République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Chadli Bendjedid
Faculté des Sciences et de la Technologie
Département d'Informatique

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم الاعلام الالي

جامعة الشاذلي بن جديد
UNIVERSITE CHADLI BENDJEDID

THESIS

Submitted in partial fulfilment of the requirements for a Master's degree

A system for detecting traffic objects and estimating their distance

Field: Computer Science

Specialty: Intelligent Computer Systems

By

DOUAOUI Rayane

Defended on: 22/06 /2022

In front of the Jury composed of:

Quality	Name and Surname	Rank	University
President :	Mrs.CHEBBAT	MAB	Chadli Bendjedid El-Tarf
Supervisor :	**Mrs.BOUGUERN I**	**MCB**	**Chadli Bendjedid El-Tarf**
Examiner :	Mr.BENTRAD	MCA	Chadli Bendjedid El-Tarf

University Year: 2023/2024

Acknowledgment

I would like to thank the Almighty God for the patience and strength to overcome all difficulties.

I would also like to express my deep gratitude to my dear supervisor, Dr. BOUGUERNE Imen, for her encouragement and for being there at every moment during my research journey, for her seriousness and dedication.

I extend my sincere thanks to all the members of the jury for their presence and for having reviewed our work.

I reserve my final thanks for all my family and friends, particularly my parents for their sacrifices, their encouragement, their presence, and their daily support.

DOUAOUI Rayane

Dedication

To those who are dear to me…

Contents

List of Figures

List of Tables

List of acronyms

CNN	Convolutional Neural Network.
ResNet	Residual Network.
IoU	Intersection over Union.
MSE	Mean Squared Error.
RMSprop	Root Mean Square Propagation.
SGD	Stochastic Gradient Descent.
VOC	Visual Object Classes.
Adadelta	Adaptive Delta.
Adam	Adaptive Moment Estimation.
FLTR	Fast Low-Tolerance Recognition.
Adafactor	Adaptive Factor.
Nadam	Nesterov-accelerated Adaptive Moment Estimation.
OpenCV	Open Source Computer Vision Library.
MSCOCO	Microsoft Common Objects in Context
FPS	frames per second
R-CNN	Region Based Convolutional Neural Networks
SPP-net	Spatial Pyramid Pooling in Deep Convolutional Networks

General Introduction

Technological developments in the last two decades have significantly facilitated access to digital systems in our daily lives. Among the key elements of digital systems, great attention is being paid to images. Currently, the representation and processing of digital images is the subject of very active research. The processing of images is a very large area that has evolved considerably over the past few decades.

- **Project Context and Problem Statement**:

One of the most powerful and convincing kinds of artificial intelligence is computer vision. Computer vision is the field in computer science that focuses on replicating some parts of the complexity of the human vision system and allowing computers to identify and process objects in images in the same way that person do. Until recently, computer vision was functioning on a limited basis. Thanks to advances in artificial intelligence and innovations in deep learning, significant progress has been made in recent years, surpassing human abilities in some detection and identification tasks.

Moreover, one of the most important applications of computer vision is detecting objects on the road, which contributes to advanced driving by tailoring, automating, and enhancing cars to increase safety and improve the driving experience. This technology is designed to alert drivers to potential risks and assist them in maintaining control of their vehicles to prevent or minimize accidents, as most accidents are caused by human errors. For this reason, systems also calculate distances between the car and detected objects on the road, and between pairs of detected objects, aiming to provide users with a smart, safe, and comfortable driving experience.

- **Motivations**:

This thesis explores the application of computer vision and image processing techniques with an emphasis on identifying various road traffic entities, including cars, animals, bicycles, and more. It aims to locate these items, categorize them, and compute the distances between the objects and the camera, as well as between pairs of detected objects.

Furthermore, we propose building a Convolutional Neural Network (CNN) on top of ResNet-50, adding layers to accomplish the thesis's primary goals. The CNN used in the study is trained using the VOC 2012 dataset, and it has shown impressive results across a range of computer vision applications. To make our system practical for real-world use, we will develop

an approximation algorithm for computing distances.

- **Objectives**:

The objectives of this thesis are as follows:

1. Identify and categorize road traffic objects and calculate the distances between detected objects and the camera, as well as between pairs of detected objects.

3. Develop an efficient Convolutional Neural Network (CNN) for these tasks.

4. Propose an approximation algorithm for distance computation.

5. Evaluate the performance of the proposed model in terms of classification accuracy, classification loss, model accuracy, Mean Squared Error (MSE), and Intersection over Union.

- **Thesis Content**

The organization of this thesis is as follows:

• State of the Art

In this chapter, we provide an overview of the current state of computer vision. We delve into the fundamental concepts of image, digital image, image processing, and computer vision. The chapter explores methods of road traffic objects detection and methods for estimation distance. Additionally, we review related works and research studies that have contributed to the advancement of those methods.

• Conceptual Study

The second chapter focuses on the conceptual study of our detecting traffic objects and estimating their distance system. First, we discuss the system design. After that, the chapter covers the presentation of the dataset used for training and evaluation. Then, we present the object detection part, explaining the reason behind choosing the model and describing the architecture of our deep learning model. We conclude this section with the optimization methods used.

In the second part, we present the distance estimation algorithm. We discuss the algorithm proposition and explain why we calculate two types of distance, then provide a pseudo-code for the algorithm.

We ended the chapter by presenting the model evaluation methods and metrics.

• Results and Implementation Work

In last Chapter, we present the results and implementation work of our detecting traffic objects and estimating their distance system. We discuss the representation of the development tools and frameworks used in implementing the system. Experimental results are presented, highlighting the accuracy and performance achieved our model. Additionally, we present the prediction results and model evaluation metrics to assess the system's effectiveness. Furthermore, we showcase the user interface of our system and discuss the testing process to validate its usability and functionality.

Chapter 01:
State of art

1. Introduction

Today's civilization has incorporated images into many facets of daily life, making them an essential part of it. Since smartphones and digital cameras have made it so easy to take and share photos, we are continuously surrounded by visual content in the age of technology. This includes social media platforms and advertising campaigns. Images are a potent communication tool that let us share information and ideas visually appealingly while also expressing and transferring emotions.

As technology advances, so does our capacity to work with and decipher these images via the field of image processing. This final one is a foundational step in many fields, computer vision included.

As a result, we will go into detail about image processing and computer vision in this chapter. We will also present techniques and provide explanations for terms of object detection and distance estimation.

2. Image processing and computer vision

Before we jump into image processing and computer vision, we need first to understand what exactly constitutes the based thing to make image processing.

2.1. Digital image

2.1.1. Definition

A digital image is defined by its surface being divided into units of measurement known as pixels or cells. Every one of these pixels has a unique quality that indicates a color or grayscale level. An picture is digitized when its analog state is converted into a digital representation as a two-dimensional matrix of numerical values, or f(x, y). The conversion is depicted in the figure, where f(x, y) denotes the corresponding intensity level and the Cartesian coordinates x and y represent a particular location in the image. Thus, each point in the matrix represents the light intensity value that the sensor picked up. [3].

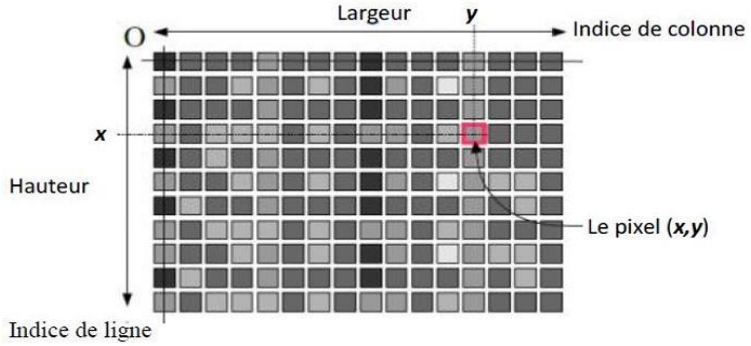

Figure 1: Representation of digital image [3].

2.1.2. Digital image characteristic

Each digital image characteristics by:

- **Pixel:** Pixels are arranged into a two-dimensional framework to produce a visual representation in a digital image**.** [20].

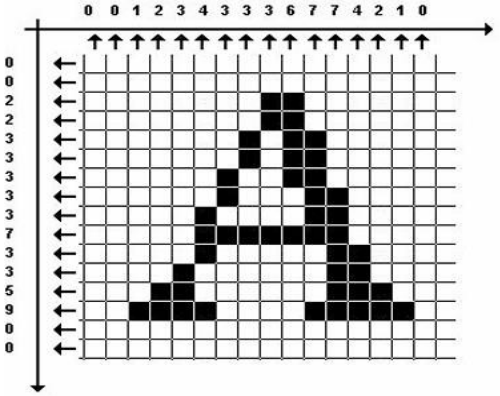

Figure 2: The letter A displayed as a group of pixels. [20]

- **Dimension:** dimensions of a picture represented as a matrix with pixel values assigned to each element [2].
- **Resolution:** is the number of pixels in the image, expressed in dpi or ppi, per length unit. It displays the degree of detail [2].
- **The size of an image:** Count the pixels in an image, which is the product of its height and width, to get its dimensions. Size is equal to pixels × each byte's size [2].
- **Noise:** Another term for abrupt changes in pixel intensity between neighboring pixels that result from illumination in an image is interference [2].

