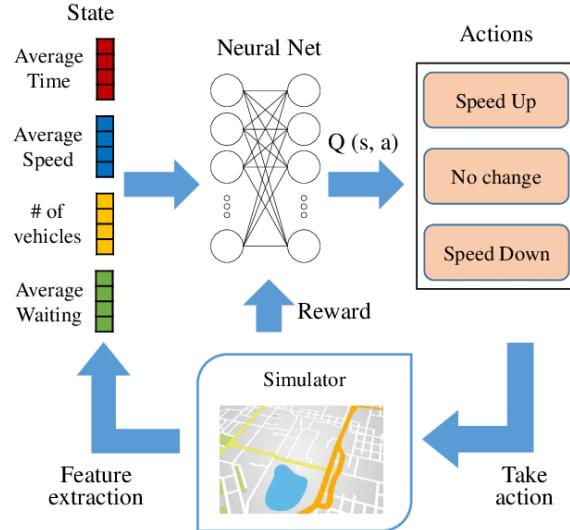


Figure 1.19: The architecture of RL.[12]

- **Transfer Learning :** a method where a pre-trained deep learning model's parameters are adjusted on a smaller dataset to fit a new task or domain. Transfer learning makes use of information gained from one job to enhance performance on a different one. [53]

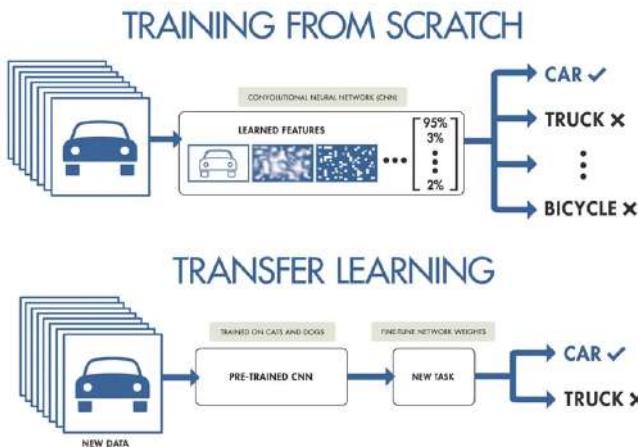


Figure 1.20: The architecture of Transfer Learning.[13]

- **Attention Mechanisms :** Attention mechanisms in deep learning optimize sequential or set-like data tasks by selectively emphasizing specific input components. They improve handling of long-term dependencies, accommodate variable-length sequences, and prevent overfitting. By computing attention weights, these mechanisms determine the contribution of each input element to the output, enhancing model performance. [54]

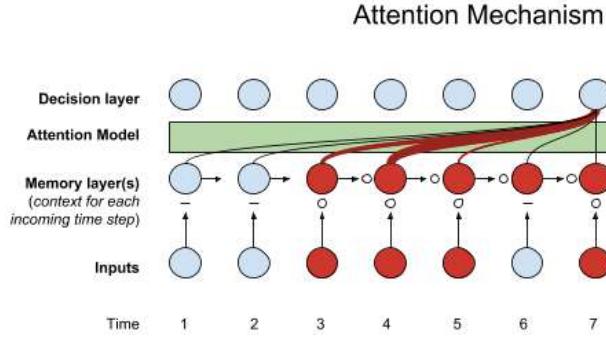


Figure 1.21: The architecture of Transfer Learning.

- **Autoencoders for unsupervised learning and feature extraction:** One kind of neural network that may be used for feature extraction and unsupervised learning is an autoencoder. Its objective is to reassemble the input data by passing through a bottleneck layer in the network's center. It consists of an encoder and a decoder. Through this method, significant characteristics from the input data are extracted and utilized for further tasks like classification and clustering. The network is encouraged to learn effective representations of the data by the autoencoder, which is taught to reduce the reconstruction error between the input and the output. [55]

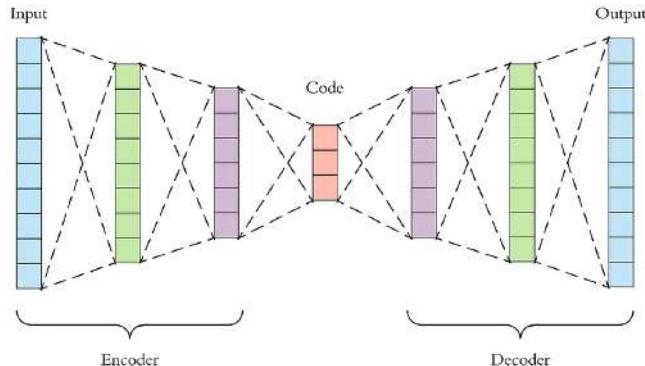


Figure 1.22: The architecture of Transfer Learning.[13]

Advanced Techniques in Deep Learning

Advanced techniques in deep learning encompass a range of methodologies tailored for tasks such as classification, detection, and segmentation. These techniques leverage complex neural network architectures and optimization strategies to achieve superior performance in handling intricate patterns and large-scale datasets.

- **Classification:** is the act of giving a specific label or category to a given input, depending on its distinctive characteristics. In the domain of image classification, deep learning models are trained using extensive datasets of labeled pictures to acquire the ability to identify various objects or patterns in images and accurately give them the appropriate label. This procedure utilizes neural networks and a range of methods, including convolutional layers, pooling, and activation functions, to extract and modify the characteristics of the input data into a format suitable for classification. [56]
- **Object detection:** Object detection is the procedure of identifying the existence of objects in an image or video and outlining a bounding box around them. The process entails classifying the item and

ascertaining its precise position within the picture or video. Typically, this procedure is accomplished by using object identification algorithms that use deep neural networks to extract features and classify objects. Deep learning has greatly enhanced the precision and effectiveness of object recognition in many fields such as surveillance, robotics, and autonomous vehicles. [57]

- **Segmentation:** picture segmentation refers to the process of dividing an image into many segments or regions, where each area corresponds to a distinct item or component of the image. This is commonly done by giving a label to each pixel in the picture, specifying which item or area it belongs to. Deep learning methods, such as convolutional neural networks (CNNs), have been extensively employed for image segmentation tasks, and have demonstrated promising results in many applications such as medical imaging, autonomous driving, and remote sensing. [58]

In general, classification focuses on assigning labels to entire images, detection aims at localizing and labeling objects within images, and segmentation entails labeling every pixel within an image.

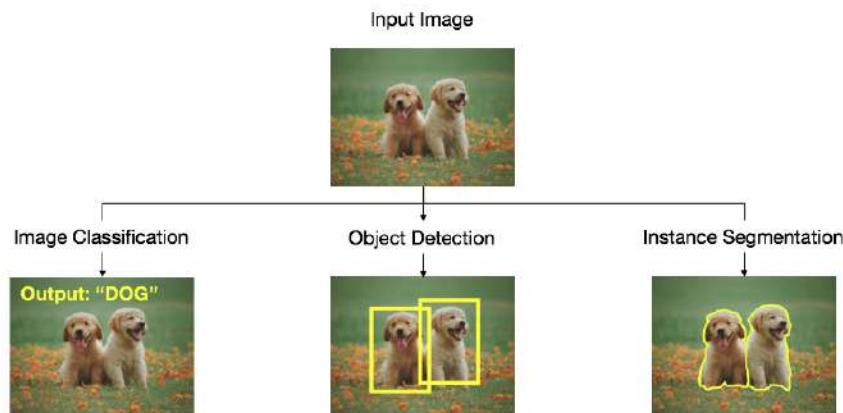


Figure 1.23: Difference between classification, detection and segmentation .[14]

1.8 Datasets(Benchmarks)

In the training phase of machine learning algorithms, data is crucial since it provides the basis for AI to understand human-like thinking processes. Additionally, data accelerates the learning curve; as algorithms consume more data, their accuracy increases. With the increased accessibility of datasets nowadays, machine learning models have experienced a dramatic metamorphosis.

This section will delve into benchmark databases for detection and their details, along with the commonly used models in each database.

1.8.1 MS COCO dataset :

Microsoft released the MS COCO dataset, which is a large-scale dataset for object identification, picture segmentation, and captioning. The COCO dataset is widely used by machine learning and computer vision researchers for a variety of computer vision applications. One of computer vision's main objectives is to understand visual situations. This includes identifying the items in the picture, localizing them in two and three dimensions,

figuring out their characteristics, and describing the relationships between them. As a result, the dataset may be used to train item identification and classification algorithms. [59]



Figure 1.24: Example of images of MS-COCO [15]

```

annotation{
  "id": int,
  "image_id": int,
  "category_id": int,
  "segmentation": RLE or [polygon],
  "area": float,
  "bbox": [x,y,width,height],
  "iscrowd": 0 or 1,
}
  "annotations": [
    {
      "segmentation": [[510.66, 423.01, 511.72, 420.03, ..., 510.45, 423.01]],
      "area": 702.10,
      "iscrowd": 0,
      "image_id": 397133,
      "bbox": [433.07, 355.93, 138.65, 228.67],
      "category_id": 18,
      "id": 1768
    },
    {
      "segmentation": {
        "counts": [12, 56, 198, 10]
        "size": [120, 240]
      }
      "area": 500.2,
      "iscrowd": 1,
      "image_id": 397122,
      "bbox": [473.07, 355.93, 38.65, 28.67],
      "category_id": 18,
      "id": 1768
    }
  ]
}
  
```

Figure 1.25: Example of annotation of MS-COCO

Task	Popular Models
Object Detection	Faster R-CNN, RetinaNet, YOLO, EfficientDet, Cascade R-CNN
Instance Segmentation	Mask R-CNN, Panoptic FPN, BlendMask, SOLOv2, HTC
Keypoint Detection	Mask R-CNN, HRNet, SimpleBaseline, HigherHRNet
Image Captioning	Show and Tell, Neural Baby Talk, Att2in, AoANet
Visual Question Answering	Bottom-up and Top-down, Stacked Attention, MCB

Figure 1.26: The most popular models used for various tasks on the MSCOCO

1.8.2 PASCAL VOC dataset :

PASCAL VOC (Visual Object Classes) stands as a renowned computer vision dataset, pivotal for tasks like object detection, segmentation, and classification. Comprising 20 distinct object classes ranging from animals to household items, it annotates images with bounding boxes and class labels. The dataset is split into training, with around 11,000 images, and validation, with about 5,000, all annotated with segmentation masks and

labels. Serving as a benchmark, PASCAL VOC has spurred advancements in object detection and segmentation algorithms, shaping the landscape of computer vision research. [60]

```

<annotation>
    <folder>Kangaroo</folder>
    <filename>00001.jpg</filename>
    <path>./Kangaroo/stock-12.jpg</path>
    <source>
        <database>Kangaroo</database>
    </source>
    <size>
        <width>450</width>
        <height>319</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>kangaroo</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>233</xmin>
            <ymin>89</ymin>
            <xmax>386</xmax>
            <ymax>262</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 1.27: Example of annotation of PASCAL VOC

Task	Popular Models
Object Detection	Faster R-CNN, YOLO, SSD, R-FCN, RFCN-3000
Object Detection + Segmentation	Mask R-CNN
Object Detection + Fine-grained classification	FRCNN-MLP, FRCNN-CNN
Object Detection + Multiple Detections per Image	R-FCN, Faster R-CNN, YOLO, SSD

Figure 1.28: The most popular models used for various tasks on the PASCAL VOC

1.8.3 ImageNet dataset

The WordNet hierarchy is used to arrange the images in the ImageNet dataset. "Synonym set" or "synset" refers to any relevant notion in WordNet that may be defined by a group of words or word phrases. WordNet has more than 100,000 synsets, of which more than 80,000 are nouns. Our goal at ImageNet is to offer each synset with an average of 1000 photos. Every concept's image has undergone quality control and human annotation. When ImageNet is finished, we anticipate that it will provide tens of millions of neatly labeled and arranged pictures for the majority of the ideas found in the WordNet hierarchy. [61]



Figure 1.29: Example of images of ImageNet

Task	Popular Models
Image Classification	ResNet, Inception, VGG, AlexNet
Object Detection	Faster R-CNN, SSD, YOLO
Semantic Segmentation	FCN, U-Net, DeepLab
Image Generation	GANs (Generative Adversarial Networks)
Image Captioning	LSTM, Transformer-based models

Figure 1.30: The most popular models used for various tasks on the open Image

1.8.4 Open Images dataset

Open Images is a large dataset with more than 9 million photos and a large number of annotations (15.4 million bounding boxes in 600 categories). With its extensive annotations, this dataset is a goldmine for data-intensive techniques and is establishing new benchmarks in object recognition. Open Images provides a wealth of information covering more than 6,000 categories, arranged into subsets for training, validation, and testing. This allows for the advancement of cutting-edge computer vision algorithms. [62]



Figure 1.31: Example of images of Open Image

Task	Popular Models
Image Classification	InceptionV3, ResNet50, VGG16, EfficientNet, MobileNet.
Object Detection	Faster R-CNN, SSD ,YOLO, RetinaNet, EfficientDet.
Visual Relationship Detection	BAR-CNN with ResNet50, Graph R-CNN,Neural Motifs, Scene Graph Generation Models.

Figure 1.32: The most popular models used for various tasks on the Open Image

1.9 Conclusion

In summary, this chapter gave a general overview of AI-based assistive technology for the blind. We looked at the drawbacks of conventional approaches and emphasized how artificial intelligence (AI) may be useful in helping people with vision impairments overcome their obstacles. We spoke about what the human visual system is capable of, how we see things in three dimensions, and how deep learning plays a part in creating artificial intelligence-based solutions. We also covered popular deep learning frameworks and benchmark datasets. This chapter lays the groundwork for the next chapters, which will go into further detail on the particular AI models and methods used to increase accessibility and improve the lives of those who are visually impaired.

Chapter 2

Real-Time Object Detection with Deep Learning

2.1 Introduction

object detection is a crucial area of computer vision , which is essential to many applications including self-driving cars, security systems, and medical diagnostics. Object recognition has been transformed by the use of deep learning algorithms, which provide previously unheard-of levels of accuracy and efficiency above conventional computer vision methods. This chapter will examine the field of deep learning-driven real-time object recognition, with an emphasis on popular models like R-CNN, SSD, RetinaNet, and YOLO. We will also look at how these developments may be used in the real world, especially in creating AI-based aids for those who are blind or visually impaired. These technologies, which provide real-time support and navigation capabilities, not only increase accessibility but also greatly improve quality of life. We will also talk about the possibility of real-time object identification in the future, including the usage of more advanced neural network designs and the integration of edge computing.

2.2 Real-Time Object Detection

Real-time object detection has become a crucial component of computer vision, focusing on the swift and accurate identification of objects within images and video streams. This capability is widely used across various applications, such as autonomous vehicles, robotics, surveillance, and augmented reality. The advent of deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized this field by enabling the autonomous extraction of intricate feature representations from raw pixel data, allowing for robust object recognition.

Several deep learning-based object detection models have been developed, each with distinct architectures and advantages. Notable examples include YOLO (You Only Look Once), SSD (Single Shot Multibox Detector), and Faster R-CNN (Region-based Convolutional Neural Networks). These models strike a balance between accuracy and speed to address the demands of real-time applications. The real-time object detection process

involves collecting and annotating data, training the model, and performing inference, where the models analyze live video streams or image sequences to generate bounding boxes and class labels for detected objects.

Despite significant advancements, real-time object detection remains an ongoing area of research. Optimization techniques, such as model quantization and hardware acceleration using GPUs or TPUs, enhance processing speed and efficiency. Researchers are continuously exploring new algorithms, architectures, and hardware improvements to increase model accuracy, efficiency, and adaptability, with the goal of developing robust real-time object detection systems.

Moreover, real-time object detection supports the creation of assistive technologies for the visually impaired, including wearable devices that can detect objects and provide auditory or haptic feedback. These innovations hold the potential to greatly improve the quality of life for individuals with visual impairments by enhancing their independence and mobility. [63]

There are several features that can influence real-time object detection:

- **Model architecture:** The selection of model architecture significantly impacts the speed and accuracy of real-time object detection. To strike a balance between speed and accuracy, neural network architectures such as YOLO (You Only Look Once), SSD (Single Shot Multibox Detector), and Faster R-CNN (Region-based Convolutional Neural Networks) are crucial. Each architecture possesses unique features that make it suitable for different real-time applications.
- **Data Quality and Quantity:** Effective object detection relies on high-quality and diverse datasets. Abundant and well-prepared data improve the model's ability to learn and generalize, enhancing accuracy in real-world scenarios.
- **Hardware Acceleration:** Using hardware accelerators like GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units) significantly boosts processing speed and efficiency. These accelerators handle the complex computations required for deep learning, enabling real-time inference.
- **Algorithm Efficiency:** The efficiency of algorithms used for feature extraction, object proposal generation, and classification directly impacts the speed and accuracy of real-time object detection. Efficient algorithms are crucial for processing visual data quickly and accurately.
- **Environmental Conditions :** Real-time object detection can be affected by varying environmental conditions such as lighting and weather. Robust models must handle these variations to maintain accuracy in different scenarios.
- **Sensor Quality :** The quality and capabilities of sensors, including cameras and depth sensors, influence object detection performance. High-resolution and high-frame-rate sensors provide clearer and more detailed data, improving detection accuracy.

By considering and optimizing these features, real-time object detection systems can achieve higher accuracy, faster inference times, and greater robustness across various applications.

2.2.1 Accuracy

Accuracy in detecting objects in real time is a sensitive and important criterion, as it directly impacts the reliability and effectiveness of the application. High accuracy ensures that objects are correctly identified and localized, which is crucial in critical applications such as autonomous driving, where misdetections or false positives can lead to severe consequences. Accurate detection also enhances the user experience in assistive technologies for visually impaired individuals, providing precise and timely information that supports safe navigation and interaction with the environment. Furthermore, in surveillance and security systems, accuracy is vital for correctly identifying threats and ensuring appropriate responses. Therefore, optimizing for accuracy in real-time object detection is essential for maximizing the benefits and minimizing the risks associated with these technologies.

2.2.2 Frames per second (Fps)

The frame rate, or frames per second (FPS), is a critical factor in real-time object detection, indicating the system's efficiency in processing video frames rapidly. Achieving high FPS is essential to reduce latency in identifying and tracking objects. Factors like algorithm selection, input image resolution, scene intricacy, and hardware capabilities influence the FPS of an object detection system. To reach high FPS rates, techniques such as model optimization, parallel computing, and hardware enhancements are employed. Striking a balance between accuracy and speed is critical for functional real-time object detection systems, with ongoing research focused on enhancing both aspects.

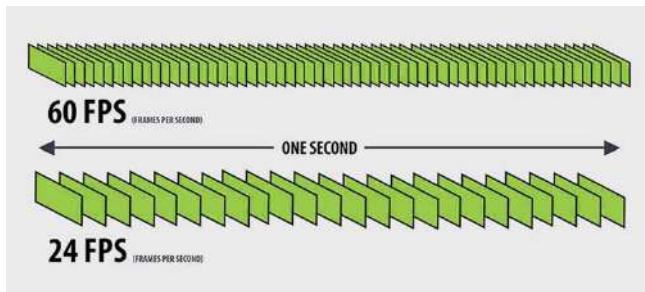


Figure 2.1: Difference Between 60FPS And 24FPS[16]

2.3 How to choose the best model for object detection system ?

Selecting the most suitable model for an object detection system is a crucial process that involves several key considerations to ensure both high performance and practical applicability. The selection process must account for the following factors:

- 1. Accuracy Requirements:** The main goal of an object detection system is to accurately identify and localize objects within images or video streams. Models like Faster R-CNN, YOLO, and SSD each offer different trade-offs between precision and speed. Faster R-CNN is renowned for its high accuracy, making it ideal for applications where precision is crucial. On the other hand, YOLO and SSD strike a balance between accuracy and speed, providing real-time detection capabilities that are suitable for scenarios where speed is also essential.

2. Computational Efficiency: The computational resources available significantly impact the choice of model. Models like YOLO, which prioritize speed, are optimized for deployment on devices with limited computational power, such as mobile phones or embedded systems. On the other hand, more complex models like Faster R-CNN may require robust hardware, including GPUs or TPUs, to process data efficiently.

3. Input Data Characteristics: The nature of the input data, including image resolution and scene complexity, influences model selection. High-resolution images and complex scenes with numerous objects demand models capable of handling detailed information without compromising on performance. Models should be evaluated on their ability to maintain high FPS (frames per second) while processing diverse and intricate datasets.

4. Application Context: The specific use case of the object detection system guides model choice. For instance, autonomous driving systems necessitate real-time detection with high reliability, favoring models like YOLO that offer quick responses. In contrast, applications in medical imaging may prioritize accuracy over speed, thus benefiting from models with higher precision like Faster R-CNN.

5. Scalability and Adaptability: The model's ability to scale and adapt to different environments and tasks is also crucial. Transfer learning and fine-tuning capabilities enable models to be repurposed for various applications with minimal additional training. Models that can be easily adapted to new datasets and evolving requirements are preferable for long-term use.

6. Robustness to Variability: Ensuring the model's robustness to changes in lighting, occlusion, and object variations is essential. Models should be tested extensively under diverse conditions to evaluate their performance consistency. Techniques such as data augmentation during training can enhance the model's ability to generalize across different environments.

In conclusion, selecting the optimal model for an object detection system involves a comprehensive assessment of accuracy, computational efficiency, input data characteristics, application context, scalability, and robustness. By meticulously evaluating these factors, researchers and practitioners can identify the most suitable model that meets the specific requirements and constraints of their application, thereby maximizing the system's effectiveness and reliability.

2.4 Object Detection Models

Object detection using deep learning has seen significant advancements, and we will explore several models that have been developed to assist in navigation, particularly in the context of helping blind people navigate outdoors.

2.4.1 EfficientDet

- **Description:** EfficientDet is a family of object detectors that leverage a new model scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. It is designed to achieve a balance between speed, accuracy, and computational efficiency, making it suitable for deployment on mobile and edge devices. [64]

- **Use Case for Blind Navigation:** Given its efficiency and compact size, EfficientDet can be implemented in mobile devices or wearable technology that assists blind users in real-time object detection.

2.4.2 MobileNetV2

- **Description:** MobileNetV2 is a lightweight deep learning model optimized for mobile devices. When combined with SSD, it provides a powerful yet efficient object detection system that can run on devices with limited computational resources, making it ideal for portable applications. [65]
- **Use Case for Blind Navigation:** This combination is perfect for low-power devices like smartphones or wearable glasses that assist blind individuals by detecting objects in real-time without requiring heavy computational power.

2.4.3 Faster R-CNN (Region Convolutional Neural Network)