Figure 4: Image without noise. [2]

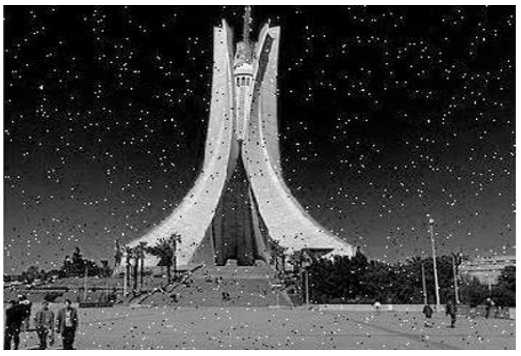

Figure 3: Image with noise.[2]

- **Histogram:** is a statistical graphical representation showing pixel intensity distribution in an image [20].

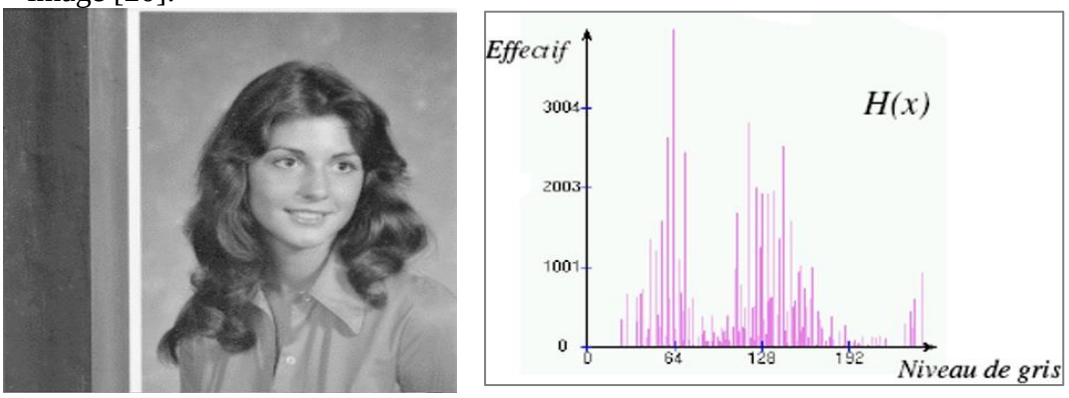

Figure 5: Representation of a histogram of an image.[20]

- **Contours and textures:** Defined contours show the edges of objects or the changes in grayscale between different pixels. Surface structure is characterized by textures [20].
- **Luminance:** This is the ratio of apparent area to luminous intensity, or the brightness level of each point in the image. Bright visuals, ideal contrast, and the lack of visual noise are characteristics of optimal brightness [20].
- **Contrast:** It's the obvious distinction between two different areas—that is, between two areas light and dark. By taking into account the luminance of both picture areas, contrast is calculated [5].

2.1.3. Digital image formats

A digital image divided to two formats:

- **The raster image:** A bitmap image is made up of a grid of pixels, each of which has a color and position. Image resolution is improved with a high pixel density. Common file types are JPG, PCX, GIF, and BMP [27].
- **The vector image:** Using mathematical formulas to describe data, vector graphics enable infinite scaling without appreciable quality loss and use less memory. Examples of formats are DXF and CGM. For resizing, mapping, and graphic design tasks, they are perfect [27].

2.1.4. Digital image types

Moreover, it splits to three types

- **Binary images**: It's a binary. The black and white mode provides simple yet sufficient image reproduction by assigning 0 (black) or 1 (white) [28].
- **Grayscale image**: Compared to plain black and white, grayscale coding offers more shades and a more accurate representation of intensity. 8-bit encoding typically produces 256 hues [28].
- **Color images:** Three or more grayscale images on different color channel red, green, and blue are combined to create a color image [27].

2.2. Image processing

2.2.1. Definition

Over the past few decades, there has been substantial advancement in the broad subject of image processing. It includes methods for improving or extracting information from digital photographs [27]; to be more precise, it converts the information contained in an image into digital data so that the computer can process it more efficiently. [1].

2.2.2. Types of Image Processing

Image processing can be divided into five basic categories [30]:

- **Visualization**: Locate objects in the picture that are not visible.
- **Recognition**: Identify or distinguish between objects in the picture.
- **Sharpening and restoration**: create an improved image from the source image

- **Pattern recognition**: Calculate the many patterns that surround the image's objects.
- **Retrieval**: Search a sizable digital collection for photos that have a striking resemblance to the original.

2.2.3. Image Processing Techniques

The process of processing images can be divided into several different steps, some of which are as follows:

- **Image Acquisition**: In order to manipulate an image on a computer, it needs to be transformed so that the system can read and use it. Digitization techniques like quantification and sampling are used to accomplish this change [4].

- **Filtering**: Using mathematical morphological techniques or interpolation, this approach seeks to minimize image noise. Linear and nonlinear filters are classified [5].
  - ✓ When a new pixel value is obtained via a linear combination of nearby pixel values, the filter is referred to as linear. It falls into a number of categories, including band-pass filter (differentiation) and high-pass filter (sharpening) [5].
  - ✓ Non-linear filters: To overcome the shortcomings of linear filters, such as persistent outliers and inadequate transition preservation, non-linear filtering entails replacing each pixel value with a non-linear combination of its nearby pixel values. Low-pass and high-pass filters are examples of non-linear filters. Methods of non-linear filtering: Canny filter, median filter [5].

- **Segmentation**: An image can be divided into its several sections of interest using segmentation. The method involves segmenting the image into smaller areas referred to as regions. A region is a group of related pixels that are distinguished from nearby regions by shared characteristics (such as texture, intensity, etc.) [29].

Two groups comprise a variety of segmentation techniques.

- ✓ Segmentation according to region: A region-based method prioritizes contextual technique—that is, the affinity between related places. Connected points are grouped together based on shared characteristics such as texture, color, or intensity of gray [5].

- ✓ Contour-based segmentation: This method ignores the relationships between various image parts because it is non-contextual. Pixels are clustered according to a global characteristic that includes edge detection methods. The intended partition may not always be reached by the resulting edges. Because edges are usually formed by disconnected pixels, contour closure methods are needed. Regions do not appear until after closure and are defined by inner contours [5].

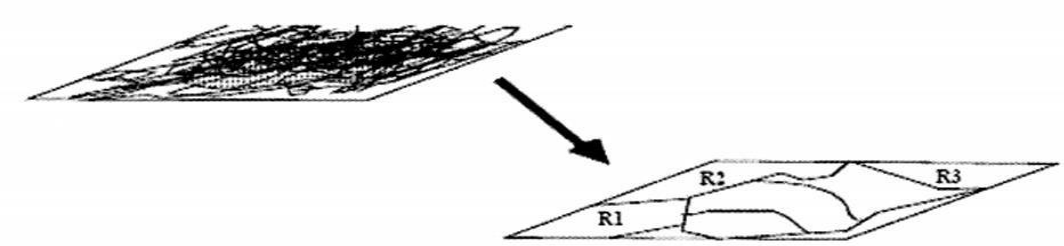

Figure 6: image segmentation. [27]

2.2.4. Applications of Image Processing

Medical image retrieval, image segmentation, character recognition, and text/draw separation are just a few of the many applications for image processing. To guarantee that the image is sufficiently clear and effectively identifies information, image processing needs to be supported by various technologies to improve the resolution and quality of the image. The development of artificial intelligence technology also encourages the development of image processing technology, making it widely used in areas such as pattern recognition, multimedia technology, and computer vision [1]. Since we are interested in one of the fields of computer vision in this search, we will concentrate on computer vision in the following section of the chapter**.**

2.3. Computer vision

2.3.1. History

The Summer Vision Project was started in 1966 by AI pioneers Seymour Papert and Marvin Minsky with the goal of building a computer that could recognize things in pictures. They faced difficulties because of the predominate symbolic AI approach in their efforts to construct software that could recognize objects based just on pixels. Due to differences in object appearance and background interference, manually specifying object detection rules proved to be impractical.

A computer vision system called the "neocognitron," developed by Japanese scientist Kunihiko Fukushima in 1979, was influenced by research on the human visual cortex. The neocognitron, in spite of its drawbacks, set the stage for important developments in the field of computer vision [33].

2.3.2. Definition

One of the AI research areas that is expanding the fastest is computer vision. It started with neural network technology in the 1980s [8], and its main objective is to use computers to mimic human learning and vision, as well as to be able to make decisions and take actions based on visual data. This technology relies on algorithms for computer perception, allowing machines to recognize objects they capture [6].Computer vision divided into three main categories [32]:

- **Classification**: it determines the classifications of objects in pictures by allocating binary or categorical labels together with corresponding probabilities. Its objective is to define a classification challenge for discrete datasets by classifying outputs into particular groups [32].
- **Detection**: it finds things in pictures or videos by first classifying them, then encircling them with bounding boxes. A computer vision engineer sets the threshold for each box's confidence score [32].

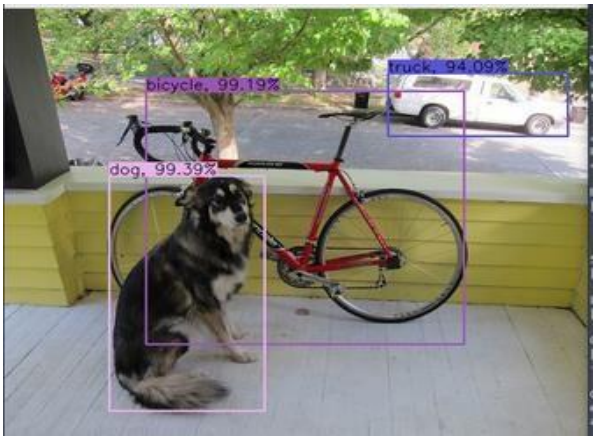

Figure 1: Object Classification and Detection example. [33]

- **Segmentation**: • There are two primary categories of segmentation [32]:
  - ✓ Semantic segmentation groups all instances of a class together by assigning a label to each pixel in an image.
  - ✓ Identifying and locating items in a picture by treating every instance of a class as a distinct object is known as instance segmentation.