- **Description:** Faster R-CNN is an advanced object detection model that incorporates a Region Proposal Network (RPN) to generate region proposals likely to contain objects. It is highly accurate but slower compared to YOLO and SSD, making it more suitable for applications where accuracy is prioritized over speed. [66]
- **Use Case for Blind Navigation:** Faster R-CNN could be used in applications where accuracy in detecting and classifying objects is critical, possibly in scenarios where blind users need to identify specific objects, like bus stops or traffic signs.

2.4.4 SSD (Single Shot MultiBox Detector)

- **Description:** SSD is another popular object detection model known for balancing speed and accuracy. It detects objects in images in a single pass, without requiring a region proposal network, which makes it faster than traditional methods. SSD is particularly effective for detecting multiple objects of different scales within an image. [67]
- **Use Case for Blind Navigation:** SSD's ability to quickly detect multiple objects makes it ideal for outdoor navigation systems where real-time processing is crucial.

2.4.5 RetinaNet

- **Description:** RetinaNet is a popular object detection model that addresses the problem of class imbalance in object detection by introducing a new loss function called "Focal Loss." This makes it particularly effective at detecting small objects and handling difficult cases where certain classes are underrepresented. RetinaNet offers a good balance between speed and accuracy. [68]
- **Use Case for Blind Navigation:** RetinaNet is useful in scenarios where detecting smaller or less prominent objects (like curbs or small obstacles) is essential for safe navigation. It can be deployed in wearable devices or mobile applications that assist visually impaired individuals by detecting these objects in real-time.

2.4.6 YOLO (You Only Look Once)

The YOLO (You Only Look Once) method revolutionized object identification with its precise and efficient approach. It was first reported in the publication "You Only Look Once: Unified, Real-Time Object Detection" by Joseph Redmon et al. in June 2016[17]. In contrast to conventional techniques, YOLO does not need region suggestions since it can anticipate bounding boxes and class probabilities directly in a single assessment. The network makes accurate predictions by reasoning globally across all objects by using the attributes of the whole picture. Predicting bounding boxes for every class at the same time improves detection unity and efficiency. Faster inference times are achieved by enabling collaborative training of the whole detection pipeline via the end-to-end (Figure 2.2) learning of the YOLO network. Amazingly, YOLO maintains a high average accuracy in spite of its real-time speed, which makes it ideal for a variety of applications.

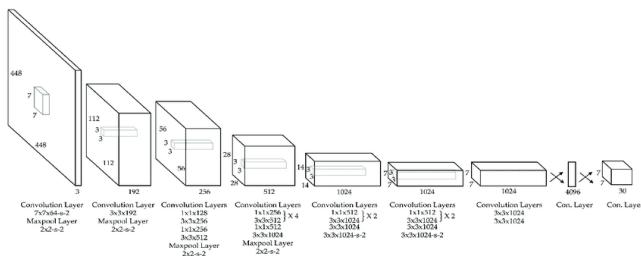


Figure 2.2: YOLO model architecture [17]

There are 2 fully linked layers and 24 convolutional layers in the network design. For picture classification, it draws inspiration from the GoogLeNet model, however it uses different initialization modules. YOLO, on the other hand, uses 3x3 convolutional layers after 1x1 reduction layers. With this improvement, YOLO can accurately identify objects in real-time and offers a single method for completing the job.

The YOLO algorithm model is divided into three stages:

- **Image Division into Cells:** The image is split into a grid of size $S \times S$, such as 3×3 in this example, resulting in a total of N cells. Each cell in the grid is responsible for detecting objects within its assigned region.

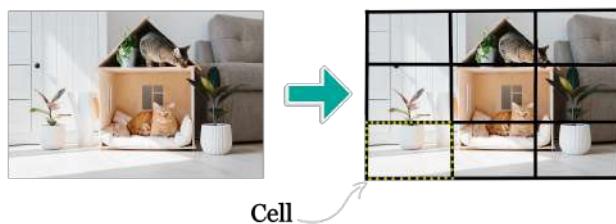


Figure 2.3: Divide the image into $(S \times S)$ grid.

- **Each cell predicts B bounding boxes:** After dividing the image into N cells, each cell in the grid predicts B bounding boxes along with confidence scores for those boxes. Each bounding box

prediction consists of five components: x , y , w , h , and confidence. The coordinates (x , y) represent the center of the box relative to the boundaries of the grid cell. The width and height predictions (w , h) are given relative to the entire image. Lastly, the confidence score reflects the Intersection over Union (IoU) between the predicted box and any ground truth box. [17]. **Example:** Imagine that the picture is split into a 3×3 grid of cells ($S=3$). The prediction is done for a single bounding box ($B=1$) in each cell, and the objects that are present are either people (2 in this case) or horses (1 in this case). For every cell, the CNN predicts a vector Y . (Figure 2.4).

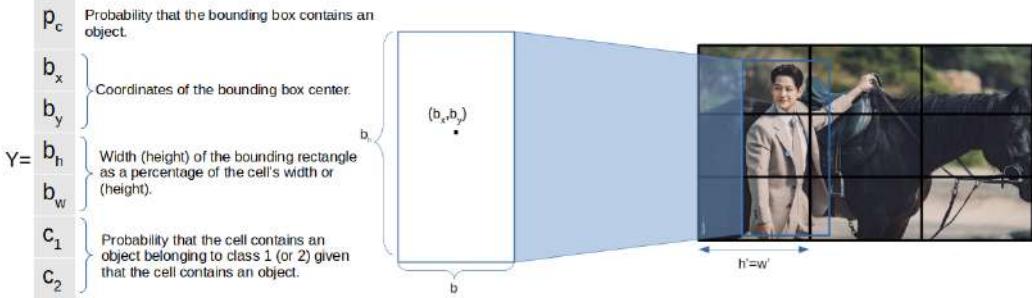


Figure 2.4: The predicted vector in the case of a single box.

The values of vector Y in the YOLO format are calculated as follows:

- * **Pc:** The confidence prediction represents the Intersection over Union (IoU) between the predicted box and the ground truth box.
- * **bx:** Computed as $(x - h') / h'$
- * **by:** Computed as $(y - w') / w'$
- * **bh:** Computed as $h / 416$
- * **bw:** Computed as $w / 416$
- * **Intersection over Union (IoU):** Intersection over Union (IoU) is a widely used evaluation metric in object detection. It measures the overlap between the predicted bounding box and the ground truth box , it acts as a gauge for forecast accuracy. A precision criterion is chosen in order to assess the performance [69] . The area of intersection divided by the area of union, or IoU (Figure 2.5) , provides important information about the accuracy of object detection predictions.

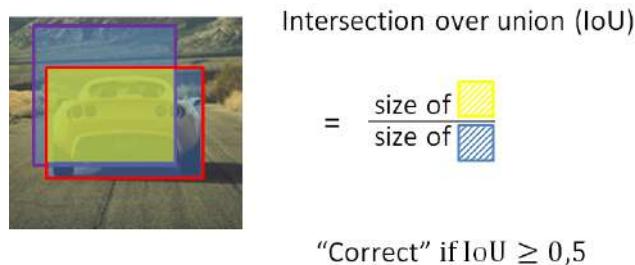


Figure 2.5: Intersection over union.

In the training phase, the projected bounding box's confidence is assessed by comparing its Intersection over Union (IoU) score to that of the ground truth box. figure 2.6, provides instances of both high and low IoU scores. It is clear that greater points are awarded to projected bounding boxes that show a significant overlap with the ground truth boxes, while lower values are given to those that show less overlap.

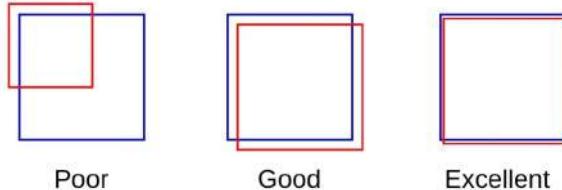


Figure 2.6: Examples of IoU.

- * **Anchor Box :** In the preceding instance, we examined the situation in which a single bounding box was anticipated. On the other hand, YOLO uses anchor boxes to deal with situations when there are many bounding boxes in a single grid cell. Each cell is represented by a vector, as was previously explained. The vector is changed to provide room for the extra bounding boxes when a cell contains more than one box.

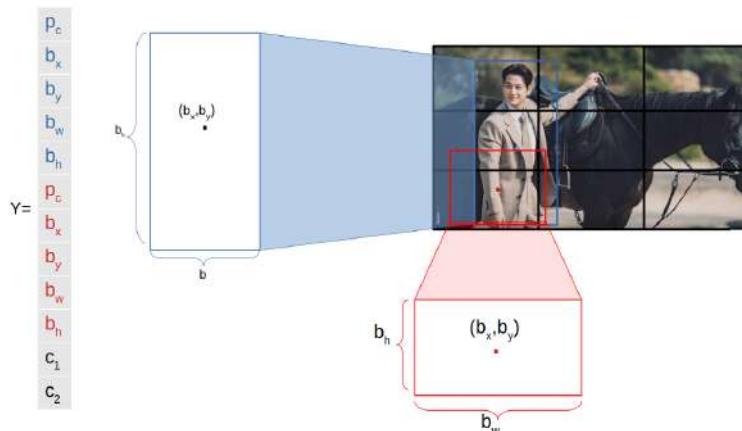


Figure 2.7: The predicted vector in the case of multiple boxes in the cell.

Generally speaking, each individual cell in the $S \times S$ grid that we create from the picture predicts B bounding boxes, the related confidence scores, and the class probabilities C . Next, a tensor with dimensions is encoded using these predictions. .[17]

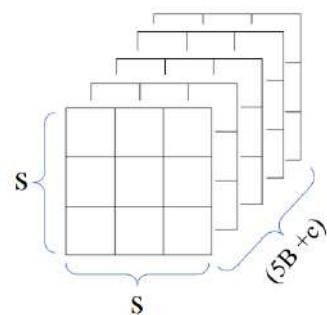


Figure 2.8: A tensor that specifies the bounding box

- * **Non Maximum Suppression :** is used as the last step of the detection process when more than one bounding box finds the same item in the picture. Its goal is to keep just the most accurate bounding box after removing the less probable ones. To do this, the approach consists of five phases.[18]

- **Step 1:** Select the bounding box with the highest confidence score as the initial choice.
- **Step 2:** Assess the overlap (Intersection over Union) between this chosen box and the other boxes.
- **Step 3:** Remove bounding boxes that have an overlap (Intersection over Union) exceeding 50%.
- **Step 4:** Proceed to the next bounding box with the highest confidence score.
- **Step 5:** Repeat steps 2 through 4 until all objects in the image have been addressed.

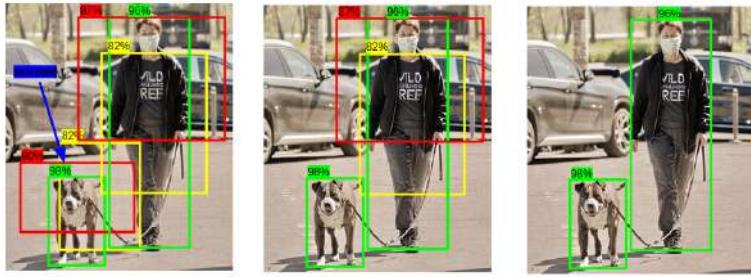


Figure 2.9: The output after different steps of NMS [18]

YOLOv2 Model

Joseph Redmon and Ali Farhadi set out to enhance the YOLO (You Only Look Once) model in December 2016 with the goal of making it more resilient, quick, and efficient. As a result, YOLOv2, a major improvement over the original YOLO system, was created. A unique multi-scale training technique, one of the many enhancements introduced by YOLOv2, enables the model to operate well at varying scales while providing a balance between speed and accuracy. YOLOv2 outperforms state-of-the-art techniques like SSD and quicker R-CNN with ResNet, achieving 76.8 mAP at 67 FPS and 78.6 mAP at 40 FPS on the PASCAL VOC 2007 dataset while retaining quicker processing rates.

Additionally, YOLOv2 integrates object detection and classification tasks into a unified training approach. By training on both the ImageNet classification dataset and the COCO detection dataset, YOLOv2 can predict detections for classes that lack labeled detection data. This is demonstrated by YOLO9000 (an extension of YOLOv2), which achieves a mean average precision (mAP) of 19.7 on the validation set with only 44 out of 200 classes having detection data, and 16.0 mAP for the remaining 156 classes not included in COCO.

By expanding its detection capabilities to more than 9000 item types, YOLO9000 demonstrates how effective and thorough detection algorithms may make use of large amounts of classification data. This development highlights the potential of YOLOv2 in applications that need fast and precise object recognition across a variety of categories. [70].

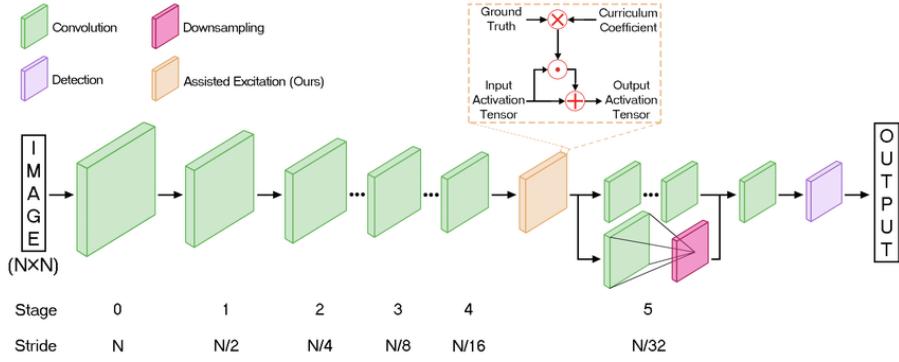


Figure 2.10: YOLOv2 architecture[19]

YOLOv3 Model

YOLOv3, an enhanced iteration of the YOLO (You Only Look Once) object detection paradigm, was unveiled by Joseph Redmon and Ali Farhadi in 2018 [71]. With a number of design changes, this update sought to improve the model's efficiency and accuracy. Even with its improved accuracy, YOLOv3 maintains the speed of its predecessors. With a mean Average Precision (mAP) of 28.2, it runs at 320 x 320 resolution in 22 milliseconds, three times quicker than SSD. YOLOv3 outperforms RetinaNet, which hits 57.5 AP50 in 198 milliseconds, by achieving 57.9 AP50 for the .5 Intersection over Union (IOU) mAP detection measure in 51 milliseconds on a Titan X. With a successful combination of speed and accuracy, this new version further demonstrates the promise of the YOLO framework for real-time object recognition. The fact that the whole code and trained models are accessible online encourages improvements and uses in a range of computer vision applications.

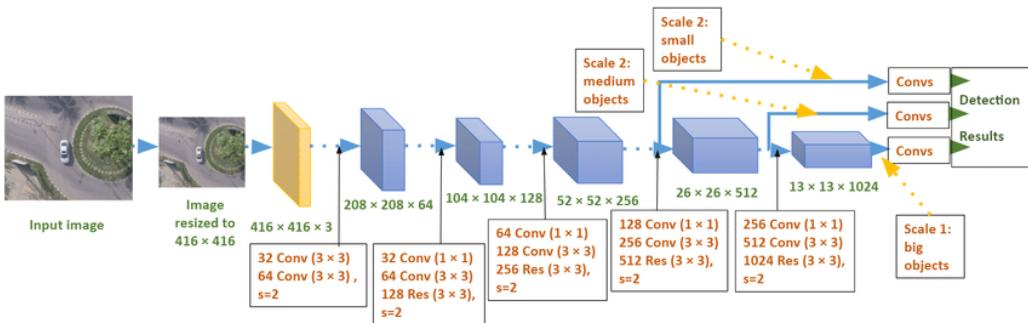


Figure 2.11: YOLOv3 Architecture

YOLOv4 Model

The improved YOLOv4 (You Only Look Once) object identification model was presented by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao in 2020. The goal of this iteration was to improve the model's efficiency and accuracy for real-time applications. YOLOv4 incorporates several advanced features, such as Mish activation, Cross mini-Batch Normalization (CmBN), Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), and Self-adversarial-training (SAT). With these improvements, YOLOv4 can operate at about 65 frames per second (FPS) on a Tesla V100 GPU and achieve impressive performance, with a 43.5% Average Precision (AP) and 65.7% AP50 on the MS COCO dataset. YOLOv4's main objective is to provide an extremely accurate and efficient object recognition

model that can run in real-time on traditional GPUs, enabling sophisticated object identification to be used in a variety of applications. [72]

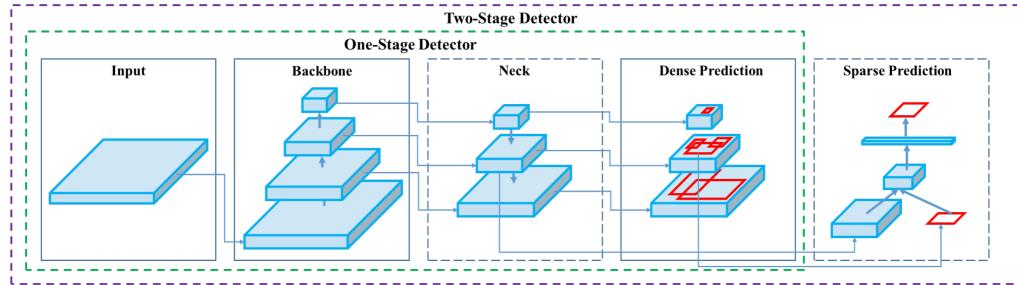


Figure 2.12: YOLOv4 Architecture [20]

YOLOv5 Model

The creator and CEO of Ultralytics, Glenn Jocher, created the object detection model YOLOv5 in 2020. YOLOv5, which was developed using PyTorch, builds on the improvements made in YOLOv4 and has a modified CSPDarknet53 backbone, a neck that uses SPPF and a modified CSP-PAN, and a head that resembles YOLOv3.

The Ultralytics AutoAnchor algorithm, which dynamically modifies anchor boxes to better match the dataset and training parameters, is one of the main advances in YOLOv5. In order to strengthen stability against runaway gradients, YOLOv5 also adds a number of augmentations, including MixUp, random affine, Mosaic, and HSV augmentation.

YOLOv5 provides many scaled variants (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x) to accommodate varying hardware limitations and application requirements. These variations range in width and depth from high-performance models geared for speed to lightweight ones for devices with limited resources.

In general, YOLOv5 offers improved object recognition skills, real-time performance, and a speed-accuracy trade-off that is well-balanced. It has a large and vibrant community of contributors and is actively updated, open source, and supported by Ultralytics.[73]

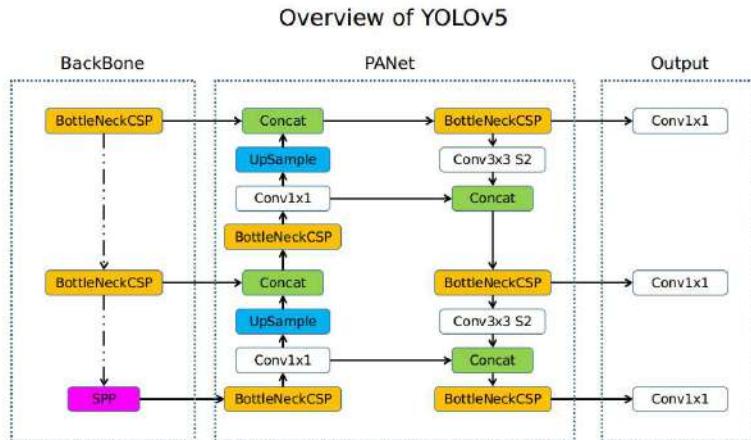


Figure 2.13: YOLOv5 Architecture [21]

YOLOv6 Model

YOLOv6, introduced in 2022, improves upon earlier versions with enhanced network design and training methods, striking a balance between speed and accuracy. It features a Bi-directional Concatenation (BiC) module in the neck for better localization and the SimCSPSPPF block, which simplifies the SPPF block for faster performance. YOLOv6 also uses anchor-aided training (AAT), blending anchor-free and anchor-based paradigms while maintaining inference efficiency. Additionally, it extends the backbone and neck for improved COCO dataset performance. A self-distillation approach enhances smaller models, with the RepBi-PAN design ensuring precise localization, especially for small objects.

YOLOv6 offers many models to meet different requirements. For example, YOLOv6-N can obtain 37.5% AP at 1187 FPS on the COCO dataset, while YOLOv6-S can get 45.0% AP at 484 FPS, exceeding other standard detectors of comparable size. While still retaining competitive inference rates, models like as YOLOv6-M and YOLOv6-L provide even greater accuracy, hitting 50.0% and 52.8% AP, respectively.

In terms of object detection, YOLOv6 is a major advancement as it combines real-time performance with great accuracy. In keeping with the tradition of the YOLO series, it is actively updated, open source, and supported by the YOLO community, offering state-of-the-art solutions for both research and real-world applications [74].

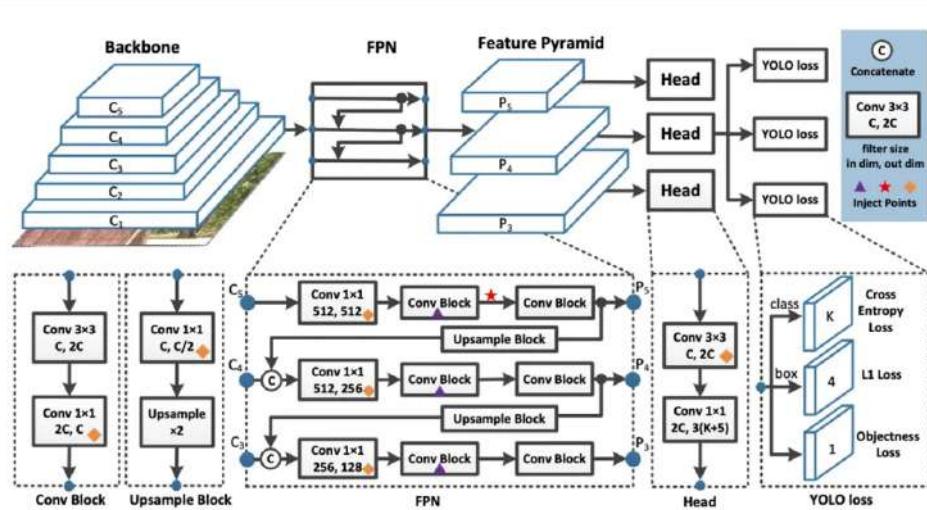


Figure 2.14: YOLOv6 Architecture [22]

YOLOv7 Model

YOLOv7, released by WongKinYiu and Alexey Bochkovskiy in 2022, sets a new standard in object detection with its balance of speed and accuracy. It operates at 5-160 FPS, achieving 56.8% AP at 30+ FPS on an NVIDIA V100 GPU, outperforming all real-time detectors. The YOLOv7-E6 variant reaches 55.9% AP at 56 FPS, surpassing models like ConvNeXt-XL and SWIN-L Cascade-Mask R-CNN.

YOLOv7's standout feature is its use of only the MS COCO dataset for training, without additional datasets or pre-trained weights. It includes innovations such as anchor-aided training, the SimCSPSPPF block for better representation, a BiC module for localization, and a self-distillation approach for smaller models. YOLOv7's optimized modules and "trainable bag-of-freebies" enhance accuracy without adding

inference costs, making it a leading model in real-time object detection.[75]

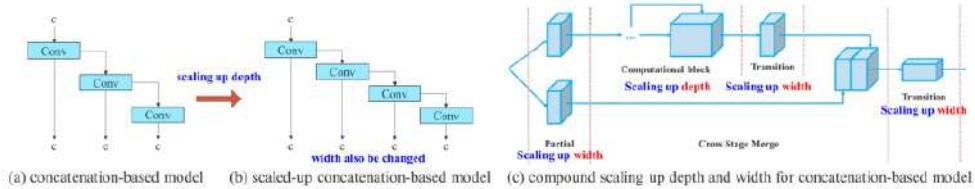


Figure 2.15: YOLOv7 Architecture [23]

YOLOv8 Model