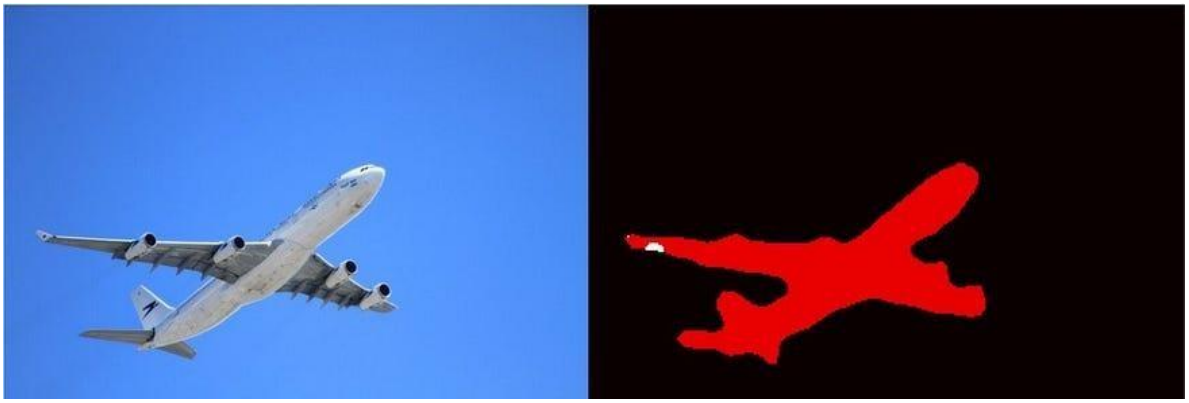

Figure 7: Image Segmentation example [33]

2.3.3. Why we need computer vision

The field of computer vision is devoted to building devices with visual perception. You may question why it would be worthwhile to spend in creating machines that can imitate the human visual system, considering its amazing capabilities. Well, there are a number of strong arguments. First of all, a lot of the mundane jobs we complete every day could be assigned to machines, freeing up our time for more rewarding pursuits. The tasks around the house and the commute to work are two examples. Second, although our vision is strong, it is usually subjective as opposed to objective. It finds it difficult to make accurate measurements in the real world. Finally, and perhaps most significantly, a computer vision system is able to derive insights from the visual world that are not possible for humans to do [31].

2.3.4. Computer vision and deep learning

Although computer vision is not new, its importance has lately increased as a result of developments in deep learning, artificial intelligence, and neural network technologies [33]. Convolutional neural networks (CNNs) are the backbone of modern computer vision, offering significant performance gains over earlier techniques. CNNs use layers to condense data and compare it with pre-existing data to classify it [34].

- ## Convolutional Neural Networks

A typical convolutional neural network (CNN) architecture typically comprises of alternating layers of convolution and pooling, which are then followed by one or more fully connected layers at the end. Occasionally, a global average pooling layer is used as a substitute for a fully linked layer. Furthermore, the CNN's efficiency is optimized by incorporating multiple regularization units, such as normalization and batch dropout, in addition to diverse mapping algorithms. The configuration of CNN components is crucial in the development of novel architectures and the attainment of enhanced performance. This section provides a concise overview of the function of these elements within a Convolutional Neural Network (CNN) structure. [38]

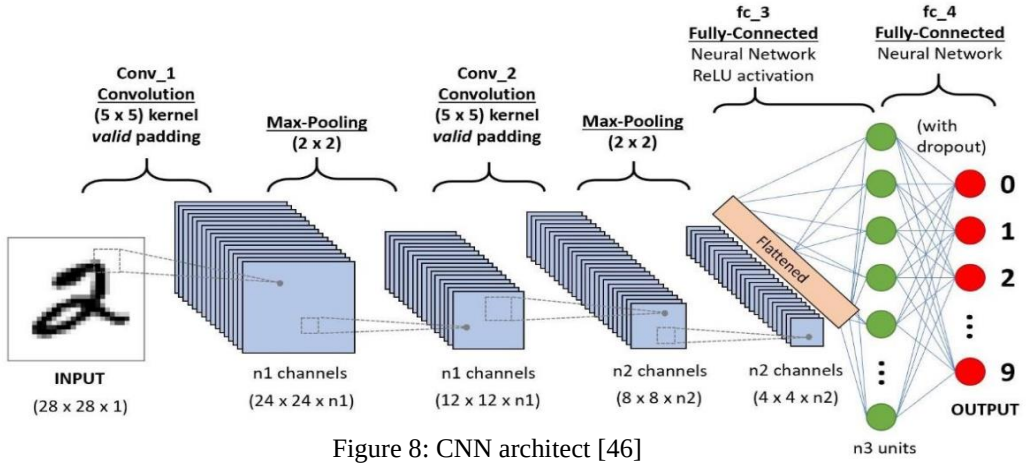

Figure 8: CNN architect [46]

The Convolution Layer: Features are extracted from the image via the convolution layer, also known as the feature extraction layer.

In order to perform a convolution operation and determine the dot product between the filter and the receptive field - a local area of the input image with the same size as the filter - a section of the image is first linked to the convolution layer. One integer representing the output volume is the outcome. The process is then repeated by swiping the filter over the subsequent receptive field of the original image. Until the entire image has been scanned, these steps are repeated. [41]

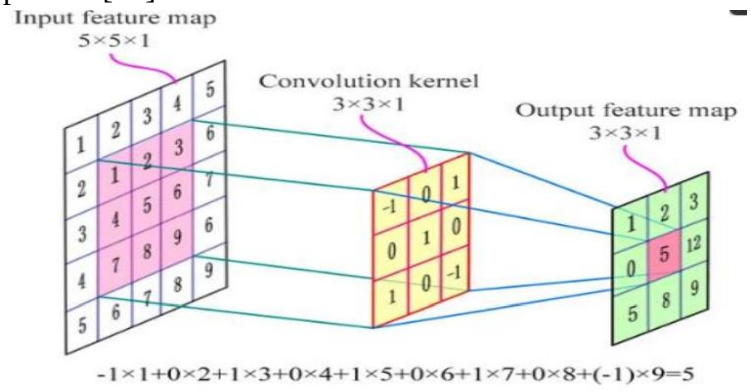

Figure 9: The Convolution Layer [47]

The Pooling Layer: A subsampling technique called the pooling layer (POOL) is usually used in between two convolutional layers. Its job is to gradually lower the feature map's (convolution matrix's) size in order to save calculations and network parameters while maintaining the necessary information. [39]

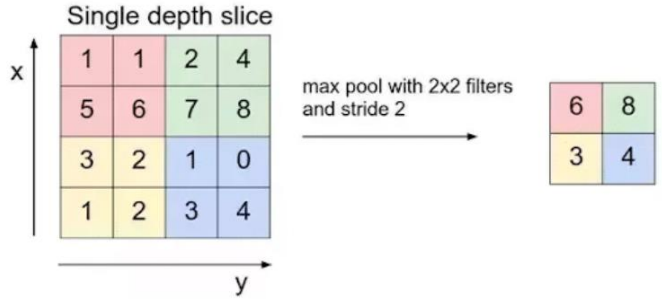

Figure 10: The Pooling Layer [41]

The Fully Connected Layer: In addition to carrying out the same functions as ordinary ANNs, the fully connected layers attempt to generate class notes from the activations for use in categorization. To increase performance, it is also advised to employ ReLu in between these layers. [40]

The Output Layer: The output layer comprises neurons responsible for classifying the model's classes, and the amount of neurons in this layer is contingent upon the number of classes. [47]

2.3.5. Application of computer vision

All computational activities involving visual content, including photographs, movies, and even icons, fall under the umbrella of computer vision. But this broad field has many subfields, including object categorization, handwriting recognition, image segmentation, and image restoration. Numerous domains are covered by these branches [33]:

- ✓ Health: Image processing is necessary for diagnostics, such as interpreting MRI and X-ray results.
- ✓ Computer vision offers significant support in sports. Major League Baseball, for example, employs AI to track the ball precisely.
- ✓ Virtual and augmented reality. Facial recognition: Used in consumer electronics like cellphones for user verification.

This study will concentrate on an area within a particular field: autonomous cars. autonomous vehicles to recognize around objects or other vehicles. A portion of it, object detection in traffic and distance estimate, will be our focus.

3. Objects detection

3.1. Definition

The combination of image processing and computer vision methods is known as object detection. It can identify objects belonging to the same class in pictures and videos [7]; it can identify an object's exact location in a natural image by using a variety of pre-established classifications. This is among the complex and fundamental issues in computer vision research. Object detection uses image recognition algorithms to classify and locate items in images, then predicts frames surrounding those objects for localization. [8]

3.2. Road traffic objects detection

An essential component of advanced driver assistance systems (ADAS) and intelligent transportation systems (ITS) is traffic object detection. It recognizes cars, bikes, pedestrians, and signs in real-time using computer vision, machine learning, and sensor fusion with the goal of enhancing road safety and maximizing transit [35].

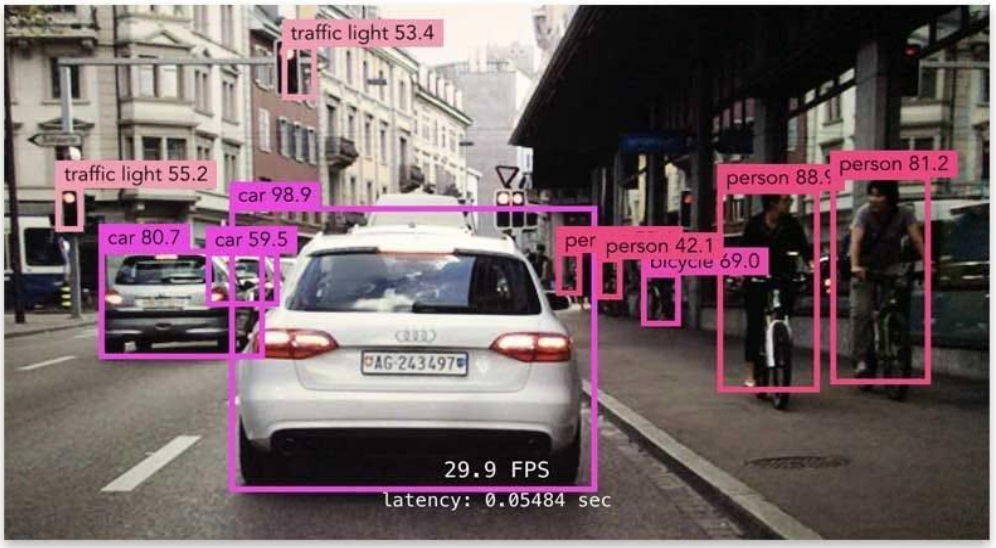

Figure 11: Exemple of road traffic objects detection. [36]

3.3. Importance of road traffic Objects Detection

In order to provide adaptive cruise control, lane departure warning, blind-spot detection, and pedestrian collision warning systems, traffic object detection is essential to ITS and ADAS. In addition to helping with traffic management, flow estimation, and signal timing optimization, it improves road safety by delivering prompt alerts and actions [35].

3.4. Methods

The "traditional object detection period (before 2014)" and the "deep learning-based detection period (after 2014)" are the two historical periods in which object detection has mostly developed. As noteworthy methods have developed after the introduction of deep learning to the field, we will examine contemporary approaches, which may be broadly divided into two categories: one-stage detection algorithms and two-stage detection algorithms.

Ref. No	Year	Methods		
			Method used	Results, Avantage, Limits, Robustness, …etc.
Algorithms based on ***C***onvolutional ***N***eural ***N***etworks				
[9]	2014	***Two-stage detection algorithm***	R-CNN	▪ mAP: 31.4% on ILSVRC 2013 and 62.9% on PASCAL VOC 2010. ▪ Notable advancement over conventional techniques. ▪ Effective in employing selective search to produce about 2000 ROIs. ▪ Long computation times because of several distinct phases. ▪ Necessitates producing a lot of ROIs. ▪ Repeated computations result from overlapping region suggestions. ▪ Less effective but more accurate than sliding window techniques.
[9] [11]	2014		SPPNET	▪ In the 1-scale version, 102× faster than R-CNN with only 1.2% lower mAP. ▪ 38 times faster in the 5-scale version with similar mAP. ▪ Produces images of a fixed size, increasing scaling invariance. ▪ Uses a single convolution operation to minimize overfitting and computational expense. ▪ Needs a large amount of storage space. ▪ Personal training is intricate. ▪ A slower, more deliberate search technique that uses the CPU to retrieve features. ▪ Increased speed and decreased overfitting, but at the expense of some training complexity. ▪ Addresses problems with R-CNN's fixed input size image blocks and low detection efficiency.
[9] [10]	2015		FAST R-CNN	▪ It has a better detection quality than R-CNN and SPPnet. ▪ It uses a multi-task loss during the single stage of training.

				▪ The network's layers can all be updated through training. ▪ Selective search is sluggish, resulting in a high processing time; it does not require disk storage.
[9] [12]	2015		FASTER R-CNN	▪ Faster and more accurate than Fast R-CNN. ▪ For effective region proposals, use Region Proposal Networks (RPN). ▪ The creation of region proposals more quickly and accurately. ▪ May generate bounding boxes that are erroneous. ▪ Increased accuracy and speed, but there are still some issues with bounding box precision. ▪ Quicker than R-CNN Implementation is slower than YOLO. ▪ Proposal for a slow object
[9]	2017		FPN	▪ It functions as a feature extractor and can alter the extractors of other detectors. ▪ FPN presents a pyramid approach to improve the quality of features for object detection. ▪ The architecture used by FPN is layered, with increasing semantic values at each step. ▪ The efficacy of FPN is mostly reliant on the semantic layers; high-resolution layers are generated by leveraging layers with rich semantics. ▪ Effective connection management is one of FPN's challenges. ▪ At every level, rich semantic information is available. ▪ Cutting off top-down links results in a reduction in precision.
[9] [13]	2015		MASKR-CNN ***-Instance Segmentation-***	▪ The mask R-CNN yields three outputs: a class label and a bounding-box offset for each candidate item, and a branch that predicts a segmentation mask within each region of interest (RoI). The other two outputs are similar to the Faster R-CNN outputs.
[15]	2016		SSD	▪ SSD improves detection accuracy by introducing multireference and multiresolution detection algorithms, especially for small objects. ▪ SSD outperforms YOLO, FPN, Fast R-CNN, Faster R-CNN, SPPnet, and RCNN in terms of detection speed and accuracy, achieving a COCO mAP@.5 of 46.5% and running at 59 frames per second (fps) in a fast version. ▪ SSD detects objects of varying scales across different layers of the network, leading to superior performance.

				▪ SSD improves the accuracy of one-stage detectors significantly, surpassing previous methods. ▪ SSD excels in both detection speed and accuracy, surpassing previous methods in terms of accuracy and speed.
[17] [18]	2017	***One-stage Detection algorithm***	RETINNET	▪ Two task-specific subnetworks and a backbone network make up the unified network known as RetinaNet. ▪ Over the whole input, the backbone network computes a convolutional feature map. ▪ image and is a convolutional network that already exists. ▪ Based on the backbone's output, one of the subnetworks does convolutional object categorization. ▪ The second subnetwork executes the convolutional bounding box regression. ▪ The two subnetworks share a common architecture that is optimized for one-stage, dense detection applications. Although these components can be configured in a variety of ways, most design parameters are not extremely sensitive to specific values.
[9] [14] [15]	2016		YOLO	▪ In order to create a grid-based structure for object detection, Yolo divides the input image into multiple grid cells. The classification and localization algorithm is applied independently to each grid cell. ▪ Based on the object's center position within the cell, each grid cell assigns class labels or tags. ▪ Yolo predicts a fixed number of bounding boxes and associated scores or class probabilities for multiple objects within each grid cell. ▪ Yolo is extremely efficient, providing fast processing speeds (ranging from 5 to 160 fps) while maintaining a high level of accuracy in object detection tasks. ▪ Yolo minimizes background errors and concentrates on accurately detecting objects of interest. ▪ YOLO might have trouble identifying several items in a single grid cell, which could result in missed detections.Small things are difficult for it to pinpoint, and it frequently detects the same object more than once. ▪ There are eight versions of YOLO, YOLOv7 is a later version of YOLOv4, which was first introduced. ▪ This is achieved by putting optimized frame works into practice, which include dynamic label assignment and re-parameterizing the model structure.

[16]	/		YOLOP	▪ YOLOP is an extremely effective multi-task network created to handle the three crucial tasks of autonomous driving: lane identification, segmentation of drivable zones, and object detection. It is noteworthy because it is the first system to accomplish real-time performance on embedded devices. ▪ When YOLOv5s, MultiNet, DLT-Net, and Faster R-CNN findings are compared, YOLOP produces superior outcomes.
[18]	2018		CORNERNET	▪ The innovative method achieves competitive results by using key point detection (upper left and lower right corners) for object detection. ▪ Absence of anchors lowers computational overhead and simplifies detection. ▪ Flexibility: Enables customized feature extraction networks since there are no pre-trained models and training begins from scratch. ▪ Corner Pooling: Presents a novel pooling technique that raises the precision of corner key point identification. ▪ Training Complexity: Since training is done from scratch, it is more time- and computationally-intensive. ▪ Edge Cases: Possible issues with partially concealed or poorly defined corners of objects. ▪ Generalization: Good generalization across datasets; nonetheless, the complexity of the scene and the objects affects performance. ▪ Scalability: For consistent performance, more testing on bigger, more varied datasets is necessary. ▪ Future study will be influenced by the switch from anchor-based to key point detection in key point detection. ▪ Corner Pooling: Increases precision
[19]	/		CENTERNET	▪ Investigates visual patterns by using a triplet of keypoints inside each bounding box. ▪ Employs a triplet of keypoints for object detection, so partially inheriting the capability of ROI pooling, but with a one-stage detection approach. ▪ Minimizes computational cost by concentrating mostly on center information. ▪ Utilizing center pooling and cascade corner pooling techniques, the keypoint identification algorithm integrates visual patterns seen inside objects.

Table 1: Road traffic objects detection methods.

3.5. Faster R-CNN

Faster R-CNN is a well-known object detection method that combines efficiency and accuracy in a balanced manner. Applications requiring a high degree of precision, such security systems, driverless vehicles, and medical imaging, are especially well suited for this approach. In order to propose regions of interest (RoIs) more effectively and save computational expenses, Faster R-CNN uses a region proposal network (RPN) that shares full-image convolutional characteristics with the detection network. End-to-end training is made easier, model tuning is made simpler, and overall performance is improved by the integration of the detection network and the RPN into a single framework.