YOLOv8 was created by Ultralytics in January 2023 [24]. Its architecture, which consists of a head, neck, and backbone, is identical to that of YOLOv5. But with a more sophisticated detection head and improved convolutional layers in the backbone, YOLOv8 offers a number of advancements that make it a superior option for real-time object recognition. Additionally, it supports a number of computer vision methods, including instance segmentation, which makes it possible to identify many objects in pictures or movies.

The Darknet-53 backbone network, which powers YOLOv8, is quicker and more precise than the YOLOv7 network. It predicts bounding boxes using an anchor-free detection head, which increases efficacy. More features and a better convolutional network help the model perform faster and more accurately. Moreover, YOLOv8 uses feature pyramid networks to identify objects with varying sizes.

Moreover, YOLOv8 provides an intuitive API that makes it easier to implement in a variety of applications. With its enhanced convolutional layers, enhanced architecture, and enhanced detection capabilities, YOLOv8 is a potent solution for real-time object identification workloads.

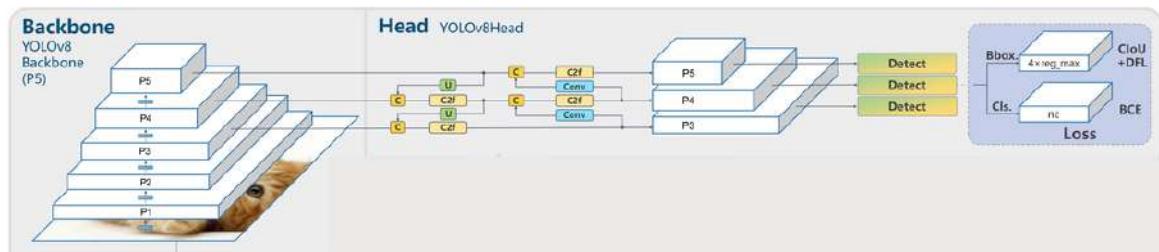


Figure 2.16: YOLOv8 Architecture [24]

YOLO NAS Model

Deci.ai has developed a state-of-the-art object detection model called YOLO-NAS [25], which outperforms YOLOv6 and YOLOv8 in terms of mean average precision (mAP) and inference delay. Pre-trained on datasets like COCO and Objects365, YOLO-NAS is suitable for real-world applications. The model is available through Deci's Super Gradients library, providing pre-trained models for various computer vision tasks.

YOLO-NAS employs Neural Architecture Search (NAS), specifically Deci's AutoNAC technology, to automate the design of the neural network architecture. This approach optimizes stage sizes, topologies,

block types, and the number of channels, balancing accuracy, computational complexity, and model size while considering hardware and data constraints.

The model incorporates Quantization-Aware RepVGG (QA-RepVGG) blocks to enable Post-Training Quantization (PTQ), minimizing accuracy loss through 8-bit quantization and reparameterization. YOLO-NAS uses a hybrid quantization method to optimize performance, balancing accuracy and latency. Additionally, attention mechanisms and inference time reparametrization further enhance the model's real-time object detection capabilities. [76]

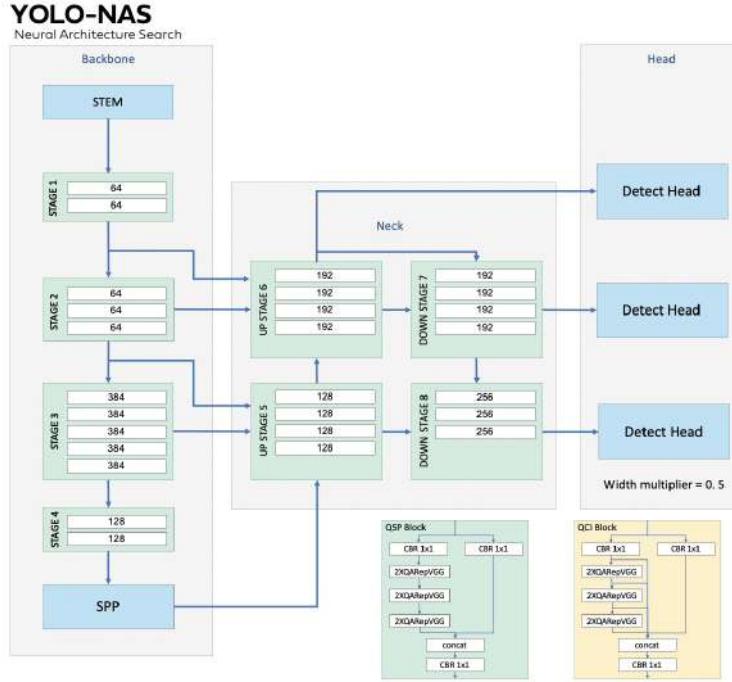


Figure 2.17: YOLO NAS Architecture [25]

2.5 Conclusion

We have studied the area of deep learning for real-time object identification in this chapter. We spoke about the importance of object detection in a variety of applications, including surveillance systems, medical imaging, and self-driving automobiles. The YOLO model, which is a kind of deep learning approach, has significantly improved object identification efficiency and accuracy.

We have selected YOLOv8 as the paradigm for our notion in this area. one of the newest version of the YOLO series is called YOLOv8, and we will examine its intricate architecture as well as its features and capabilities in the next chapter.

In conclusion, computer vision has been completely transformed by real-time object identification with the YOLO paradigm. It is a great option for real-time applications because to its single-pass image processing capability. A look at YOLOv8's comprehensive architecture and its consequences for object detection will be covered in the next chapter.

Chapter 3

Conception

3.1 Introduction

In this chapter We will explore the YOLOv8 model's intricate architecture as well as the procedures for getting data ready and training the model for our system . This procedure is essential to creating a system that offers real-time item detection and identification to blind people. We will also examine the architecture and operation of the system, showing how it provides users with precise and rapid support.

3.2 What makes YOLO a better choice for Object Detection ?

YOLO (You Only Look Once) is a widely recognized object detection model, celebrated for its real-time performance and accuracy. Here are the key differences and strengths of YOLO compared to other object detection models:

- **Single-shot detection :** YOLO is a one-shot object detection model that completes both object classification and detection in a single network run. In contrast, numerous passes or area suggestions are needed for two-shot or multi-shot models like Faster R-CNN. YOLO is much quicker than many other object identification techniques since it is a single-shot model.
- **Speed :** YOLO is a real-time object detection system that processes photos and videos quickly. In order to do this, the input picture is divided into a grid of cells using a fully convolutional architecture, where each cell predicts a certain number of bounding boxes and the class probabilities that go along with it. Because of its speed advantage, YOLO is a good fit for applications like live video analysis, autonomous driving, and video surveillance that need for quick inference times
- **Simplicity :** Comparing YOLO's design to other intricate object detection models such as Faster R-CNN or Mask R-CNN, it is comparatively simple and easy to comprehend. It is composed of fully connected layers for class probabilities and bounding box predictions, after a sequence of convolutional layers. YOLO is easy to train, deploy, and adjust for a variety of applications due to its simplicity.
- **Multi-scale predictions :** YOLO performs predictions at various spatial resolutions by utilizing feature maps from different network layers, enabling detection at multiple scales . With the use of

this multi-scale technique, YOLO can handle items at multiple scales and identify objects of varying sizes, which makes it adaptable to a variety of situations.

- **High accuracy :** Even while it may not always reach the same degree of accuracy as certain cutting-edge models like Cascade R-CNN or EfficientDet, YOLO has shown comparable accuracy on a number of benchmark datasets. Yolo is a popular option for many real-time object detection applications because of its ability to balance speed and accuracy, particularly in situations when prompt reaction is essential.
- **Objectness score :** YOLO introduces the concept of an "objectness" score, which represents the probability that an object is present in a bounding box. This score helps filter out low-confidence detections and reduces false positives, thereby improving the reliability of the detection results.
- **Unified Detection Framework :** YOLO optimizes a combined loss function that addresses both localization (bounding box regression) and classification tasks simultaneously. This unified approach allows YOLO to treat object detection as a single problem, making it more computationally efficient and easier to implement. However, it's important to note that this unified method may not achieve the same performance as two-stage models, which handle each task separately.

YOLO has a large and active community, contributing to its continuous improvement. Each version of YOLO (e.g., YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, YOLOv8, and recently YOLONAS and YOLOv9 And YOLOv10) introduces enhancements and modifications to the original architecture. These advancements further refine the accuracy, speed, and usability of YOLO for diverse object detection tasks.

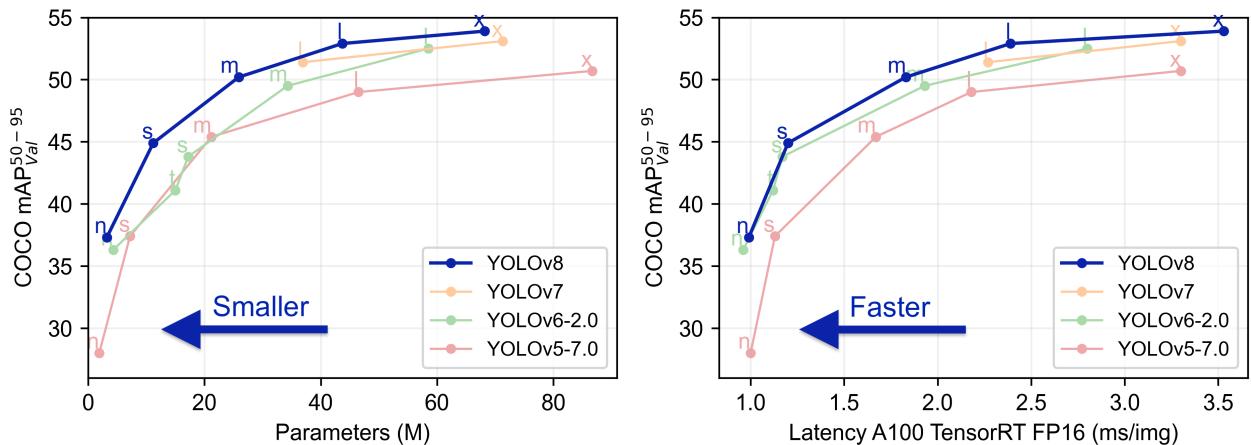


Figure 3.1: Comparing YOLOv8 to Other YOLO Models: A Comparative Analysis.[26]

3.3 Global architecture

These are the main steps in our system's process, as illustrated in the global architecture figure 3.6 .

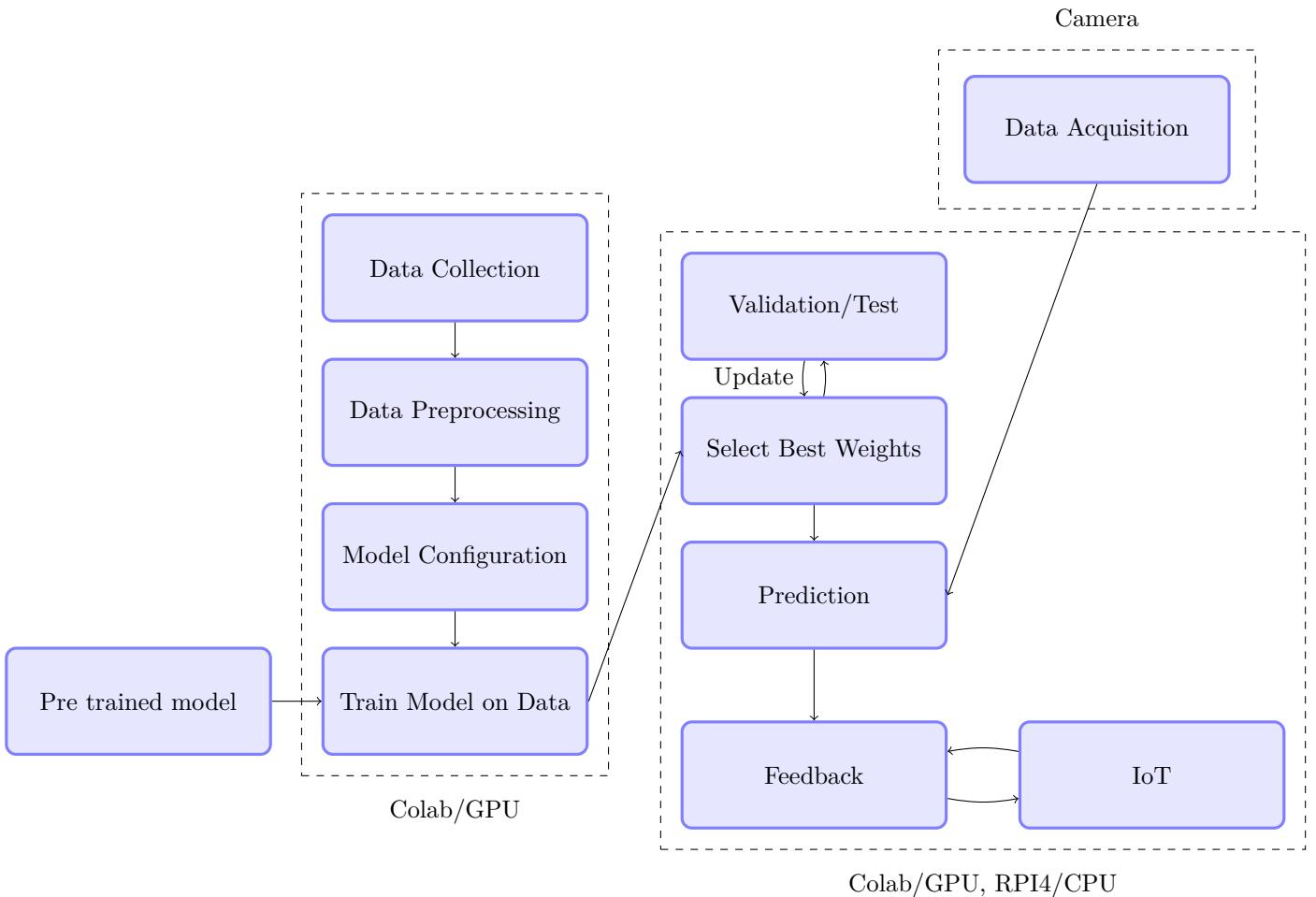


Figure 3.2: Global architecture

3.4 Data Acquisition

The Data Acquisition module is designed for optimal camera placement, either on a chest holder or a head-mounted device, to capture the most relevant environmental data for our system. This setup allows the Kinect camera to continuously collect RGB images and depth information from the user's surroundings. The chest or head mounting ensures a stable and consistent field of view, crucial for accurate object detection and depth estimation. The RGB images provide visual context, while the depth data adds a critical third dimension, enabling our model to not only identify objects but also determine their distance from the user. This real-time data stream is essential for the system's ability to assist visually impaired users in navigating their environment safely and effectively.

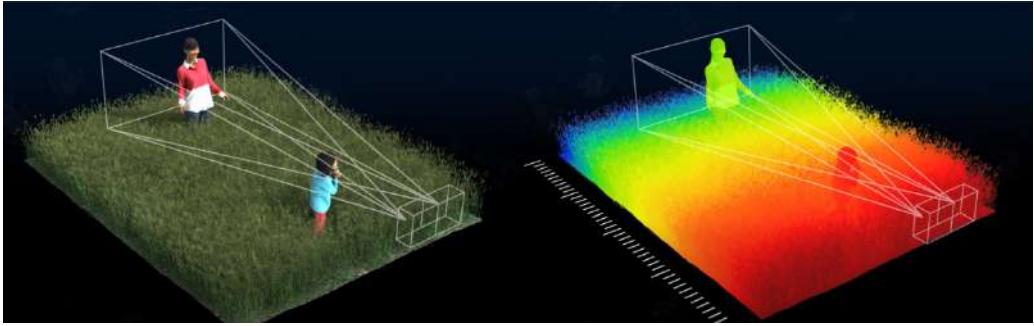


Figure 3.3: RGB Image VS Depth Image

3.5 Data preparation

3.5.1 Data collection

Choosing a dataset was not a straightforward task, as we aimed to create a large database of images for detecting outdoor objects. We considered using object detection performance tests to aid in building the database, but this proved challenging. Our goal was to simplify the labeling process and assist people in navigating outdoor environments more freely. Initially, we attempted to select subclasses from ImageNet, OpenImages, or COCO. However, due to the enormous size of these databases, we were unable to load them, making it difficult to create new subclasses. Next, we tried merging the PASCAL VOC database with MS COCO, but we encountered issues with overlapping classes and significant differences between samples within each class. Following this, we explored datasets specifically designed for the visually impaired. Fortunately, we discovered a dataset that includes the most crucial items commonly found on roads, captured from the perspective of an average person. This dataset, named '**WOTR A Dataset for the Visually Impaired Walk on the Road**' [77], contains 13,928 images , which includes 15 types of common obstacles and 5 types of road judging objects 3.4, Covering two major scenarios of walking on the road and crossing the road covering that individuals with visual impairments should be aware of, whether to avoid or follow. Additionally, we added a class for potholes [78] The dataset contains **665 images with bounding box annotations** provided in PASCAL VOC format , which pose a significant help to visually impaired pedestrians.

Label	Total		Train		Test		Val	
	Image	BBox	Image	BBox	Image	BBox	Image	BBox
Tree	6462	22515	4322	15613	1120	3667	1020	3235
Reflective cone	1308	4125	840	2549	243	819	225	757
Ashcan	2267	2857	1495	1869	405	524	367	464
Warning column	3118	10431	2115	7099	514	1681	489	1651
Roadblock	1093	4402	741	2978	185	728	167	696
Pole	8569	31144	5610	21120	1536	5240	1423	4784
Fire hydrant	1306	1384	856	905	232	246	218	233
Pedestrian	7708	35245	5069	23288	1349	6099	1290	5858
Dog	786	1022	511	665	148	193	127	164
Bicycle	3101	5995	2076	4015	530	992	495	988
Bus	1360	1788	855	1118	254	358	251	312
Truck	2555	3537	1653	2289	478	657	424	591
Car	7353	27585	4862	18372	1293	4841	1198	4372
Motorcycle	4295	12163	2859	8066	754	2205	682	1892
Tricycle	1331	1580	861	1025	248	295	222	260
Red light	3102	4961	2045	3242	550	873	507	846
Green light	2972	4965	1937	3269	548	902	487	794
Crosswalk	5268	8558	3491	5737	933	1493	844	1328
Tactile paving	1723	2381	1161	1599	296	410	266	372
Sign	2549	3360	1665	2189	467	625	417	546
Total	13928	189998	9056	127007	2534	32848	2338	30143

Table 3.1: WOTR dataset statistics
[79]

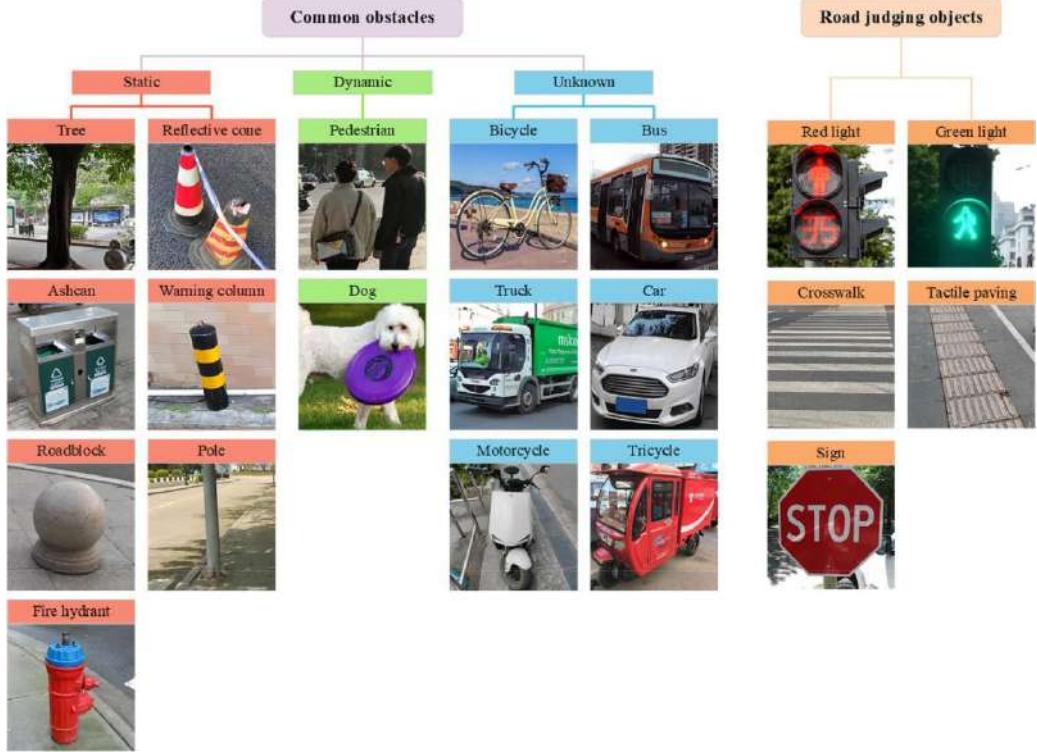


Figure 3.4: Object categories in the WOTR dataset
[79]



Figure 3.5: Some Pictures from Pothole dataset

3.5.2 Data preprocessing

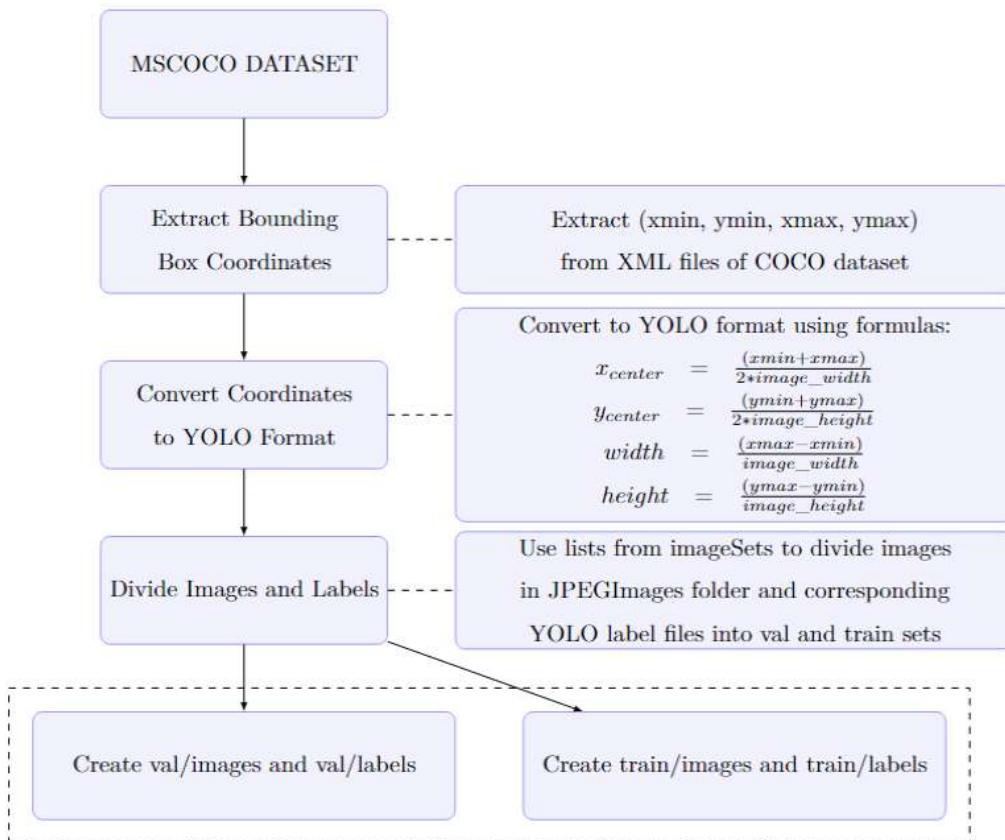


Figure 3.6: Data Preprocessing

1. Extract and Convert Bounding Box Coordinates

In the first step, the bounding box coordinates ($xmin$, $ymin$, $xmax$, $ymax$) are extracted from the XML files of the COCO dataset. These coordinates are then converted into the YOLO format, which includes the class index and normalized values for the bounding box center (x_center , y_center) and its dimensions ($width$, $height$). The resulting YOLO labels are formatted as: **class_index x_center y_center width height**.

- **x_center**: Calculated as $(xmin + xmax)/(2 \times image_width)$.
- **y_center**: Calculated as $(ymin + ymax)/(2 \times image_height)$.
- **width**: Calculated as $(xmax - xmin)/image_width$.
- **height**: Calculated as $(ymax - ymin)/image_height$.

2. Divide Images and Labels

In the second step, the images in the *JPEGImages* folder are divided into *val* and *train* sets based on the lists in the *imageSets* text files. Corresponding YOLO label text files are also divided accordingly. The result is two main directories, *val* and *train*, each containing an *images* folder with the respective images and a *labels* folder with the corresponding YOLO format label text files.

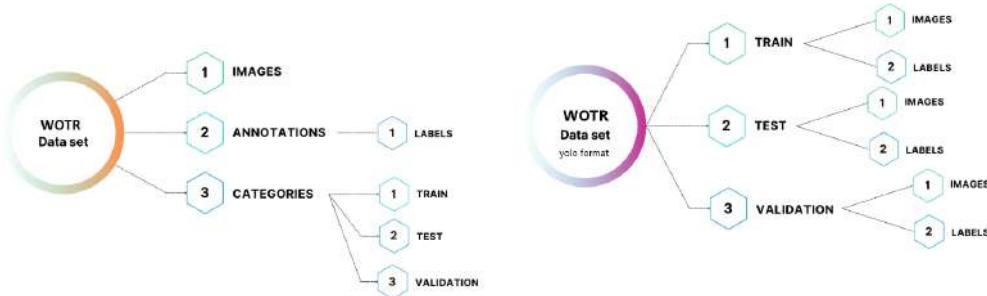


Figure 3.7: WOTR format conversion to WOTR-Yolo Format

3.5.3 challenges faced in data preparation

When collecting and processing data, several important characteristics should be considered. Here are some key points to keep in mind :

1. **Class Imbalance:** There may be a substantial difference in the number of instances for each class. Common things such as cars and people, may make up the majority of the collection but uncommon objects, such as fire hydrants and potholes, may be underrepresented.
Impact : This imbalance can lead to a model that is biased towards the more frequent classes, resulting in poor performance on the underrepresented classes.
2. **Annotation Quality and Consistency :** It may be difficult to guarantee consistent and high-quality annotations, particularly when working with big datasets. All objects must have exact bounding boxes in their annotations.
Impact : Since the model learns from these annotations, poor annotation quality may drastically reduce model performance. During training, the model may get confused by inconsistent annotations.
3. **Handling Diverse Scenarios :** The collection comprises items situated in several contexts, including varying lighting conditions, weather patterns, and backdrops.
Impact : A model that was trained on a non-diversified dataset may not be able to adapt adequately to novel or untested situations. Robust model performance depends on ensuring that the dataset encompasses all potential situations.
4. **Object Occlusion and Overlap :** In real-world photographs, objects often obscure one another. For instance, a bus may block a person or a bicycle.
Impact : As a result, the model may find it challenging to correctly identify and categorize obscured items. To tackle these circumstances, appropriate augmentation and annotation procedures are required.

5. Normalization and Conversion Accuracy : Precise computations are necessary to guarantee accuracy when converting bounding box coordinates from the source format (such as COCO) to the YOLO format.

Impact : Any mistakes made during this conversion procedure might result in inaccurate bounding box estimations, which would be detrimental to the model's functionality.

6. File Management and Organization : It might be difficult to manage the whole dataset structure, make sure that each picture has a matching label file, and properly organize files into training and validation sets.

Impact : Inadequate file management might result in mistakes and inconsistencies during model training, such as missing files or labels that are not aligned.

7. Managing Big and Small Items : The items in the collection range in size from little ones like reflecting cones and poles to big ones like buses and trees.