The Faster R-CNN has many benefits. Because of its exceptional ability to achieve high detection accuracy, it is a good fit for complicated situations where accurate object localization is essential. Because of its adaptability, it can be used for a variety of object identification tasks, including as identifying common objects in photos and spotting small, closely spaced objects. Reliable performance in real-world applications is ensured by the model's robustness under a variety of situations, such as varying lighting, angles, and occlusions. By lowering the quantity of candidate areas that must be analyzed, the RPN's effective region proposal generation further expedites the detection process.

Faster R-CNN does have certain drawbacks, though. Although it is more effective than its predecessors, it is not as good for real-time applications where speed is crucial because it is typically slower than single-shot detectors like YOLO and SSD. It also needs a lot of memory and processing power, which can limit its deployment on devices with minimal resources. Compared to simpler models like YOLO, the architecture of Faster R-CNN is quite complex, requiring careful hyper-parameter adjustment and a more advanced implementation.

The Faster R-CNN architecture is still being improved upon, and variations like Mask R-CNN which has instance segmentation capabilities are being created to overcome some of its shortcomings and improve speed. These enhancements uphold the applicability and efficacy of Faster R-CNN, solidifying its position as the go-to option for object identification jobs where robustness and accuracy are more important than processing speed in real time. All things considered, Faster R-CNN is a great choice for a variety of object identification applications due to its robustness, adaptability, and precision.[21][9][12]
We will use the Faster R-CNN model for comparison with our model.

4. Distance estimation

4.1. Definition

Detecting objects is the first step towards measuring distance. Assisting in calculating the distance between a device and an object is vital for safety systems. The effective interaction with the object is therefore guided by this information [23].

Figure 12: Exemple of distance estimation.[37]

There are two approaches of estimating distance: vision-based and sensor-based [26] techniques.

- **Vision-based**: includes monocular vision (using a camera with a single lens) and stereo vision (using a camera with two lenses).
- **Sensor-based Techniques**: ultrasonic, radar, and lidar.

Using Lidar, stereo vision, and other deep learning techniques, we can train models.

4.2. Methods

Ref. No	Year	Methods		
			Method used	Results, Avantage, Limits, Robustness, …etc.
Methods based on ***Deep Learning***				
[22]	/		DisNet	▪ The YOLO-based object classifier employed in this design. ▪ A trustworthy approximation of the separations between items in static railway scenes and a single RGB camera scenes captured with cameras. ▪ DisNet measures distances to different objects in dynamic road scenes and static railway sceneries with high accuracy. ▪ Comparing DisNet with HiSpe3D: DisNet identified more objects and offered similar or superior distance estimations in dynamic road scenarios to HiSpe3D. ▪ Bounding Box Extraction: Relies on the YOLO classifier at the moment; may require enhancements. ▪ Classifier Dependency: For improved performance, a 2D image processing-based classifier is intended to take the place of YOLO.
[25]	/		Faster R-CNN based on the ZF model	▪ The technique shows less than 2% inaccuracy in short-range detection. It meets the accuracy standards for vehicle distance sensing and performs well, particularly at a real distance of 10 meters. Error rises with real distance by a small amount, up to 1.2 meters at 40 meters. ▪ Additionally, the application's versatility can be restricted by the usage of a fixed camera and the choice of particular photos for analysis. ▪ In terms of vehicle distance detection, the suggested strategy has shown to work satisfactorily, especially over short distances.
[23] [24]	/		YOLO	▪ When compared to the first and second versions, Yolo v3 has a stronger recall. ▪ Yolo v3 is the best option for real-time applications since it provides excellent operational precision along with speed. ▪ When compared to Faster R-CNN based on the ZF model, Yolo has lowered the error rate to 2%.

ADAS system - ***A***dvanced ***D***river ***A***ssistance ***S***ystems -				
[26] [27]	/	***Based on the sensor***	Lidar	▪ Uses laser pulses to measure distance by timing the return of reflected light. ▪ Detects objects at different speeds, ranging from slow to fast; Operates independently of illumination conditions. ▪ Expensive to install and maintain; sensitive to weather and atmospheric conditions, which might impair accuracy.
	/		Radar	▪ Unlike other systems, RADAR is unaffected by rain, fog, or other unfavorable weather conditions (unlike ultrasonic). ▪ It can identify objects at great distances. ▪ It gives dependable distance estimation. ▪ It can detect objects at long distances. ▪ The resolution of radar may be restricted, which may impair its capacity to discriminate between objects that are extremely near to one another. ▪ Complexity in data processing because precise distance calculations and signal processing are required.
	/		ultrasonic	▪ High-frequency sound waves are used to measure the distance between a transmitter and a receiver. ▪ The device is widely used in automatic parking systems and has good penetration despite inclement weather, including snow, fog, and rain. ▪ Limited precision in cluttered settings; due to the Doppler effect, its application in high-speed vehicles is limited by its relatively sluggish propagation speed.
[26]	/	Method proposed by Tzu-Yun Tseng, Jian-Jiun Ding		▪ Principle: By utilizing the geometric relationship between the camera and a few key characteristic points, one can estimate the distance. ▪ Outcome: dependability, resilience, and the capacity to function with a single camera. With an average inaccuracy of 0.1776, this approach performs better across both short and long distances than previous methods.

Table 2: distance estimation methods.

5. Conclusion

We presented image processing, computer vision and their relationship in this chapter. Following that, we discussed traffic object and distance estimation, as well as their respective methodologies, which led to the conclusion that both of them are becoming more and more essential study topics. The creation of efficient systems that offer prompt alerts and interventions has the potential to improve road safety.

Many deep learning method hybridizations have been developed to address object detection and distance estimation, where deep learning has emerged as a promising approach. Numerous research have shown that object detection and distance estimation systems based on deep learning perform better than those based on classic machine learning methods.

Overall, the study discussed in this chapter emphasizes the significance of object identification and distance estimate in traffic, as well as the promise of deep learning to help overcome the difficulties in this area. More investigation is required to create object detection and distance estimate systems that are more precise and effective.

Chapter 02: Conceptual Study

1. Introduction

Detecting objects in an image involves two steps: finding the objects and categorizing them. The act of localizing an object involves drawing a border box around it; the process of classifying the localized object involves giving it a category. Convolutional Neural Networks (CNNs) are widely utilized techniques for objects detection.

Objects detection is a fundamental task in computer vision, with applications spanning from autonomous driving to image comprehension. The suggested objects detection model is the main topic of this chapter, with a focus on utilizing ResNet50 as backbone and adding convolutional layers to meet the objectives of our thesis. We analyze the architecture of this model and provide an explanation of its reasoning, using the Pascal VOC 2012 dataset to monitor its performance. We also offer Faster R-CNN as another crucial tactic. Finally, we investigate distance estimate subsequent to detection, which is essential for comprehending the spatial relationships between objects and the camera, as well as between detected object pairs.

2. System design

Our system detects traffic objects (car, truck, pedestrian, bicycle, traffic light, motorcycle, bus, stop sign, etc.) and measure the distance between camera and object and between pair of detected objects in the images. Our system begging with upload the dataset then it prepare the data to work with. The dataset is splited to train set and test set. After that we trained the model with those data, the model will be predict this part of detection is ready.

The second part is estimate distance; we will take the results to make them as an input to our distance estimation algorithm. With the interface we will see carefully the results of the system.

We represent below the general architecture of our system.

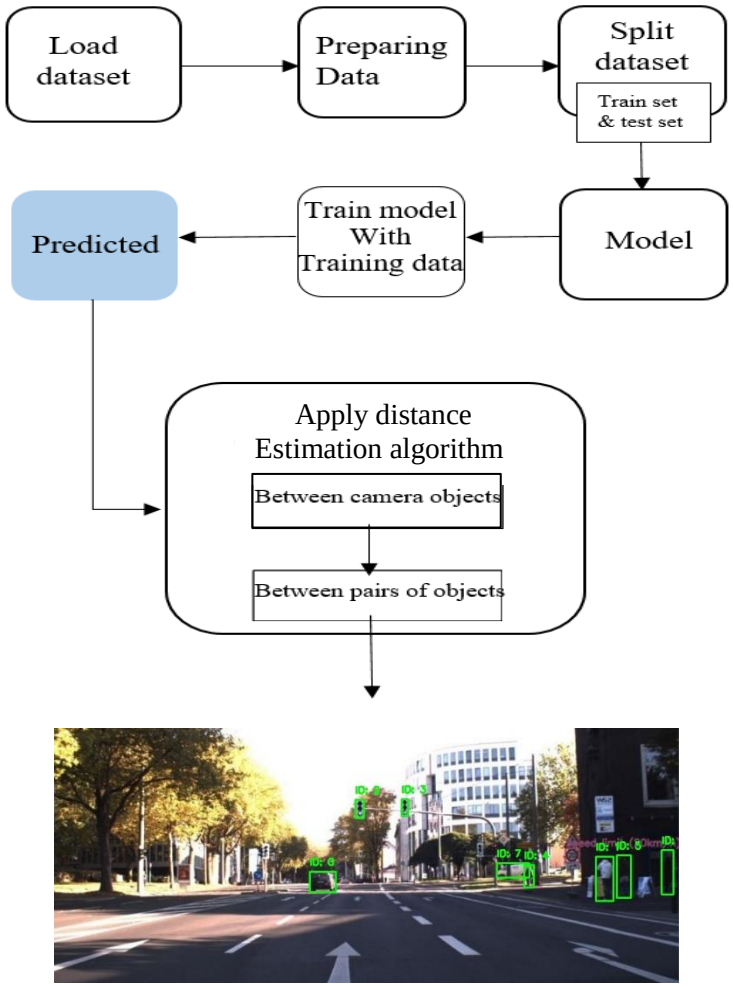

Figure 13: Our system architecture

3. Dataset

The dataset consists of 20 classes. 11,530 photos with 27,450 annotations for regions of interest make up the dataset, which can be used for training and validation. The dataset has twenty classes, which are as follows:

- Person: person.
- Animal: Sheep, dogs, horses, cats, birds, and cows.
- Vehicles: motorbike, train, bus, bicycle, boat, airplane, and car.
- Indoors: TV/monitor, sofa, dining table, bottle, and potted plant.

Trainval (2913 images), train (1464 images), test (1456 images), and val (1449 images) are the three subsets that make up the dataset. For development tasks like feature selection and parameter adjustment, the trainval set is employed. Results on unseen data are reported using the test set. It is crucial to remember that algorithms must only be executed once on the test data in order to guard against misuse of the evaluation server.

The most common problems are segmentation (which class does each pixel belong to), detection (where are the examples of a certain object class in the image?), and classification (does the image contain any instances of a given object class?). Two more issues exist in addition to these: action classification (i.e., what action is the person indicated in this image performing?) and person arrangement (where are the people's hands, feet, and heads in this picture?).

Because of its size and excellent annotations, the PASCAL VOC 2012 dataset has been extensively utilized as a benchmark for computer vision applications. It's also important to note that this dataset contains the neutral class. This unique class permits less accurate manual tagging of object boundaries during neural network training. However, with the advent of interactive systems that offer high-quality pixel-level segmentation and have substantially sped up human labeling, this strategy is becoming less common.[49]

4. Objects detection

4.1. The proposed model

4.1.1. The idea behind choosing the model

a. What makes ResNet50 special?

We utilize ResNet50 for several purposes, particularly in the context of object identification and classification tasks:

Rich architecture: ResNet50, a 50-layer deep convolutional neural network, is very helpful for difficult tasks like object detection and classification since it can extract complicated features from the data.[52]

Residual Learning: The introduction of residual connections, also known as skip connections, is one of the main advances of ResNet50. By reducing the effects of the vanishing gradient issue, these connections enable the network to train more efficiently even at deeper depths.[52]

Performance: ResNet50 is a dependable option for many computer vision jobs because of its outstanding performance on multiple benchmarks. Its well-balanced architecture offers a decent compromise between computing efficiency and precision.[52]

Versatility: ResNet50 is adaptable and may be used for a wide range of tasks, including segmentation and object detection, in addition to image classification. By building bespoke layers on top of the underlying model, ResNet50's feature extraction capabilities can be used for a variety of downstream applications.[52]

In our model, ResNet50 is used as the basic model for feature extraction:

High-level features are extracted from the input photos using the pre-trained ResNet50 model. This facilitates the use of the potent representations that ResNet50 has acquired.

In order to work with ResNet50 we need to modify in the architecture we have two parts the first one about the main ResNet50 and the second one about adding convolutional layers, in this part will explain the first one, the next title for the second one.

After importing ResNet50, we excluded the fully connected layers at the top, which are typically used for classification, which allow us to add our own layers for custom tasks (object detection), from the convolutional layers we need the output feature maps, which represents in 7*7*2048 dimensional array.

These feature maps 7*7*2048 serve as the foundation for predicting bounding box coordinates and object classes. The modifications shown below:

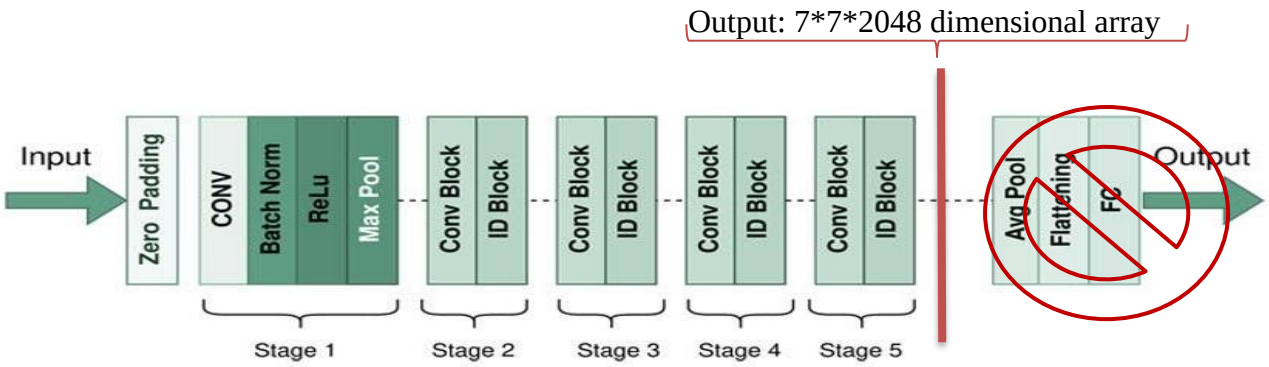

Figure 14: modified ResNet50

After that we Freeze Layers of the pre-trained ResNet50 model. This prevents their weights from being updated during training, the raisons why to freeze those layers are:

- Avoid Overfitting: Assists in preserving stable representations of previously trained layers, thereby lowering the possibility of overfitting in the case of insufficient training data.
- Computational efficiency: Lowers the amount of memory needed and training times by reducing the number of parameters that need to be updated.
- Early Stages of Transfer Learning: When fine-tuning or transfer learning is just starting, layers are usually frozen. As a result, you may use the pre-trained layers for feature extraction during the early training epochs and concentrate on changing only the weights of the newly added layers.

b. Why Include More Convolutional Layers?

- Customization for Particular activities: The model is more suited for specialized activities like object detection thanks to custom layers, which enable better feature extraction catered to the particulars of the issue.
- Performance Improvement: By strengthening the hierarchical structure of features, further layers help to capture more intricate patterns and finer details, which raises detection rates and accuracy.
- Flexibility and Control: Custom layers enable the development of output heads with specified functions, such as predicting classification probabilities and bounding box coordinates. Better Transfer Learning: Make use of an already-trained model and let it adjust to new, task-specific data; this will save training time and data requirements while improving performance.

5.1.1. Description of the architecture

We have developed a model architecture based on the ResNet50 framework, specifically intended for intricate image processing applications like object detection. Images with a resolution of 224x224 and three color channels (RGB) are initially accepted by the input layer. The crucial layer, conv5, is obtained after four convolutional layers. It employs 512 and 2048 filters to reduce spatial dimensions to 7x7 and to increase depth to 2048. The 7x7x2048 dimensional array that represents the final feature map from the ResNet50 backbone provides the input for later layers that are designed for certain tasks. The architecture has two Conv2D layers after ResNet50 . The depth is first lowered from 2048 to 512 and then again to 256 in the second. Two independent output heads intended for bounding box prediction and classification tasks subsequently process these improved feature maps. In order to represent bounding box coordinates, the output is reshaped to (49, 4) by the bounding box output head using a Conv2D layer with four filters. Class probabilities are represented by reshaping the output to (49, 10) using a Conv2D layer with 10 filters in the classification output head. This architecture is ideal for object detection since it makes use of the strong feature extraction capabilities of ResNet50 and the adaptability of extra Conv2D layers. With the effective reduction of spatial dimensions and enhancement of feature depth, it guarantees precise and comprehensive predictions for bounding boxes and class labels. Remaining blocks and task-specific layers together provide a strong design that can handle challenging image analysis tasks.

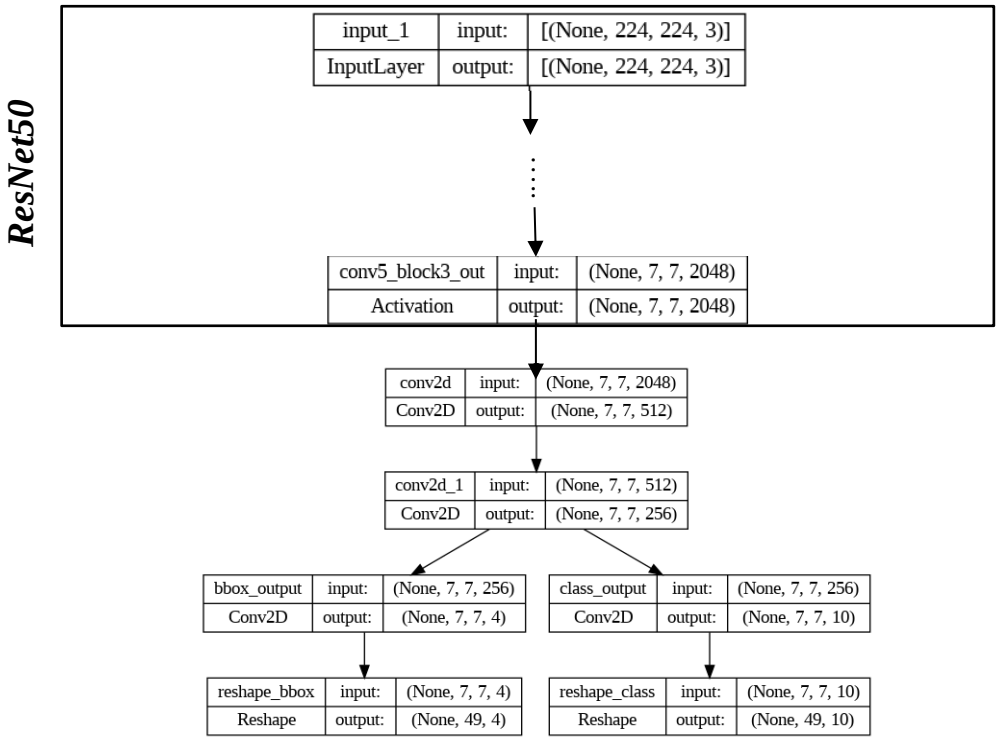

Figure 15: System Architecture Design

5.1.2. Model configuration and number of parameters

Our model is intended for item recognition and categorization in image processes input photos with three color channels (RGB) and a size of 224 x 224. It seems to be intended for object detection jobs. The input layer of the model is the first layer that feeds into the convolutional layers of the ResNet50 backbone network, which has already been trained. High-level characteristics are extracted using this backbone, which shrinks the spatial dimensions to 7x7 with 2048 channels. Further convolutional layers analyze these features after the backbone, lowering the channel dimensions successively to 512 and subsequently to 256. Next, two concurrent convolutional layers are applied to the generated feature maps: one predicts bounding box coordinates using four channels, and the other predicts class probabilities using ten channels, which correspond to ten distinct object classes. In order to prepare these outputs for additional processing or loss calculation, they are reshaped to 49x4 and 49x10, respectively. 34,208,910 parameters make up the model; 10,621,198 of those are trainable, while 23,587,712 are not. This architecture is appropriate for object detection applications because it makes use of the feature extraction capabilities of a pre-trained backbone while tailoring the output layers for the particular tasks of localization and classification.