Impact : To guarantee that the model can recognize items of varied sizes with accuracy, meticulous annotation and maybe distinct augmentation techniques are needed to address the different scales.

3.6 YOLOv8 architecture

YOLOv8, the latest version of the YOLO model family, is designed to handle advanced computer vision tasks such as instance segmentation, object detection, and image classification. Developed by the same team behind YOLOv5, YOLOv8 brings significant architectural improvements and an enhanced developer experience.

Key upgrades to YOLOv8's design include advancements in the head, neck, and backbone components compared to earlier YOLO models. The architecture incorporates more efficient convolutional layers, which increase both the speed and accuracy of the model. One of the standout features of YOLOv8 is its ability to handle instance segmentation, allowing it to not only detect objects but also differentiate between multiple instances of the same object within an image or video. This makes it particularly well-suited for real-time object detection in dynamic environments.

The backbone of YOLOv8, based on the Darknet-53 network, delivers superior performance over its predecessor, YOLOv7. It benefits from a larger feature map, which helps the model better capture details and improve its object detection capabilities. YOLOv8 also employs an anchor-free detection head, simplifying the process of predicting bounding boxes and further boosting detection accuracy.

In addition to its architectural improvements, YOLOv8 is built with user experience in mind. It features an intuitive API that simplifies its integration into various applications, making it accessible to both beginners and experts in the field of computer vision. The model also uses feature pyramid networks (FPN) to detect objects of varying sizes, enhancing its versatility across different scenarios.

Overall, YOLOv8's improvements in speed, accuracy, and ease of use make it a powerful tool for tackling the latest challenges in object detection and instance segmentation.[80]

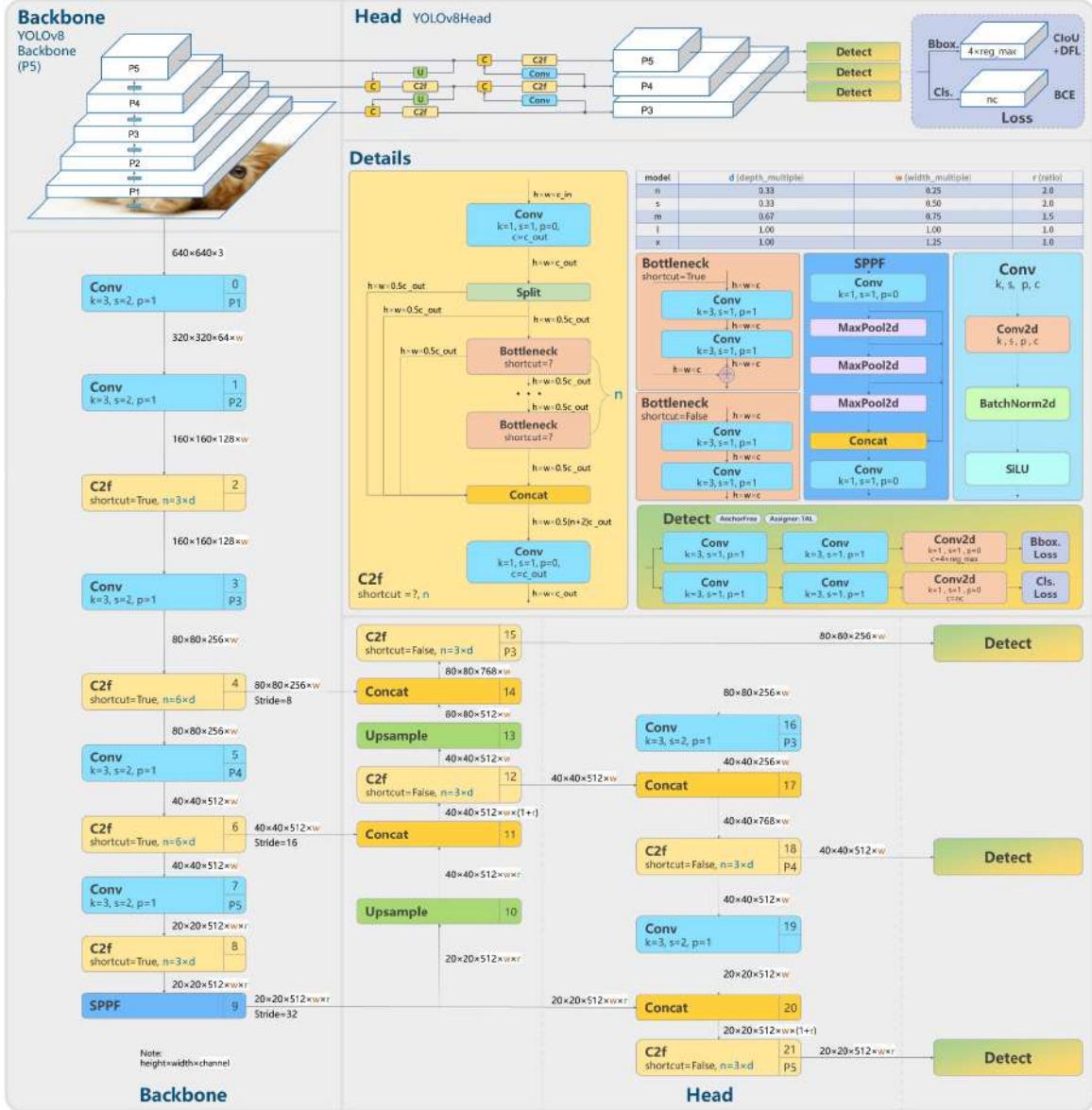


Figure 3.8: YOLOv8 Architecture.[26]

3.6.1 What are the main features in YOLOv8?

YOLOv8 introduces improvements in both its architecture and development experience.

Anchor-Free Detection In contrast to traditional models that use anchor boxes to predict offsets from predefined anchors, YOLOv8 adopts an anchor-free approach. The model directly predicts the center of an object, eliminating the need for anchor boxes. This simplification enhances the detection process and improves accuracy.

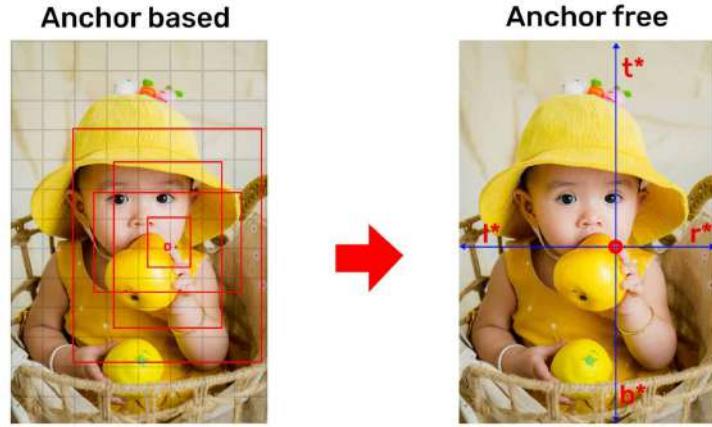


Figure 3.9: Visualization of an anchor box

New Convolutions

The convolutional layers in YOLOv8 have been significantly upgraded. The initial 6x6 convolution in the stem is replaced by a more efficient 3x3 convolution. Additionally, the main building block has been modified, with C2f substituting C3. The C2f module concatenates the outputs from all Bottleneck units, which include two 3x3 convolutions with residual connections. unlike C3, which only uses the output of the last Bottleneck unit. This change improves feature extraction and model performance.

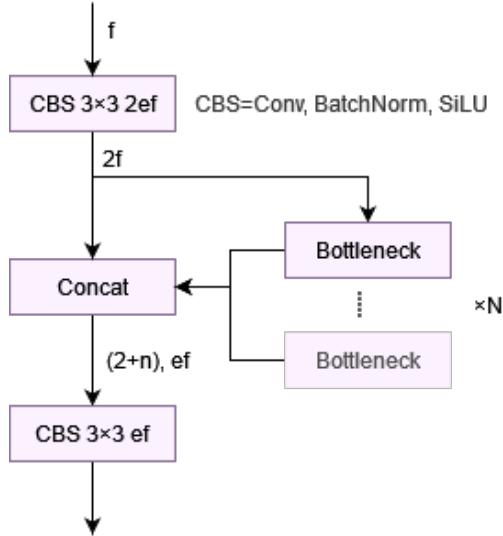


Figure 3.10: New YOLOv8 C2f module [27]

Modified Kernel Size

Although the Bottleneck structure remains the same as in YOLOv5, YOLOv8 changes the kernel size of the initial convolution from 1x1 to 3x3. This adjustment brings the model closer to the ResNet block architecture, improving its ability to capture detailed features in the image.

Efficient Feature Concatenation

In the neck section of YOLOv8, features are concatenated directly without preserving the same channel dimensions. This reduces the number of parameters and the overall tensor size, making the model more efficient and contributing to faster inference times.

3.7 Model configuration

The model setup includes the backbone network architecture, feature pyramid construction, anchor sizes and ratios, loss functions, input image size, confidence thresholds, and post-processing methods such as Non-Maximum Suppression (NMS). Further details are illustrated in the figure 3.8. These setup parameters are critical because they control the model's general behavior during inference and training, as well as its speed, accuracy, and performance. The accuracy and efficiency of the model's object detection may be greatly impacted by fine-tuning and refining the model design.

3.7.1 choose the best model

There are five YOLOv8 models. Table 3.11 illustrates the accuracy and key metrics of these model's performance on the MSCOCO dataset .

Model	mAP <i>val</i> \ 50-95	Speed ONNX(<i>ms</i>)	Speed A100 TensorRT (<i>ms</i>)	params (<i>M</i>)	FLOPs \ (<i>B</i>)
YOLO__v8_nan o	37.3 \%	80.4	0.99	3.2	8.7
YOLO__v8_sma ll	44.9 \%	128.4	1.20	11.2	28.6
YOLO__v8_me dium	50.2 \%	234.7	1.83	25.9	78.9
YOLO__v8_larg e	52.9 \%	375.2	2.39	43.7	165.2
YOLO__v8_X large	53.9 \%	479.1	3.53	68.2	257.8

Figure 3.11: YOLOv8 COCO evaluation[28].

Figure 3.11 provided from the official Ultralytics website illustrates the variations among five YOLOv8 models in terms of efficiency, speed, FLOPS (floating-point operations per second), and the number of parameters. Each model offers a different balance between these factors:

- **Smaller Models:** These models, while faster in execution, tend to have fewer parameters and lower FLOPS, which results in quicker processing times but reduced accuracy. The trade-off here is that although they are ideal for applications requiring high-speed performance, they may not achieve the level of precision needed for detailed object detection tasks.
- **Larger Models:** In contrast, larger YOLOv8 models have more parameters and higher FLOPS, leading to greater accuracy in detection. However, this increased accuracy comes at the expense of

slower execution speeds. These models are more suitable for scenarios where precise object identification is critical, but they may not meet real-time performance requirements due to their longer processing times.

Given that our goal is to assist visually impaired individuals, we need to strike a balance between speed and accuracy. Therefore, the medium-sized YOLOv8 models are selected. These models offer a reasonable compromise, providing sufficient accuracy without the high computational demands and longer processing times of the larger models. This balance ensures that the system can perform effectively in real-time scenarios while still delivering the precision necessary for reliable object detection and navigation assistance.

3.7.2 Augmentation

Augmentation settings are crucial for enhancing the training dataset, which, in turn, improves the effectiveness, speed, and accuracy of YOLO models. By transforming the training data, these settings help generate a more diverse and representative dataset, leading to better model performance. Key variables in augmentation include transformation type, intensity, probability, additional characteristics, dataset size, composition, and task specificity.

A well-designed augmentation strategy involves careful adjustment and testing of these variables to create a robust training dataset. This process helps in training a more effective YOLO model by increasing the variety of the data.

Some of the augmentation parameters used include:

- **HSV-Hue, HSV-Saturation, and HSV-Value:** Adjustments to the hue, saturation, and value of images to enhance color diversity.
- **Rotation:** Rotating images to provide different viewing angles.
- **Translation:** Shifting images horizontally or vertically to simulate movement.
- **Scale:** Changing the size of objects to introduce variability in object size.
- **Shear:** Applying shear transformations to alter the shape of objects.
- **Perspective:** Adjusting the perspective to simulate different viewpoints.
- **Flip Up-Down and Flip Left-Right:** Flipping images to create mirror effects.
- **Mosaic:** Combining multiple images into a single mosaic to increase dataset variety.
- **Mixup:** Blending two images to create a new, combined image.
- **Segment Copy-Paste:** Copying and pasting segments from one image to another to enhance object diversity.

The precise values assigned to each parameter determine the extent of the modifications applied to the images during training. By fine-tuning these augmentation parameters, researchers and practitioners can effectively improve the quality and diversity of the training dataset. This results in YOLO models that are more effective and accurate in object recognition tasks, making them more valuable for practical applications.

3.8 Train model on our data

3.8.1 Hyperparameter Choices to Train YOLOv8

Here are some key points explaining our hyperparameter choices during training:

- **Batch Size:** The batch size refers to the number of samples processed in a single training iteration. It affects both training time and memory usage. While a larger batch size can speed up training, it requires more memory. We have chosen a batch size of **16**.
- **Learning Rate:** The learning rate controls the step size during the optimization process. A high learning rate may cause the model to overshoot the optimal solution, whereas a low learning rate might lead to slow convergence. We set the learning rate to **0.01**.
- **Image Size:** The size of the images used for training impacts both model performance and training time. Larger image sizes can enhance model accuracy but demand more memory and processing power. We use an image size of **640x640**.
- **Number of Epochs:** The number of epochs indicates how many times the entire dataset is passed through the model during training. We have set the number of **epochs to 40**.
- **Data Augmentation:** Techniques such as random cropping, flipping, rotation, and color distortion are used to increase the diversity of the training data and improve the model's robustness.
- **Weight Decay and Momentum:** These regularization techniques help prevent overfitting and enhance the model's generalization. We use a **weight decay of 0.0005 and a momentum of 0.8**.
- **Optimizer:** The optimizer determines the algorithm used for training. We employ the **Adam** optimizer, which integrates the benefits of *Adagrad* and *RMSprop*. The **optimizer** is set to **auto**.

Other factors that may affect the training process :

Key	Description and Value
model	we set the Path to model file, Value= yolov8m.pt
data	Set the Path to data file, Value=data盲ind-oor_v1500.yaml
patience	Epochs to wait for no observable improvement for early stopping of training, Value= 50
save	Save train checkpoints and predict results, Value=True
save_period	Save checkpoint every x epochs (disabled if < 1), Value= -1
cache	True/ram, disk or False. Use cache for data loading, Value+ False
device	Device to run on, i.e. cuda device=0 or device=0,1,2,3 or device=cpu, Value= None
workers	Number of worker threads for data loading (per RANK if DDP), Value= 8
project	Project name, Value= None
name	Experiment name, Value= None
exist_ok	Whether to overwrite existing experiment, Value= False
pretrained	Whether to use a pretrained model, Value= False
verbose	Whether to print verbose output, Value= False
deterministic	Whether to enable deterministic mode, Value= True
single_cls	Train multi-class data as single-class, Value= False
rect	Rectangular training with each batch collated for minimum padding, Value= False
cos_lr	Use cosine learning rate scheduler, Value= False
resume	restarting the last checkpoint's training, Value= True
amp	Automatic Mixed Precision (AMP) training, choices=[True, False], Value= True
fraction	Dataset fraction to train on (default is 1.0, all images in train set), Value= 1.0
profile	Profile ONNX and TensorRT speeds during training for loggers, Value= False
lr0	Initial learning rate (i.e. SGD=1E-2, Adam=1E-3), Value= 0.01
lrf	Final learning rate (lr0 * lrf), Value= 0.01
warmup_epochs	Warmup epochs (fractions ok), Value= 3.0
warmup_momentum	Warmup initial momentum, Value= 0.8
warmup_bias_lr	Warmup initial bias lr, Value= 0.1
box	Box loss gain, Value= 7.5
cls	Cls loss gain (scale with pixels), Value= 0.5
dfl	Dfl loss gain, Value= 1.5
pose	Pose loss gain (pose-only), Value= 12.0
kobj	Keypoint obj loss gain (pose-only), Value= 2.0
nbs	Nominal batch size, Value= 64
overlap_mask	Masks should overlap during training (segment train only), Value= True
mask_ratio	Mask downsample ratio (segment train only), Value= 4
val	Validation and testing are integral parts of the training process, Value= True

Table 3.2: YOLOv8 Training Configuration.[29]

3.8.2 Loss Function

The YOLOv8 loss function integrates several components to address various aspects of object detection. It includes three main elements:

1. **Localization Loss:** This measures the difference between the predicted and actual bounding box coordinates (x, y, width, height). YOLOv8 employs either Mean Squared Error (MSE) or Smooth L1 loss to evaluate localization accuracy.

2. **Confidence Loss:** This assesses the accuracy of the predicted objectness score for each bounding box, using binary cross-entropy loss to compare the predicted score with the ground truth.
3. **Class Loss:** This component handles the classification task by calculating the difference between predicted class probabilities and the actual class labels, using categorical cross-entropy loss.

The overall loss function is a weighted sum of these components, with weights adjusted according to their importance and impact on the final loss. The significance of each component can be modified by scaling their respective losses.

3.8.3 training procedure

After configuring the YOLOv8 model, the training process unfolds as follows:

1. Feed the preprocessed images into the YOLOv8 model.
2. Calculate the overall loss using the designated loss function.
3. Use backpropagation to compute the gradients.
4. Update the model's weights with the chosen optimizer and save the weights that perform best.
5. Repeat these steps for the specified number of epochs.
6. Finally, assess the trained model on a separate test set to evaluate its performance

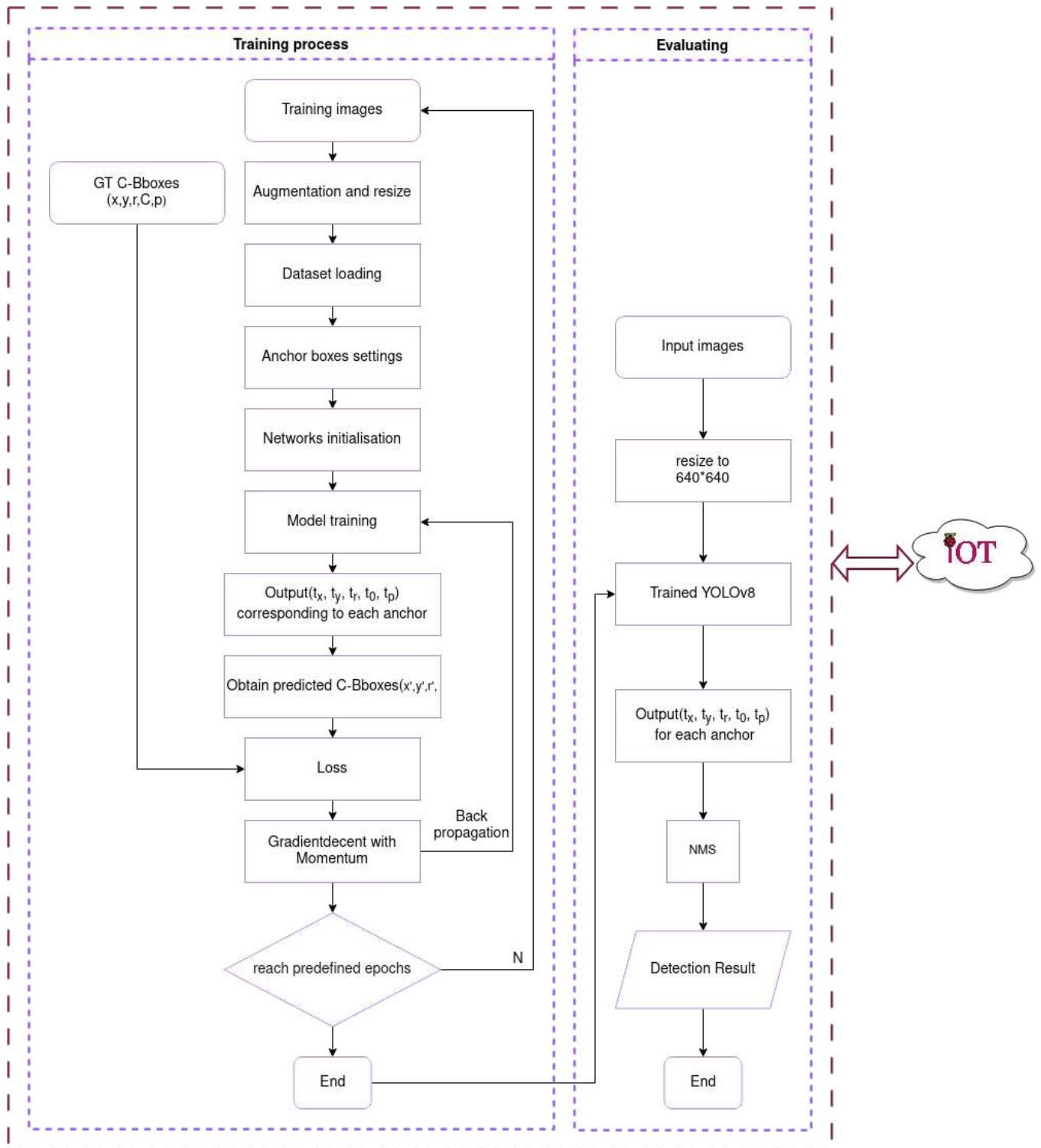


Figure 3.12: training procedure

3.8.4 Challenges Faced During Training

During the training process with our data, we faced several significant challenges :

- Resource Limitations:** Handling large datasets proved difficult due to limited computational resources.
- Inconsistencies Between Drive and Colab:** We faced synchronization issues between Google Drive and Colab, which disrupted the workflow.

3. **Colab's GPU Limitations:** Colab's GPU sessions would automatically terminate after a few hours, requiring substantial time to restart and continue training.
4. **Server Training Delays:** When attempting to train on a server, the process was extremely slow and often unresponsive.
5. **Time Constraints:** Finally, the time was the big challenge that we face.

3.9 metrics

- **True Positives (TP):** Instances where the model correctly identifies a positive sample (e.g., car) as positive.
- **False Positives (FP):** Instances where the model incorrectly classifies a negative sample (e.g., other objects) as positive.
- **True Negatives (TN):** Instances where the model correctly identifies a negative sample (e.g., other objects) as negative.