Model:"model_1"

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 224, 224, 3)]	0	[]
.	.	.	.
.	.	.	.
.	.	.	.
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	['conv5_block3_add[0][0]']
conv2d_2 (Conv2D)	(None, 7, 7, 512)	9437696	['conv5_block3_out[0][0]']
conv2d_3 (Conv2D)	(None, 7, 7, 256)	1179904	['conv2d_2[0][0]']
bbox_output (Conv2D)	(None, 7, 7, 4)	1028	['conv2d_3[0][0]']
class_output (Conv2D)	(None, 7, 7, 10)	2570	['conv2d_3[0][0]']
reshape_2 (Reshape)	(None, 49, 4)	0	['bbox_output[0][0]']
reshape_3 (Reshape)	(None, 49, 10)	0	['class_output[0][0]']

Total params: 34208910 (130.50 MB)
Trainable params: 10621198 (40.52 MB)
Non-trainable params: 23587712 (89.98 MB)

Figure 16: The configuration of our model

5.1.3. Optimizer

In machine learning, an optimizer is an algorithm that is used to adjust the parameters of a model in order to minimize the error or loss function. The loss function represents the difference between the predicted output of the model and the true output, and the optimizer is responsible for finding the set of model parameters that will result in the lowest possible loss.

The choice of optimizer can have a significant impact on the performance of a machine learning model. Different optimizers use different algorithms to search for the optimal set of model parameters, and some may be better suited to certain types of data or model architectures than others.

Some popular optimizers used in machine learning include stochastic gradient descent (SGD), Adam, RMSprop, and Adagrad. These optimizers use various techniques such as adaptive learning rates, momentum, and gradient clipping to improve the efficiency and effectiveness of the optimization process.

a. ***Adadelta***: The Adadelta optimizer uses the shown equations (1, 2, 3, 4, 5, 6) to update the parameters:

$$\mathrm{RMS}[g]t = \sqrt{\mathrm{E}[g2]t + \varepsilon} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (1)$$

$$\mathrm{RMS}[\Delta x]t = \sqrt{\mathrm{E}[\Delta x2]t + \varepsilon} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (2)$$

$$\Delta x = -\frac{\mathrm{RMS}[\Delta x]t - 1\ \mathrm{RMS}[g]t \cdot g}{\mathrm{RMS}[g]t} \dots\dots\dots\dots\dots\dots\dots\dots (3)$$

$$\mathrm{E}[g2]t = \rho \cdot \mathrm{E}[g2]t - 1 + (1-\rho) \cdot g \dots\dots\dots\dots\dots\dots (4)$$

$$\mathrm{E}[\Delta x2]t = \rho \cdot \mathrm{E}[\Delta x2]t - 1 + (1-\rho) \cdot \Delta x \dots\dots\dots\dots\dots (5)$$

$$\theta t = \theta t - 1 + \Delta x \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (6)$$

Where:

- g is the gradient of the objective function with respect to the parameters
- ε is a small constant added for numerical stability
- ρ is a hyper-parameter that controls the decay rate of the moving averages
- E[g2]t is the exponentially decaying average of the squared gradient
- E[Δx2]t is the exponentially decaying average of the squared parameter updates
- RMS[g]t and RMS[Δx]t are the root mean square (RMS) values of g and Δx respectively at time step t
- θt is the vector of parameters at time step t.

The Adadelta optimizer is a variant of the Adagrad optimizer that addresses some of its limitations, such as the need to manually tune the learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients and uses the RMS value of the parameter

updates to adaptively adjust the learning rate. [53]

b. ***Adam***: Adam is an optimizer that adapts the learning rate for each parameter based on the first and second moments of the gradients. The update rule for Adam can be written as formulas (7, 8, 9, 10).

$$mt = \beta 1 * mt - 1 + (1 - \beta 1) * g \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (7)$$

$$vt = \beta 2 * vt - 1 + (1 - \beta 2) * g2 \dots\dots\dots\dots\dots\dots\dots\dots\dots (8)$$

$$mthat = \frac{mt}{1 - \beta 1t} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (9)$$

$$vthat = \frac{vt}{1 - \beta 2t} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (10)$$

Where:

- θ is the vector of parameters
- g is the gradient vector
- m is the exponentially weighted moving average of the gradient
- v is the exponentially weighted moving average of the squared gradient
- η is the learning rate
- β1 and β2 are the exponential decay rates for the moving averages
- t is the current iteration step
- ε is a small constant added for numerical stability to avoid division by zero.

The Adam optimizer computes the learning rate for each parameter by dividing the initial learning rate η by the square root of the biased-corrected second moment estimate vt. The first moment estimate mt is also bias-corrected to account for its initialization at zero. [53]

c. ***Adafactor*** : Adafactor is an optimizer that adapts the learning rate for each parameter, based on the geometry of the gradients. The update rule of Adafactor can be written as formulas (11, 12, 13, 14).

$$vt = vt - 1 + g^2 \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (11)$$

$$mt = \beta 1 * mt - 1 + (1 - \beta 1) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (12)$$

$$lr = \min(1.0, (1 - \beta 1) * lrt/\sqrt{(vt)}) \dots\dots\dots\dots\dots\dots\dots\dots\dots (13)$$

$$wt = wt - 1 + lr * mt \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (14)$$

Where:

- w is the vector of parameters
- g is the gradient vector
- v is an exponentially weighted moving average of the squared gradients,
- m is an exponentially weighted moving average of the gradients
- η is the learning rate

- ηt is an initial learning rate
- β1 is the exponential decay rate for the moving averages step t. [53]

d. ***Ftrl*** : The FTRL optimizer update rule can be written as formulas(14, 15). [53]

$$zt = zt - 1 + gt - (\sqrt{(nt + g2\ t)} - \sqrt{(nt)})/\alpha t * wt \ldots\ldots (14)$$

$$nt = nt - 1 + g2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (15)$$

Where:

- gt is the gradient of the loss function with respect to the weights
- zt and nt are accumulators for the weight updates
- vt is an accumulator for the scaling of the weight updates
- α is a time varying learning rate

e. ***Nadam***: or Nesterov-accelerated Adam, is a variant of the Adam optimizer that includes Nesterov momentum. The NAdam update rule can be written as formulas(16, 17). [53]

$$vt = \beta 2 * vt - 1 + (1 - \beta 2) * g_t^2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (16)$$

$$vthat = \frac{vt}{(1 - \beta 2t)f} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (17)$$

Where:

- gt is the gradient of the loss function with respect to the weights
- β2 are exponential decay rates for the moment estimates

f. ***SGD*** : The stochastic gradient descent (SGD) optimizer update rule can be written as formula(18).

$$wt = wt - 1 - \alpha * gt \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (18)$$

Where:

- wt represents the updated weights, gt is the gradient of the loss function with respect to the weights, α is the learning rate. [53]

g. ***RMSprop*** : The RMSprop optimizer update rule, can be written as formulas(19, 20). [53]

$$cachet = decayrate * cachet - 1 + (1 - decayrate) * g2 \ldots (19)$$

$$wt = wt - 1 - \alpha * \frac{gt}{\sqrt{(cachet) + \varepsilon)}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (20)$$

Where:

- wt represents the updated weights (cachet)+ε), gt is the gradient of the loss function with respect to the weights
- Cachet is the moving average of the squared gradients, decay rate is the decay rate for the moving average, α is the learning rate, ε is a small constant to prevent division by zero.

6. Distance estimation

6.1. Method Selection for Distance Calculation

In chapter 1, we discussed various methods for calculating the distance between a camera and detected objects, which are based on deep learning techniques. Our system proposes an algorithm to measure this distance, providing a more accurate and efficient approach to estimating the distance between objects.