- **False Negatives (FN):** Instances where the model incorrectly classifies a positive sample (e.g., car) as negative.
- **Mean Average Precision (mAP):** mAP is a frequently used statistic for measuring the performance of object identification algorithms like YOLOv8. It tests the model's capacity to reliably recognize objects by evaluating both accuracy and recall across multiple Intersection over Union (IoU) criteria. A higher mAP score denotes greater detection accuracy, suggesting that the model is successful at properly detecting and localizing objects inside pictures.
- **Intersection over Union (IoU):** IoU estimates the overlap between the predicted bounding boxes and the ground truth bounding boxes in object detection tasks. It is determined by dividing the area of intersection between the two boxes by the area of their union. IoU serves as a critical parameter for determining whether an anticipated detection should be classed as a true positive or a false positive.
- **Precision** refers to the fraction of successfully detected items out of all objects predicted by the model. It is derived by dividing the number of genuine positives by the total number of true positives and false positives.

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

Recall : also called as sensitivity, estimates the fraction of properly detected items out of all real objects contained in the collection. It is obtained by dividing the number of true positives by the total number of true positives and false negatives.

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

These metrics are essential for assessing the efficacy of the YOLOv8 model in object identification. They give insights on the model's accuracy, precision, recall, and general detection skills. By assessing these

indicators, you may discover the model's strengths and areas for development, allowing for tweaks that can boost its performance.

3.10 IOT module

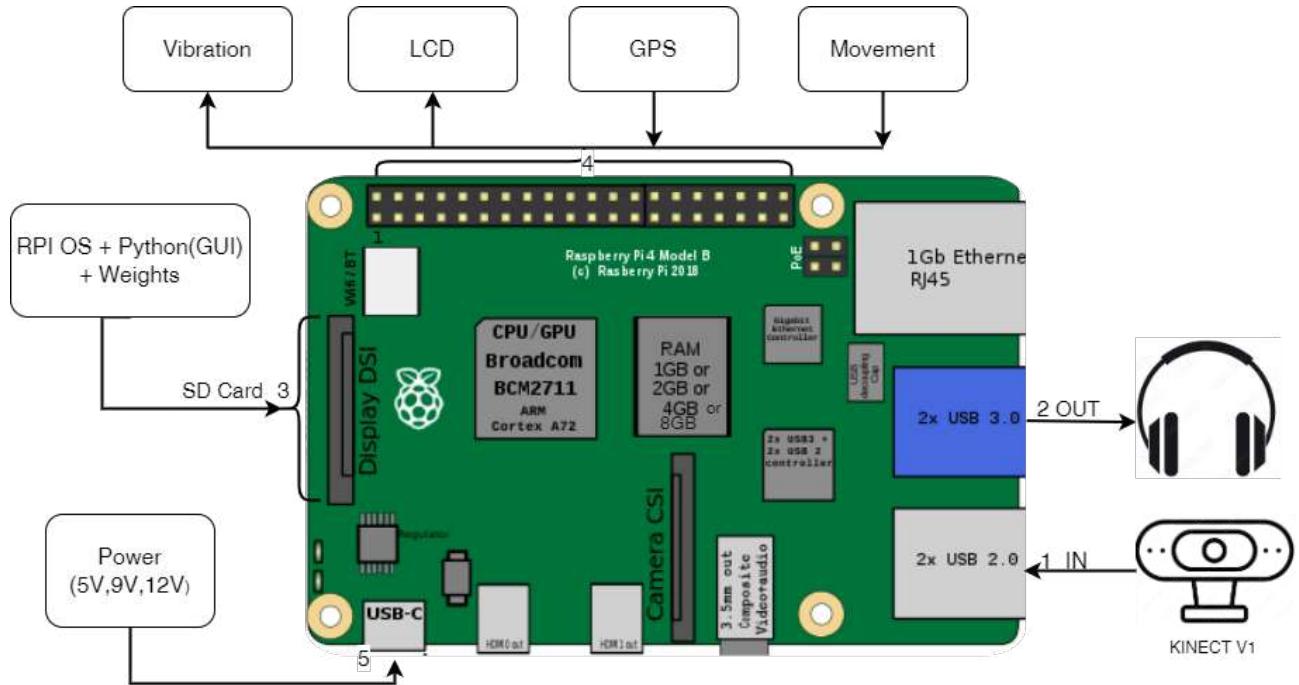


Figure 3.13: System architecture (IOT)

3.10.1 Sensors

Sensor	Definition
kinect Camera (1)	uses an optical sensor to collect visual data or pictures, allowing for image processing, visual identification, and surveillance.
Sonore Feedback Sensor (2)	Provides audio notifications or warnings by delivering auditory feedback or sound signals to the user.
SD Card (3)	Serves as the storage device for the RPI4's operating system (RPI OS), Python (GUI), and model weights
Vibration Feedback Sensor (4)	Produces vibrations that provide the user tactile feedback—that is, bodily feelings or warnings.
LCD (4)	Liquid crystal displays, or LCDs for short, are a kind of flat-panel display technology that produce visual images using liquid crystals.
kinect depth sensor (4)	Captures depth information by emitting light pulses and measuring the time it takes for the reflected light to return from nearby objects. The camera uses this timing information to calculate the distance between itself and the object based on the speed of light

Table 3.3: Sensors for the IoT System

3.11 Conclusion

In this chapter, we meticulously analyzed the training process of the YOLOv8 model and the data preparation for a system designed to assist vision impaired persons outside. We examined the system architecture, including data collection and preprocessing, specific attributes of YOLOv8, model setup, and the training procedures. Furthermore, we examined the prediction methodology, assessment metrics, and the incorporation of an IoT module to augment the system's capabilities. This chapter provides a thorough overview of the system's design and execution. The findings of this study will be addressed in the subsequent chapter .

Chapter 4

Implementation and Results

4.1 Introduction

An integral part of the software development process, a development environment provides developers with the necessary hardware and software tools to design, create, and test programs. It includes software tools such as text editors, programming languages, frameworks, and integrated development environments (IDEs), as well as hardware components like computers, displays, and peripherals.

We will provide a summary of the development environment that we employed for our project in this chapter. Let's start by talking about hardware configuration and stressing the value of a dependable workstation. We will next go into detail on the software tools that were essential to the development process, such as the particular IDEs, libraries, and frameworks that made it possible for us to create and test our application quickly.

Furthermore, we will delineate the distinct stages of system design and implementation, offering discernments into the used techniques and approaches. The difficulties we ran into with hardware constraints, software incompatibilities, and debugging complexity will also be covered in this chapter. Ultimately, we will outline our accomplishments to date, emphasizing the significant turning points attained and their role in the project's overall success.

This exploration not only highlights the technical aspects of the development environment, but also emphasizes its critical role in the successful delivery of a robust and functional system.

4.2 Development environment

4.2.1 Hardware Environment

Google Colab:

Google Colab^{4.1} is a cloud-based platform designed for interactive Python programming with a notebook interface, ideal for machine learning and data analysis. It integrates well with frameworks like PyTorch, TensorFlow, and Keras. Colab offers three runtime environments—CPUs, GPUs, and TPUs—with a

maximum of 12 hours of uninterrupted runtime before the virtual machine resets due to Colab's resource management policies.

- **Central Processing Unit (CPU) in Google Colab:** Colab's CPU features a 64-bit x86 architecture, suitable for various computing tasks within the virtual environment.
- **GPU in Google Colab:** The Tesla T4 GPU, available in Colab, is designed for intensive computing tasks, particularly in deep learning and scientific simulations. It features Tensor Cores optimized for deep learning and supports PyTorch, CUDA, and TensorFlow frameworks.
- **Tensor Processing Unit (TPU) in Google Colab:** Colab also provides access to TPUs, specialized hardware designed by Google for accelerating deep learning tasks. TPUs are optimized for high-speed matrix operations and can significantly enhance the performance of large-scale machine learning models, particularly those built with TensorFlow.



Figure 4.1: Google Colab Logo

Google Drive

Google Drive is a cloud storage service from Google that allows users to securely store, organize, and access files from any internet-connected device. Additionally, Google Drive offers a range of storage plans to accommodate different needs. [81]

In our setup, Google Drive is used to store datasets and model weights, ensuring these critical resources are safely backed up and easily accessible. To streamline the workflow, we have connected our Google Colab notebook with Google Drive. This integration facilitates seamless transfer of files between Colab and Drive, enhancing our efficiency in developing and refining models. Additionally, Google Drive offers a range of storage plans to accommodate different needs.

RPI4

The Raspberry Pi 4 Model B, developed by the Raspberry Pi Foundation, offers significant upgrades over previous models, making it highly versatile. Key features include:

- **Enhanced Performance:** Equipped with a 1.5 GHz quad-core Cortex-A72 CPU, it delivers improved speed for complex tasks.
- **Memory Options:** Available in 2GB, 4GB, and 8GB RAM variants to suit different needs.

- **Advanced Connectivity:** Features dual-band 802.11ac Wi-Fi, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and dual micro-HDMI ports for high-definition displays.
- **GPIO Expansion:** Retains a 40-pin GPIO connector for hardware integration.
- **Storage Options:** Supports microSD cards and external storage via USB 2.0 and 3.0 ports.
- **OS Compatibility:** Works with Raspbian (now Raspberry Pi OS), Ubuntu, and other Linux distributions.
- **Cooling and Power:** Requires adequate cooling and a 5V/3A USB-C power supply due to its increased performance.

In our project, we utilize the Raspberry Pi 4 to implement a solution that combines the YOLOv8 object detection algorithm with depth-sensing capabilities. This setup provides real-time audio feedback to visually impaired users, enhancing their ability to navigate their environment safely and effectively.

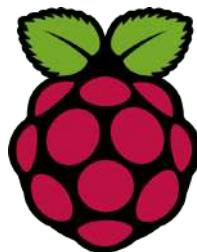


Figure 4.2: RPI4 logo

4.2.2 Software environment

Python

Python is a versatile and widely-used programming language with applications in web development, data analysis, machine learning, and scientific computing. The latest release, Python 3, brings improvements in syntax, Unicode support, memory management, and performance. Managed by the Python Software Foundation, Python has a rich ecosystem of libraries and frameworks designed for various use cases. [82]



Figure 4.3: Python logo

Torch

PyTorch, sometimes referred to as Torch, is an open-source machine learning framework that is mostly intended for deep learning uses. Because of its dynamic computational architecture that allows for real-time modifications, it offers both flexibility and efficiency for building and training neural networks. PyTorch is an excellent choice for managing large deep learning projects because of its superior GPU support. The framework simplifies the model-building process, offers numerous customization options, and comes

with a number of libraries for various purposes. PyTorch's user community, thorough documentation, and easy-to-use APIs are what make it so popular. It is always being improved and added to with new features. [83] [84]



Figure 4.4: PyTorch logo

OpenCV

OpenCV is a robust open-source library designed for computer vision and machine learning. It supports various programming languages and provides a broad range of functions for tasks such as image processing, object detection, and camera calibration. Engineered for real-time performance, OpenCV includes modules for machine learning and is extensively employed in fields like robotics, augmented reality, and surveillance. Available under the BSD license, OpenCV is freely accessible for use and modification. [85].



Figure 4.5: OpenCv logo

PyCharm

PyCharm is a dedicated integrated development environment (IDE) tailored specifically for Python programming. It offers a comprehensive suite of tools and features designed to boost productivity, such as advanced code editing, debugging, testing, and version control integration. Created by JetBrains, PyCharm is celebrated for its intuitive interface and robust capabilities. It supports a variety of popular Python frameworks and libraries, and provides options for customization, ensuring compatibility across multiple operating systems.[86].



Figure 4.6: PyCharm logo

4.3 Overview of the Assistive Navigation System for the Visually Impaired

4.3.1 Overview

Navigating busy urban environments presents significant challenges and stress for individuals who are visually impaired. Without the ability to see, these individuals must depend heavily on their remaining senses—hearing, touch, and memory—to traverse a world filled with unpredictable obstacles. Each step can be fraught with uncertainty as they navigate potential hazards such as tripping over unseen objects, colliding with poles, or misjudging distances to curbs. While traditional tools like the white cane offer some assistance, they provide limited information about the surroundings, often only detecting objects in immediate proximity.

To address these limitations, we propose an advanced assistive navigation system that leverages deep learning and Kinect technology to enhance spatial awareness and navigation. The core of the system involves a camera mounted on a wearable device, which uses a sophisticated deep learning model to detect and identify objects in real-time. The integration of a Depth Camera sensor further enriches the system by providing accurate depth information, enabling precise distance measurement.

When an object is detected within a 1-meter range, the system triggers a voice assistant that provides a clear and informative alert. For instance, the assistant might announce, "There is a pole 0.8 meters in your way, slightly to your right," offering both the distance to and direction of the obstacle. This verbal guidance helps the user understand the location of obstacles relative to their current position, whether directly ahead, to the right, or to the left.

For objects located beyond the 1-meter range, the system remains silent to avoid unnecessary interruptions. Users can interact with the system using voice commands such as "start," "stop," and "detect just my way," allowing them to control the system's functionality as needed. Additionally, the system allows users to adjust the distances for the first and second warnings, providing flexibility to accommodate personal preferences and environmental conditions.

This combination of real-time object detection, depth-sensing, and auditory feedback provides a three-dimensional understanding of the environment. It enables visually impaired individuals to navigate more safely and confidently, significantly reducing the stress and fear associated with traversing spaces they cannot see. The system empowers users to move through their surroundings with increased independence, transforming the way they experience and interact with the world.

4.3.2 User Interface (UI)

The user interface plays a crucial role in allowing the user to interact with the system, select detection modes, adjust alert thresholds, and control the object detection process.

1. Mode Selection :

- **Interaction:** The user can select between different detection modes: "Canter focus" or "Full focus."

- **Effect:** The mode selection dictates the detection behavior and alert logic, either focusing solely on the user's direct path or including objects detected on the sides as well.

2. Alert Thresholds :

- **Interaction:** The user can adjust the first and second alert thresholds via comboboxes.
- **Effect:** These thresholds alter the sensitivity of the system to detected objects, determining when alerts are triggered based on proximity.

3. Control Buttons :

- **Interaction:** Start and Stop buttons enable the user to control the object detection process.
- **Effect:** These buttons initiate or terminate the background object detection and alert mechanisms.

4.3.3 Object Detection

The object detection component is the core of the system, responsible for identifying objects within the camera's view and analyzing their positions and distances relative to the user.

1. Object Identification :

- **Interaction:** The YOLO model processes the camera feed to detect and classify objects.
- **Effect:** The system identifies the objects and their types, such as persons, vehicles, or signs, which is essential for subsequent alert generation.

2. Position Analysis :

- **Interaction:** The system analyzes the position of detected objects in relation to the user's path, determining whether they are on the left, right, or directly ahead.
- **Effect:** This analysis enables the system to generate direction-specific alerts, aiding in navigation.

3. Depth Measurement :

- **Interaction:** Depth data from the Kinect camera is used to measure the distance of objects from the user.
- **Effect:** The proximity of objects affects the timing and urgency of the alerts given to the user.

4.3.4 Alert System

The alert system is responsible for converting the detected objects and their positions into actionable audio alerts that assist the user in real-time navigation.

1. Alert Generation :

- **Interaction:** Alerts are generated based on the type, position, and distance of detected objects.
- **Effect:** The system produces spoken messages or notifications, providing guidance to the user about their surroundings.

2. Alert Thresholds :

- **Interaction:** Alerts are triggered when objects are within predefined thresholds.
- **Effect:** These thresholds control the conditions under which alerts are issued, ensuring that the user is informed of nearby obstacles without being overwhelmed by unnecessary notifications.

3. Speech Synthesis :

- **Interaction:** The system uses the pyttsx3 library to convert text-based alerts into spoken messages.
- **Effect:** The auditory feedback provided by the system allows the user to understand the nature and position of detected objects, facilitating safer navigation.

4.3.5 Camera and Data Handling

This component deals with capturing and processing visual data from the environment, which is then used for object detection and alert generation.

1. Video Capture :

- **Interaction:** The Kinect camera captures video frames in real-time.
- **Effect:** These frames provide the RGB image necessary for object detection by the YOLO model.

2. Depth Data Retrieval :

- **Interaction:** Depth data is retrieved from the Kinect camera alongside the RGB image.
- **Effect:** This data is crucial for calculating the distance of objects from the user, which influences alert generation.

3. Data Processing :

- **Interaction:** The system processes the video frames and depth data to detect objects and calculate their distances.
- **Effect:** This processing step determines the position of objects relative to the user and triggers appropriate alerts.

4.3.6 Detailed Object Interactions and Alerts

The system's alert mechanisms are finely tuned to handle various objects and scenarios, ensuring that the user receives relevant and timely information.

Person Detection :

- In the User's Path :

1. Initial Alert:

- **Condition :** A person is detected within 2 meters directly ahead.
- **Announcement :** "There is a person 2 meters in your way."

2. Close Proximity:

- **Condition :** If the person moves closer than 1 meter.
- **Announcement :** "Watch out, there is a person very close to you."
- **Additional Guidance:**
 - * If an object is detected within 1 meter on the left side, the system will add: "Move to the right."
 - * If no object is detected within 1 meter on the right side, the system will confirm: "You can move left or right."

3. Position Change :

- **Condition:** A person moves from the side into the user's path.
- **Announcement:** "A person has moved into your path."

4. No Repeated Alerts :

- **Condition:** Avoid repeating alerts unless the person moves significantly closer.

- In the User's Left/Right :

1. Position Change :

- **Condition:** Announce if a person moves from the side towards the user's path.
- **Announcement:** Only announce if the person moves directly in front of the user.

Vehicle Detection :

- In the User's Path :

1. Always Announce :

- **Condition:** Any vehicle detected directly in the user's path.
- **Announcement:** "There is a [vehicle type] [distance] in your way."
- **Additional Guidance:**

- * If an object is detected within 1 meter on the left side, the system will add: " Move to the right "
- * If no object is detected within 1 meter on the right side, the system will confirm: " You can move left or right "

- In the User's Left/Right:

1. Crowded Areas:

- **Condition:** Multiple vehicles detected on one side.
- **Announcement:** "There are [count] [vehicle type] on your [left/right]."

2. Moving Vehicles:

- **Condition:** Vehicle moving towards the user.
- **Announcement:** Always alert the user if a vehicle is approaching.

Infrastructure Detection (Poles, Trees, Fire Hydrants, etc.) :

1. **Far Away (>2m):**

Initial Detection:

- **Condition:** Object detected at a distance greater than 2 meters.
- **Announcement:** "Watch out, there is a [object type] in your way."

2. **Close Proximity (<2m):**

Distance Information:

- **Condition:** Object detected within 2 meters.
- **Announcement:** "The [object type] is [distance] meters in your way."

3. **In the User's Left/Right:**

Very Close (<1m):

- **Condition:** Object detected within 1 meter on the side.
- **Announcement:** Announce the blind about the object.

Traffic Signs and Lights :

- **Crosswalks:**

1. **Initial Detection:**

- **Condition:** Crosswalk detected.
- **Announcement:** "there is Crosswalk in your way."

2. **User Request for Crossing:**

- **Condition:** User requests assistance in crossing.
- **Response:** Guide the user to the crosswalk if detected and inform them about the light's status.

- **Light Signals:**

1. **Relevant to Crosswalk:**

- **Condition:** If red/green light is associated with a crosswalk.
- **Announcement:** "You can cross; the light is green" or "Stop; the light is red."

2. **Contextual Awareness:** **Condition:** Ignore lights not relevant to the crosswalk.

Blind Road/Signs:

1. **Detection:**

- **Condition:** Blind road or important navigation signs detected.
- **Announcement:** Provide information about hazards or navigation cues.



Figure 4.7: Case one

in this **case 4.7** the assistant announce to the person : " **Watch out, there is a ashcan in your way , There are 2 cars on your left** " because all the objects meet the condition but the second ashcan is not mentioned because it is far away and to his left.

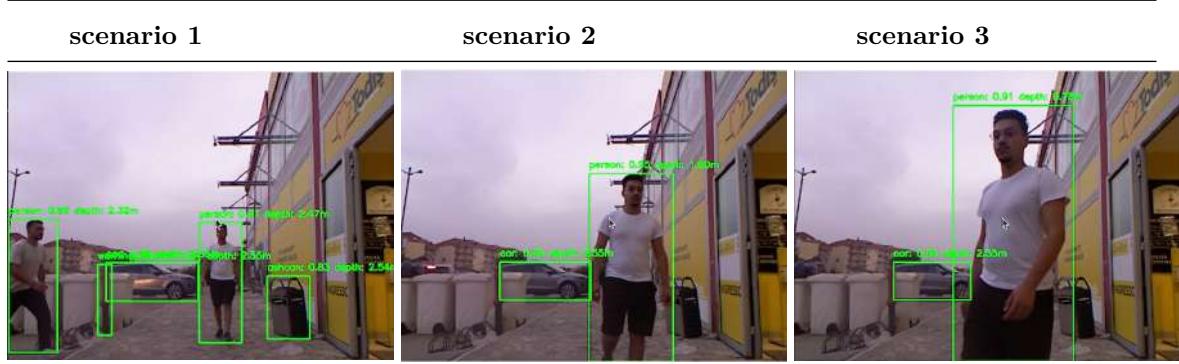


Table 4.1: Case two

scenario 1 : in this scenario 4.1 the assistant announce to the person : " **There is one car on your left** " because the object meet the condition but the person is not mentioned because it is far away ($>2m$) and other objects do not meet the condition.

scenario 2 : in this scenario 4.1 the assistant announce to the person : " **There is a person 1.8 meters in your way , There is one car on your left** " because the objects meet the condition ($1m < \text{person} < 2m$).

scenario 3 : in this scenario 4.1 the assistant announce to the person : " **Watch out, there is a person very close to you , you can move left or right , There is one car on your left** " because the object meet the condition ($\text{person} < 1m$) , Since the system does not detect any object close on his right ($<1m$), he can move either left or right. However, if there is an object detected very close on his right ($<1m$), the system tell him he should move left to avoid it.



Figure 4.8: Case tree

in this **case 4.8** the assistant announce to the person : " **Watch out, there is a roadblock, bicycle and motorcycle in your way. There are 4 cars on your right** " .

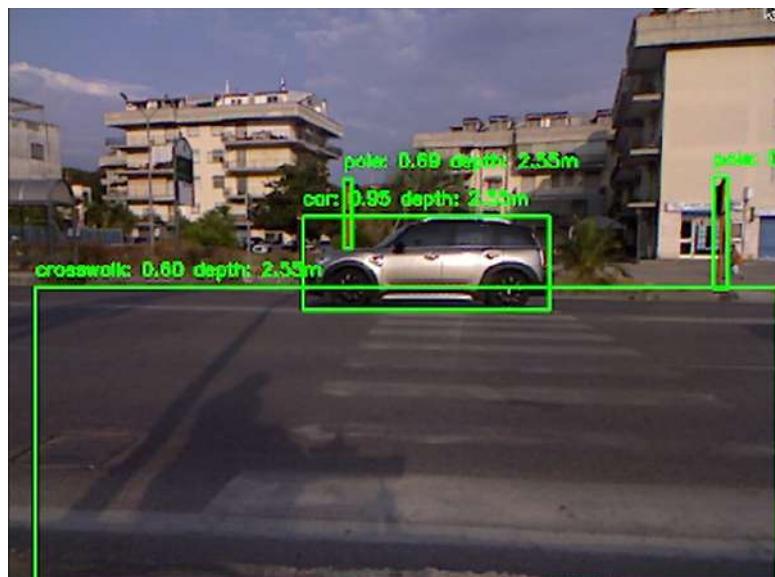


Figure 4.9: Case four

in this **case 4.9** the assistant announce to the person : " **Crosswalk in your way , Watch out, there is car and pole in your way** " , in case he detect Light Signals with the Crosswalk and the light is green the assistant tell him " **You can cross; the light is green**" and if the light is red he tell him " **Stop; the light is red**" .



Figure 4.10: Depth and RGB

Our system captures the world through two distinct modalities: RGB and depth images, as illustrated in Figure 4.10. The RGB image provides rich color and texture information, allowing the system to perceive the environment in a manner similar to human vision, identifying features like shapes, patterns, and colors. Meanwhile, the depth image adds a layer of spatial understanding by measuring the distance of objects from the camera, enabling the system to construct a three-dimensional representation of its surroundings. Together, these inputs create a comprehensive view of the environment, enhancing the system's ability to interpret and interact with real-world scenarios.

4.4 Training and validation

We trained the model with a focus and then validated its results to assess its performance. To guarantee the best possible outcomes, the training procedure was closely observed. The following are the results of this training session.

4.4.1 training results :

These are random 16 batch samples of training and validation sets :

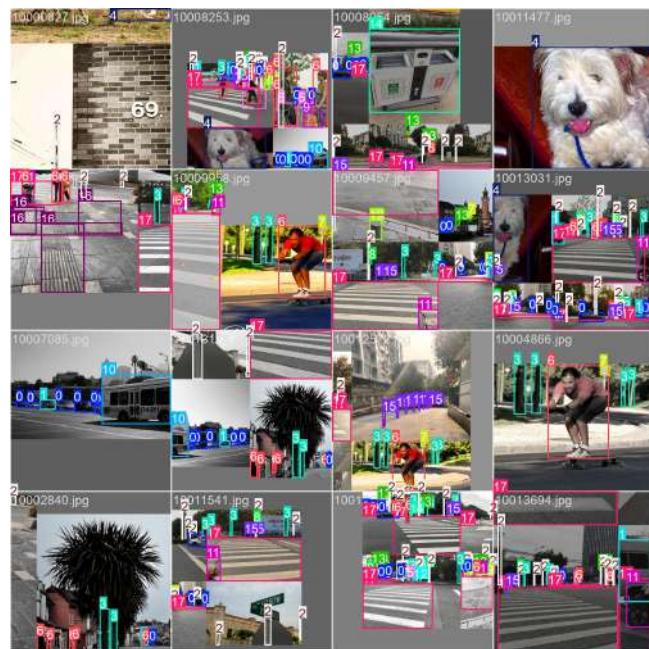


Figure 4.11: Train batch

Train batches are used to update the model's parameters. During training, the model learns from these batches by adjusting its weights to minimize the loss function. For example, in the figure 4.11.

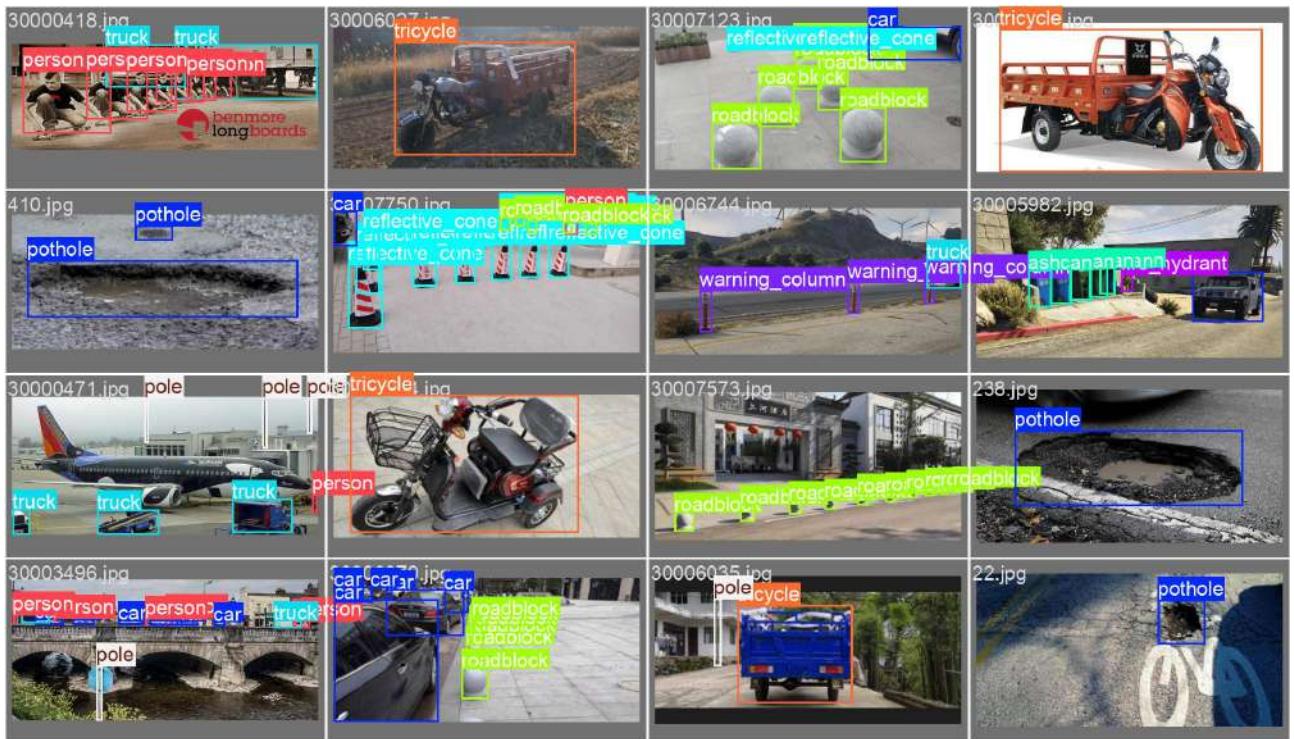


Figure 4.12: validation batches

Validation batches are used to evaluate the model's performance during training. They help in monitoring the model's progress and detecting overfitting. For instance, in the figure 4.12.

train/val metrics

these results are while the train of 40 epochs (Graphically in the figure 4.13, and statistically in the table 4.2 and also we have Confusion matrix 4.14 and Confusion Matrix Normalized 4.15) :

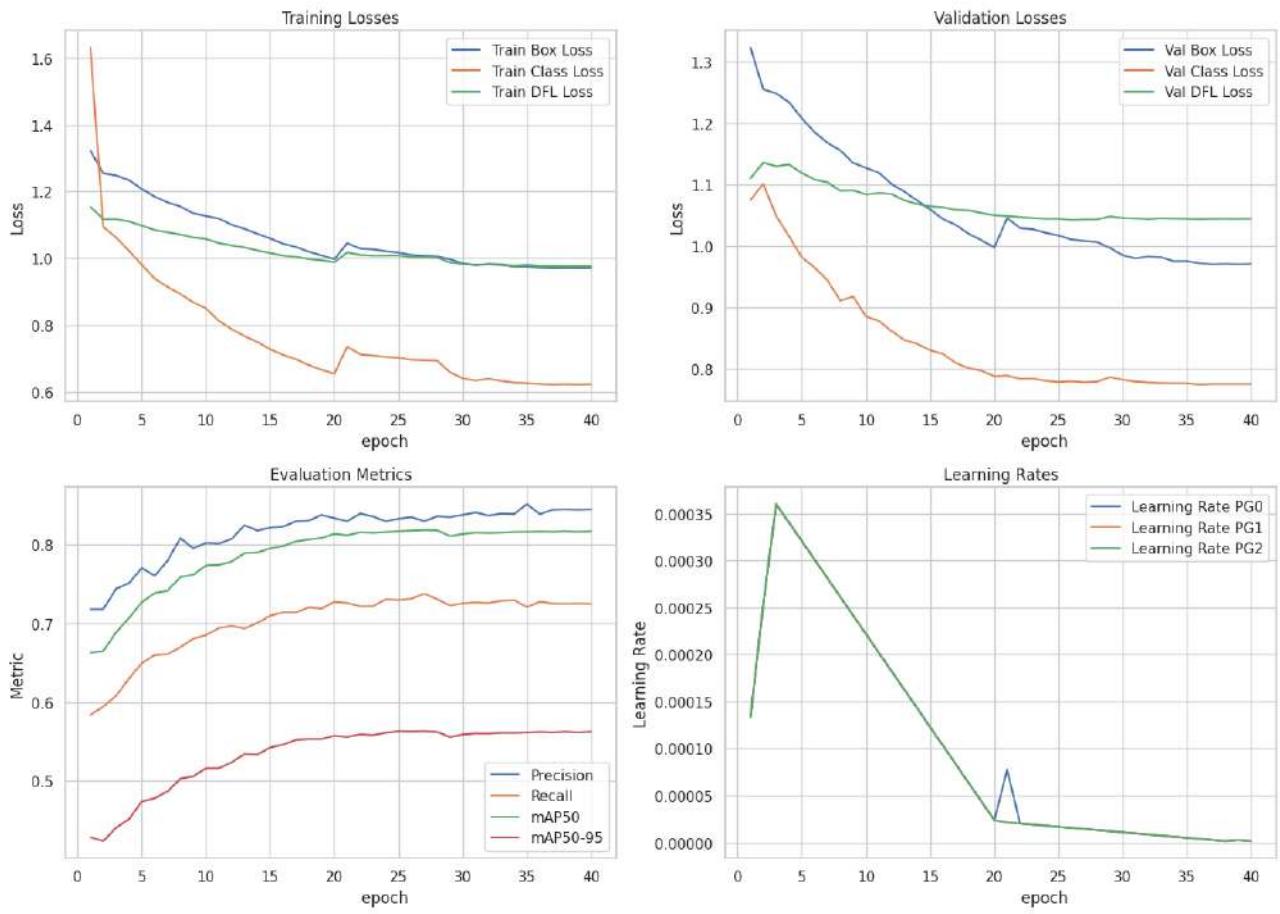


Figure 4.13: Results

Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
all	2402	30312	0.837	0.733	0.819	0.563
car	1198	4372	0.839	0.757	0.844	0.593
truck	424	591	0.687	0.538	0.612	0.441
pole	1423	4784	0.778	0.781	0.841	0.505
tree	1020	3235	0.748	0.763	0.816	0.446
dog	127	164	0.732	0.695	0.746	0.547
bicycle	495	988	0.763	0.615	0.727	0.421
person	1290	5858	0.846	0.713	0.815	0.506
sign	417	546	0.864	0.744	0.841	0.604
red light	507	846	0.871	0.835	0.898	0.574
fire hydrant	218	233	0.889	0.721	0.845	0.601
bus	251	312	0.872	0.613	0.771	0.556
motorcycle	682	1892	0.823	0.741	0.832	0.519
reflective cone	225	757	0.899	0.702	0.814	0.570
green light	487	794	0.892	0.899	0.944	0.610
ashcan	367	464	0.852	0.761	0.831	0.664
warning column	489	1651	0.890	0.715	0.833	0.485
blind road	266	372	0.844	0.836	0.870	0.671
crosswalk	844	1328	0.914	0.886	0.946	0.787
tricycle	222	260	0.879	0.719	0.793	0.656
roadblock	167	696	0.896	0.723	0.826	0.600
pothole	64	169	0.792	0.631	0.747	0.470

Table 4.2: Results of training

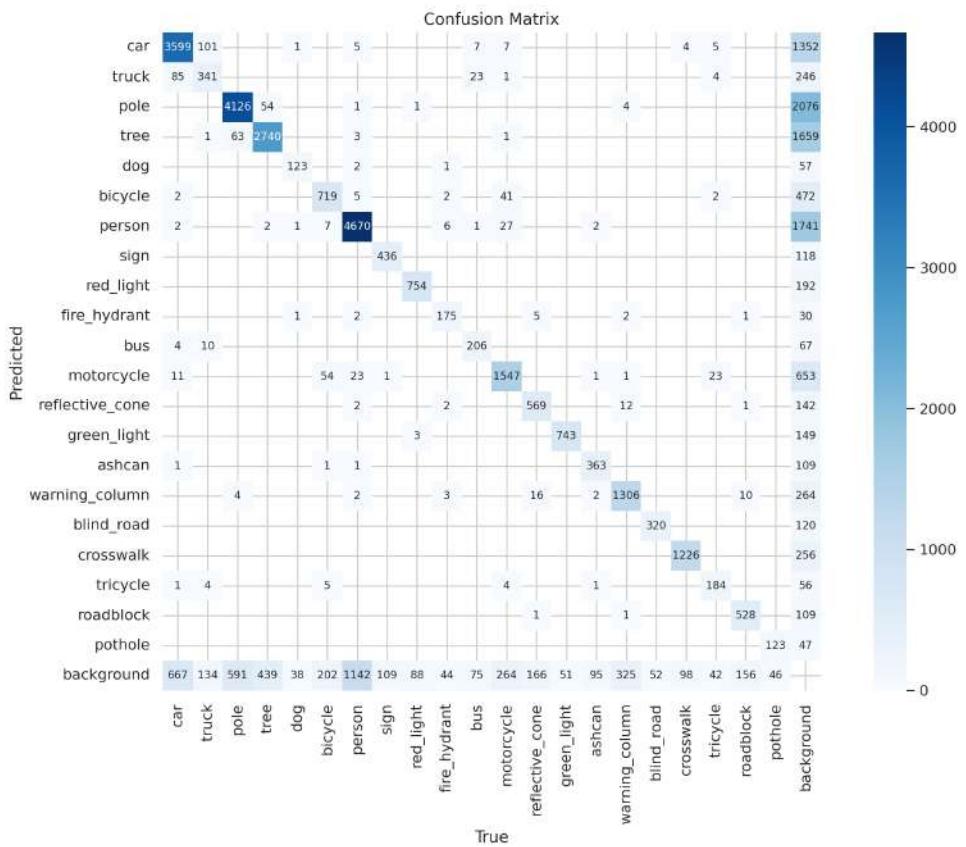


Figure 4.14: Confusion matrix

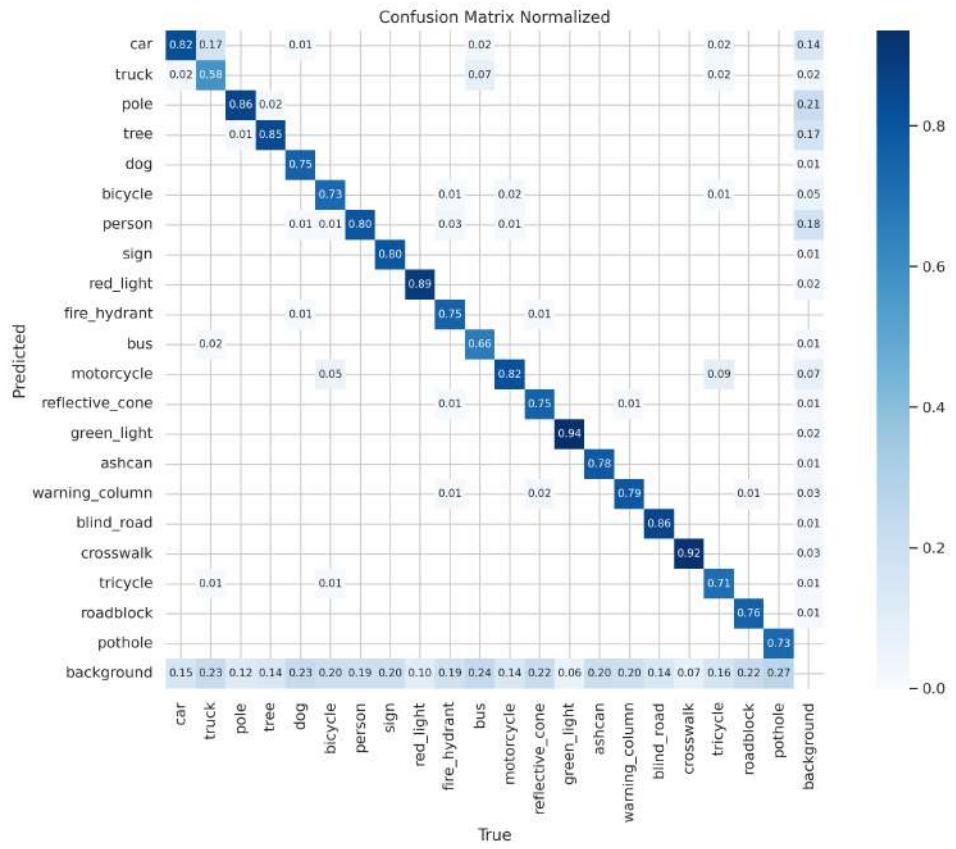


Figure 4.15: Confusion Matrix Normalized

4.5 Test, Results and Discussion

4.5.1 Test and Results

The table 4.3 below provides a comparative analysis of inferences on the same image using different weights :

Weights	Input Image	Inference	GPU Time	FLOPs
		CPU		
Best.pt		659.5ms	65.7ms	28.6

Weights	Input Image	Inference Time	GPU	FLOPs
		CPU		
yolov8n.pt		308.9ms	40.0ms	8.7
				
yolov8s.pt		789.4ms	67.86ms	28.6
				
yolov8m.pt		1264.7ms	73.18ms	78.9
				
yolov8l.pt		2763.9ms	75.38ms	165.2
				
yolov8x.pt		3585.1ms	87.18ms	257.8
				

Table 4.3: Inference times and FLOPs for different model weights on CPU and GPU

4.5.2 Discussion

The comparison of various YOLOv8 models, including the model developed in this research, provides significant insights into their performance, particularly in the context of real-time object detection aimed at assisting visually impaired individuals. This discussion analyzes the inference times, computational demands (FLOPs), and detection accuracy of each model based on a specific test scenario and demonstrates the advantages of our model.

Inference Time and Computational Complexity

The analysis of inference times and FLOPs across the YOLOv8 models highlights the trade-offs between computational efficiency and detection speed. Models with lower FLOPs, such as YOLOv8n, tend to exhibit faster inference times, particularly on CPUs, making them more suitable for environments with limited computational resources. These models prioritize quick processing but may compromise on detection accuracy due to their simpler architectures.

Conversely, models with higher FLOPs, such as YOLOv8m, YOLOv8l, and YOLOv8x, typically demonstrate slower inference times, especially on CPUs, due to their increased complexity. These models are designed to enhance detection accuracy by processing more features, but their longer inference times can be a limitation in real-time applications, where rapid feedback is crucial.

The model developed in this research, which operates with 28.6 FLOPs, demonstrates a well-balanced performance, combining efficient computational demand with robust detection capabilities. It achieves a CPU inference time of 659.5ms and a GPU inference time of 65.7ms, positioning it between YOLOv8n and YOLOv8s in terms of computational efficiency. Importantly, our model delivers competitive inference times without sacrificing detection accuracy, making it particularly suitable for real-time applications where both speed and precision are critical.

Detection Accuracy in Real-World Scenario

The test image, captured at night in a street environment with low visibility and various critical objects (a pole, fire hydrant, crosswalk, and traffic light), provided a challenging scenario for the models:

- **Our Model :** The model developed in this research demonstrated superior detection capability by accurately identifying all relevant objects, including the fire hydrant, pole, crosswalk, and green light of the traffic signal. This comprehensive detection is crucial for ensuring safe navigation for visually impaired users, making our model the most effective in this context.
- **YOLOv8n and YOLOv8s :** While both models detected some key objects, such as the fire hydrant and the traffic light, they failed to correctly identify the obstructing pole and misinterpreted other objects, potentially leading to dangerous situations for the user.
- **YOLOv8m, YOLOv8l, and YOLOv8x :** These models exhibited several false detections, such as non-existent parking meters and incorrect traffic lights, while missing critical objects like the pole in the path. Such errors make these models less reliable for outdoor navigation that demand precise and accurate object detection.

4.6 Conclusion

In this chapter, we have detailed the implementation of an object detection system developed to assist visually impaired individuals, utilizing the YOLOv8 convolutional neural network on a Raspberry Pi. The chapter began with an overview of the development environment and software tools essential for creating the system. We then provided a comprehensive overview of the Assistive Navigation System, explaining its various components and functionalities tailored for real-time object detection. Following this, we presented the testing procedures and results, highlighting the model's performance in different scenarios. Finally, a thorough discussion was conducted to analyze the system's effectiveness, emphasizing the balance between computational efficiency and detection accuracy, as well as the model's suitability for real-world applications. This chapter demonstrated the potential of integrating advanced AI models with low-cost hardware to create practical solutions for enhancing the independence of visually impaired users.

General Conclusion

In this thesis, we developed a novel solution to support visually impaired individuals in navigating outdoor environments using advanced AI-based techniques. Our system leverages YOLOv8 for real-time object detection and depth measurement, integrated with the Raspberry Pi 4 for a compact and cost-effective implementation.

Our approach distinguishes itself by not only identifying key objects but also assessing their depth, which is crucial for providing actionable spatial awareness. Depth information enhances the system's ability to guide users by offering a more nuanced understanding of their surroundings, which is particularly beneficial in complex outdoor settings.

The integration of depth sensing into our object detection framework allows for improved guidance and navigation, helping users avoid obstacles and interact more effectively with their environment. This advancement, coupled with real-time feedback through audio cues, ensures that our system offers practical and immediate assistance.

Through rigorous testing and optimization, we demonstrated that our solution effectively combines object detection and depth perception to create a reliable navigation aid. The affordability and portability of the Raspberry Pi 4 make this technology accessible, empowering visually impaired individuals with enhanced mobility and independence.

Overall, our system represents a significant step forward in assistive technology, offering a sophisticated yet practical tool for navigating outdoor spaces. We believe this work not only advances the field of AI-based assistive technologies but also holds the promise of making a meaningful impact on the daily lives of visually impaired individuals.

Perspectives:

While our system has achieved significant advancements in assisting visually impaired individuals, there are several avenues for future research and improvement:

1. **Real-world testing:** To fully validate the system's performance and usability, it should be deployed in diverse real-world environments. Gathering feedback from visually impaired users in various outdoor settings will provide insights into practical challenges and user needs, informing further refinements and improvements. This could also help identify any unforeseen issues in different weather conditions or complex environments.
2. **Integration of Optical Character Recognition (OCR):** Incorporating OCR technology could significantly enhance the system's functionality by enabling it to read and interpret text from books,

advertisements, car plates, and road signs. This addition would provide users with valuable information about their surroundings, further improving navigation and situational awareness.

3. **Expansion of the Dataset:** To improve the system's robustness and accuracy, expanding the custom dataset to include a broader range of object classes and environmental conditions is essential. This could involve adding new subclasses relevant to outdoor navigation, such as different types of terrain, various types of street furniture, and additional objects commonly encountered in urban and rural settings.
4. **Integration of Additional Sensors:** Adding more sensors, such as GPS for precise location tracking or environmental sensors for detecting weather conditions, could enhance the system's capabilities. GPS integration could provide users with location-based guidance, while environmental sensors could offer additional context about changing conditions, improving the system's adaptability and reliability

Bibliography

- [1] Freepik. Human eye anatomy. https://www.freepik.com/free-vector/human-eye-anatomy-poster-with-eyelid-optic-nerve-symbols-isometric-vector-illustration_26765445.htm. Accessed: 2024-03-18.
- [2] Magnifiers aids. https://m.media-amazon.com/images/I/71ss5rYBlL._AC_SL1500_.jpg. Accessed: 2024-03-25.
- [3] Amy Green. Assistive technology aids. <https://bhekisia.org/article/2016-03-22-game-changing-technology-for-blind-people-at-a-price/>. Accessed: 2024-03-25.
- [4] Aditi Kothiya and Dhruv Kumar Patwari. Different subdomain ai, ml, ann and dl. <https://medium.com/co-learning-lounge/what-is-deep-learning-ai-in-simple-words-ad2c39e13bf2>. Accessed: 2024-03-29.
- [5] Bishop Christopher M and Nasrabadi Nasser M. *Pattern recognition and machine learning*, volume 4. Springer, 2006. <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>.
- [6] Human learning (hl) versus machine learning (ml). https://www.researchgate.net/figure/Human-learning-HL-versus-machine-learning-ML-Model-capacity-in-ML-is-analogous-to_fig1_346857783. Accessed: 2024-05-04.
- [7] Masooma Memon. Ann vs cnn vs rnn: Neural networks guide. <https://levity.ai/blog/neural-networks-cnn-ann-rnn>. Accessed: 2024-04-10.
- [8] Qingsen Wu, Haixu Liu, Jian Xin, Lin Li, Zuochang Ye, and Yan Wang. Deep neural networks-based direct-current operation prediction and circuit migration design. *Electronics*, 12(13), 2023.
- [9] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.
- [10] Robin M Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *arXiv preprint arXiv:1912.05911*, 2019.
- [11] Jithin S L. Generative Adversarial Network (GAN). <https://www.linkedin.com/pulse/generative-adversarial-network-gan-jithin-s-1/>. Accessed: 2024-04-11.
- [12] Huaxiu Yao. Deep reinforcement learning architecture for tuning the vehicles' speeds in the simulator. <https://www.researchgate.net/figure>

[Deep-reinforcement-learning-architecture-for-tuning-the-vehicles-speeds-in-the-simulator_fig2_331344124](#). Accessed: 2024-04-11.

- [13] Harley Davidson Regua. Introducing Transfer Learning as Your Next Engine to Drive Future Innovations. <https://medium.datadriveninvestor.com/introducing-transfer-learning-as-your-next-engine-to-drive-future-innovations-5e81a15bb567>. Accessed: 2024-04-11.
- [14] O'Reilly Media. Types of object recognition tasks. <https://www.oreilly.com/library/view/neural-network-projects/9781789138900/e1f93bb9-0e51-428d-8e06-f19143ecc927.xhtml>. Accessed: 2024-04-13.
- [15] V7 Labs. Coco dataset: All you need to know to get started. <https://www.v7labs.com/blog/coco-dataset-guide>. Accessed: 2024-04-14.
- [16] szjy led. Frame Rate(FPS) vs. Refresh Rate(Hz): Do You Know Their Difference ? <https://www.szjy-led.com/frame-rate-vs-refresh-rate/>. Accessed: 2024-06-25.
- [17] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.
- [18] Aishwarya Singh. Selecting the right bounding box using non-max suppression (with implementation). <https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/>. Accessed: 2024-08-13.
- [19] Saeed Masoudnia. YOLOv2 + architecture. https://www.researchgate.net/figure/YOLOv2-architecture-YOLOv2-architecture-is-modified-with-our-new-assisted-excitation_fig3_333773329. Accessed: 2024-06-28.
- [20] ultralytics. YOLOv4 : Détection rapide et précise des objets. <https://docs.ultralytics.com/fr/models/yolov4/>. Accessed: 2024-06-28.
- [21] Ultralytics. Overview of model structure about YOLOv5. <https://github.com/ultralytics/yolov5/issues/280>. Accessed: 2024-07-05.
- [22] Chuiyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022.
- [23] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022.
- [24] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 and beyond, 2023.
- [25] 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, 2023.
- [26] Range King. Brief summary of yolov8 model structure. <https://github.com/ultralytics/ultralytics/issues/189>. Accessed: 2024-08-05.

- [27] Automatic urine sediment detection and classification based on yolov8 - scientific figure on researchgate. https://www.researchgate.net/figure/The-new-YoloV8-c2f-module_fig3_371992105, 2024. Accessed: 12-08-2024.
- [28] Ultralytics. Ultralytics yolov8 docs. <https://docs.ultralytics.com/tasks/detect/>. Accessed: 2024-08-13.
- [29] Ultralytics yolov8 docs. <https://docs.ultralytics.com/usage/cfg/>. Accessed: 2023-05-28.
- [30] Chuhan Wang and Yan Pang. Nano-based eye drop: Topical and noninvasive therapy for ocular diseases. *Advanced Drug Delivery Reviews*, 194:114721, 2023.
- [31] David Atchison. *Optics of the human eye*. CRC Press, 2023.
- [32] Helga Kolb. The architecture of functional neural circuits in the vertebrate retina. the proctor lecture. *Investigative ophthalmology & visual science*, 35(5):2385–2404, 1994.
- [33] Ian P Howard and Brian J Rogers. *Binocular vision and stereopsis*. Oxford University Press, USA, 1995.
- [34] Chang Che, Haotian Zheng, Zengyi Huang, Wei Jiang, and Bo Liu. Intelligent robotic control system based on computer vision technology. *arXiv preprint arXiv:2404.01116*, 2024.
- [35] Anne Lesley Corn and Jane N Erin. *Foundations of low vision: Clinical and functional perspectives*. American Foundation for the Blind, 2010.
- [36] Simona Caraiman, Anca Morar, Mateusz Owczarek, Adrian Burlacu, Dariusz Rzeszotarski, Nicolae Botezatu, Paul Herghelegiu, Florica Moldoveanu, Paweł Strumillo, and Alin Moldoveanu. Computer vision for the visually impaired: the sound of vision system. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 1480–1489, 2017.
- [37] Jinqiang Bai, Dijun Liu, Guobin Su, and Zhongliang Fu. A cloud and vision-based navigation system used for blind people. In *Proceedings of the 2017 international conference on artificial intelligence, automation and control technologies*, pages 1–6, 2017.
- [38] Manuel Martinez, Alina Roitberg, Daniel Koester, Rainer Stiefelhagen, and Boris Schauerte. Using technology developed for autonomous cars to help navigate blind people. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1424–1432, 2017.
- [39] Ashwani Kumar, SS Sai Satyanarayana Reddy, and Vivek Kulkarni. An object detection technique for blind people in real-time using deep neural network. In *2019 Fifth International Conference on Image Information Processing (ICIIP)*, pages 292–297. IEEE, 2019.
- [40] Jagadish K Mahendran, Daniel T Barry, Anita K Nivedha, and Suchendra M Bhandarkar. Computer vision-based assistance system for the visually impaired using mobile edge artificial intelligence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2418–2427, 2021.
- [41] Junjie Shen, Yiwen Chen, and Hideyuki Sawada. A wearable assistive device for blind pedestrians using real-time object detection and tactile presentation. *Sensors*, 22(12):4537, 2022.
- [42] Divyansh Chaudhary, Anubhav Mathur, Ayush Chauhan, and Aakanshi Gupta. Assistive object recognition and obstacle detection system for the visually impaired using yolo. In *2023 13th Interna-*

tional Conference on Cloud Computing, Data Science and Engineering (Confluence), pages 353–358, 2023.

- [43] Stefan Gherghina. An artificial intelligence approach towards investigating corporate bankruptcy. *Review of European Studies*, 7(7):5–22, 2015.
- [44] Andreas Kaplan and Michael Haenlein. Siri, siri, in my hand: Who’s the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, 2019.
- [45] Elena Popkova and Bruno S. Sergi. Human capital and ai in industry 4.0: Convergence and divergence in social entrepreneurship in russia. *Journal of Intellectual Capital*, 2020.
- [46] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.
- [47] Artificial intelligence (ai) vs. machine learning (ml). <https://cloud.google.com/learn/artificial-intelligence-vs-machine-learning#:~:text=Artificial%20intelligence%20is%20the%20overarching,systems%2C%20and%20natural%20language%20processing>. Accessed: 2024-05-04.
- [48] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.
- [49] Adem R.N. Aouichaoui, Resul Al, Jens Abildskov, and Gürkan Sin. Comparison of group-contribution and machine learning-based property prediction models with uncertainty quantification. In Metin Türkay and Rafiqul Gani, editors, *31st European Symposium on Computer Aided Process Engineering*, volume 50 of *Computer Aided Chemical Engineering*, pages 755–760. Elsevier, 2021.
- [50] Goodfellow Ia. Deep learning/ian goodfellow, yoshua bengio and aaron courville, 2016.
- [51] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans) challenges, solutions, and future directions. *ACM Computing Surveys (CSUR)*, 54(3):1–42, 2021.
- [52] Maxim Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.
- [53] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.
- [54] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.
- [55] Ozan İrsoy and Ethem Alpaydin. Unsupervised feature extraction with autoencoder trees. *Neurocomputing*, 258:63–73, 2017. Special Issue on Machine Learning.
- [56] Görkem Algan and Ilkay Ulusoy. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 215:106771, 2021.
- [57] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020.
- [58] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022.

- [59] Vidushi Meel. What is the COCO Dataset? What you need to know in 2024 . <https://viso.ai/computer-vision/coco-dataset/>. Accessed: 2024-04-14.
- [60] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.
- [61] ImageNet. About ImageNet. <https://image-net.org/about>. Accessed: 2024-04-14.
- [62] Open Images team. Open images dataset v7. <https://storage.googleapis.com/openimages/web/index.html>. Accessed: 2024-04-14.
- [63] Naif Alsharabi. Real-time object detection overview: Advancements, challenges, and applications. *Omran University Journal*, 3(6):12–12, 2023.
- [64] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.
- [65] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.
- [66] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [67] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.
- [68] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.
- [69] Seyed Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and A loss for bounding box regression. *CoRR*, abs/1902.09630, 2019.
- [70] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.
- [71] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [72] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
- [73] Ultralytics. YOLOv5. <https://docs.ultralytics.com/fr/models/yolov5/>. Accessed: 2024-07-05.
- [74] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, and Xiangxiang Chu. Yolov6 v3. 0: A full-scale reloading. *arXiv preprint arXiv:2301.05586*, 2023.
- [75] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

- [76] Vaibhav Singh. Meet yolo-nas: New yolo object detection model beats yolov6 and yolov8. <https://learnopencv.com/yolo-nas/>. Accessed: 2024-07-07.
- [77] Haiying Xia, Cong Yao, Yumei Tan, and Shuxiang Song. A dataset for the visually impaired walk on the road. *Displays*, 79:102486, 2023.
- [78] Atikur Rahman Chitholian. Pothole dataset, 2020. Available on Roboflow Universe under ODbL v1.0 license.
- [79] Haiying Xia, Cong Yao, Yumei Tan, and Shuxiang Song. A dataset for the visually impaired walk on the road. *Displays*, 79:102486, 2023.
- [80] ultralytics. Yolov8 architecture explained: Exploring the yolov8 architecture. <https://yolov8.org/yolov8-architecture-explained/>. Accessed: 2024-08-05.
- [81] David Chapet. Google drive : pourquoi et comment l'utiliser ? <https://atlanticdigital.fr/google-drive#:~:text=Google%20Drive%20est%20un%20cloud,importe%20o%C3%B9%20sur%20la%20plan%C3%A8te>. Accessed: 2024-08-23.
- [82] python team. The python tutorial. <https://docs.python.org/3/tutorial/index.html>. Accessed: 2024-08-23.
- [83] Pytorch. <https://pytorch.org/>. Accessed: 2024-08-12.
- [84] Pytorch. <https://github.com/pytorch/pytorch>. Accessed: 2024-08-12.
- [85] Oencv. <https://opencv.org/>. Accessed: 2024-08-12.
- [86] JetBrains. (n.d.). Pycharm: Python ide for professional developers. <https://www.jetbrains.com/pycharm/>. Accessed: 2024-08-12.