Improved Driver Assistance

By calculating distances, our system can provide more accurate feedback to drivers, such as alerts about impending crashes or recommendations for safe following distances. This enhanced driver assistance can help prevent accidents and improve road safety.[50]

Better Data Analysis

The distance calculations generated by our system provide valuable data for analysis, which can be utilized to optimize traffic flow, improve overall safety, and manage traffic better. By analyzing these data, traffic managers can make more informed decisions to reduce congestion, improve road safety, and enhance the overall driving experience.[50]

6.2. Explaining the algorithm

In this phase of the study, after items are successfully identified in the image, the emphasis switches to calculating the separations between and from the camera of the objects that have been identified. The program has been carefully designed to examine the submitted image in a methodical manner. It begins by analyzing the image and using an object identification model that has been trained to recognize items in it. The method then establishes the foundation for distance estimation by extracting bounding boxes from the surroundings of the objects that have been spotted. By utilizing the object's known width and the camera's focal length, the algorithm iteratively determines each object's distance from the camera. These calculated distances are carefully saved for later examination together with the bounding box center coordinates and object IDs. Additionally, the method adds value to the visual representation by overlaying labels and rectangles onto the image, which helps to graphically express the items that have been recognized together with their estimated distances.

Adding to its capabilities, the method uses the Euclidean distance, as shown in formula (21) to calculate the distances between pairs of objects that have been detected.

$$D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \quad \text{.....(21)}$$

Where:

- (x1, y1) are the coordinates of one point.
- (x2, y2) are the coordinates of the other point.
- D is the distance between (x1, y1) and (x2, y2).

In the end, the algorithm saves the altered picture with the objects annotated, giving important details on the spatial arrangement and relationships between the elements in the image. The psedo-code below resume the estimation distance algorithm.

```
function process_image(image_path):
    // Read the image from the given path
    image = read_image(image_path)

    // Detect objects in the image using the pre-trained model
    results = detect_objects(image)

    // Get the bounding boxes of the detected objects
    boxes = extract_boxes(results)

    // Initialize lists to store centers, object IDs, and distances to camera
    centers = []
    object_ids = []
    distances_to_camera = []

    // Loop through each detected object
    for idx, box in enumerate(boxes):
        // Calculate the center of the bounding box
        center_x, center_y = calculate_center(box)

        // Calculate the width of the object in the image
        width = calculate_width(box)

        // Calculate the distance to the camera using known width and focal length
        distance_to_camera = calculate_distance(width)

        // Store the center, object ID, and distance to camera
        centers.append((center_x, center_y))
        object_ids.append(idx)
        distances_to_camera.append(distance_to_camera)

        // Draw a rectangle and label on the image to visualize the detected object
        draw_rectangle(image, box)
        draw_label(image, idx, distance_to_camera, box)

    // Calculate distances between pairs of objects
    distances = calculate_distances(centers, object_ids)

    // Save the modified image with bounding boxes and labels
    output_image_path = save_image(image)

    // Return the path of the output image, distances between objects, and distances to camera
    return output_image_path, distances, distances_to_camera

// Function to calculate Euclidean distance between two points
function euclidean_distance(point1, point2):
    return sqrt((point1[0] - point2[0])^2 + (point1[1] - point2[1])^2)
```

Figure 17: Pesdo-code of distance estimation algorithm

7. Model Evaluation

7.1. Metrics

We used the categorical cross-entropy loss and accuracy metrics for the model evaluation, Mean Squared Error (MSE) ocuses on the precision of the predicted coordinates and Intersection over Union (IoU) that can help in improving the overall localization performance.

a. ***Cross Entropy Loss***: The categorical cross-entropy loss function estimates the loss of an example by calculating the formula (22).

$$L = -\sum_{c=1}^{c} y_c \log(p_c) \text{……..…….……(22)}$$

Where:

- y_c is the true label for class c (a one-hot encoded vector).
- $p_c$is the predicted probability of class c

b. ***Accuracy***: This metric describes how the model performs across all classes. It is helpful when all classes are equally important. It is evaluated as the ratio between the number of correct predictions to the total number of predictions as shown in formula (23).

$$Accuracy = \frac{TP+TN}{TP+FP+FN} \text{…….……(23).}$$

Where:

- TP = true positives (number of correctly predicted positive examples)
- TN = true negatives (number of correctly predicted negative examples)
- FP = false positives (number of incorrectly predicted positive examples)
- FN = false negatives (number of incorrectly predicted negative examples)

c. ***Mean Squared Error (MSE)***: is primarily used to measure the accuracy of continuous predictions. In the context of object detection, it is often used for bounding box regression, which involves predicting the coordinates of the bounding boxes around objects. MSE is calculated as the average of the squared differences between the predicted bounding box coordinates and the ground truth coordinates, as shown in formula (24).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\text{yi} - \text{y\^i})\text{\^2} \text{…….……(24)}$$

Where:

- yi is the actual coordinate.
- y^i is the predicted coordinate.
- n is the number of coordinate.

d. ***Intersection over Union (IoU***): IoU measures the accuracy of the predicted bounding box by evaluating how much it overlaps with the ground truth bounding box. It provides a measure of localization accuracy. IoU is the ratio of the area of overlap between the predicted bounding box and the ground truth bounding box to the area of their union as shown in formula (25)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of union}} \quad \text{.....(24)}$$

The figure below explain the formula (25)

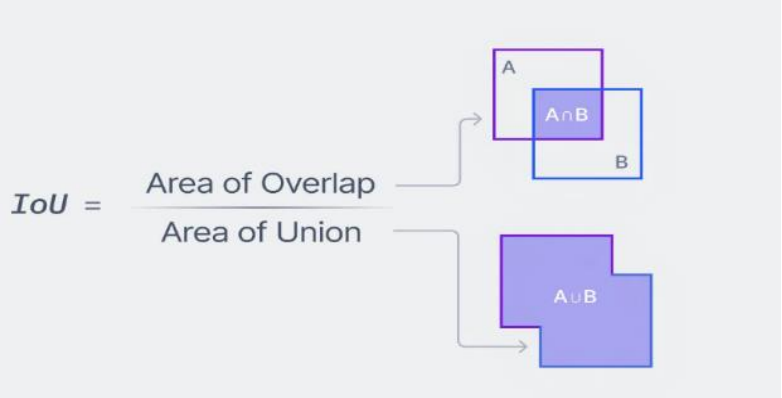

Figure 18: IoU [51]

The figure below represent that IoU evaluates the overall quality of the bounding box prediction by considering both the size and the position of the predicted bounding box relative to the ground truth. Higher IoU values indicate better overlap and more accurate localization.

Figure 19: IoU comparative performance [51]

8. Conclusion

Based on the information provided, the "Conceptual Study" chapter appears to cover four main aspects of the project:

- **The Design of the System and The Dataset Used in the Project**: This section outlines the architecture of the entire system, detailing the various components and their interactions. For the dataset, we include details about the its size and composition, providing context for the data used in training and evaluating the model.
- **Object Detection**: Here, the proposed model is discussed in detail, including the concept and architecture of the model. This section provides a detailed description of the specific neural network architecture used in the project, along with information about the number of parameters and any hyper-parameters tuned during model training. Additionally, a detailed description of the optimizer used in the project is included.
- **Distance Estimation**: This section details the proposed algorithm for distance estimation, including a pseudo-code that summarizes the distance estimation algorithm.
- **Model Evaluation**: presents an analysis of how well the model performed in terms of accuracy, loss, mean squared error (MSE), and intersection over union (IoU).

Overall, this chapter provides a high-level overview of the key elements of the project, including the problem being addressed, the data used to train the model, the architecture and configuration of the model itself, and model evaluation. This information is important for understanding the approach taken in the project and the potential strengths and limitations of the final model.

Chapter 03:
Results and Implementation work

1. Introduction

This chapter is divided into four main sections:

- The first section outlines the chosen tools and technologies for the system development. It provides an overview of the programming languages, libraries, and frameworks used in the implementation of the system. The tools chosen for the project are discussed in detail.
- The second section discusses and compares the results obtained from the system testing. The evaluation metrics used to assess the performance of the system are presented, and the results are analyzed and compared against the expectations and requirements of the system.
- The third section, its comparison between our system and one of the used method in objects detection **Faster R-CNN**.
- The fourth and final section of the chapter presents the system interface and displays a screenshot of the test phase.

Overall, this chapter provides a comprehensive overview of the development and testing of the computer vision system, from the tools and technologies used to the final user interface and test results

2. Representation of the development tools

a. Physical environment

Component	Brand/Model	Specification
CPU	Intel (R) core(TM) i7-8750H	2.20 GHz
GPU	Intel (R) UHD	630
RAM	16 Go Bicanal DDR4	2 666 MHz
Motherboard	Dell Inc.	Mobile Intel CM246
Storage	SSD	128 GO SSD

Table 3: Desktop Hardware

b. Software and libraries used in the implementation

Operating system: 64-bit Windows 10.

Python: Guido van Rossum first introduced Python, a high-level interpreted programming language, in 1991. With a syntax that prioritizes readability and simplicity of code, it is made to be simple to write, read, and maintain. Python is a widely used option for many different kinds of applications. Including automation, data analysis, scientific computing, web development, and artificial intelligence.[42]

OpenCV: Open Source Computer Vision (OpenCV) is a library that provides a collection of over 2500 computer vision and image processing algorithms that are accessible via an API for Python, C, and C++. It is distributed under a BSD license. gratuit pour toutes les médias. A community with over 40,000 active members now develops, maintains, documents, and uses the OpenCV library. [43]

Numpy: is a well-known open-source toolkit for scientific computing in Python, short for Numerical Python. It offers an array object that is multidimensional, a variety of mathematical operations, and array-related tools. Numerous scientific computer domains, including physics, economics, engineering, and machine learning, among others, heavily rely on NumPy. [44]

Flask: A lightweight framework for WSGI web applications is called Flask. It can be scaled up to complex applications and is made to be quick and simple to start using. It started out as a straightforward wrapper for Jinja and Werkzeug and has grown to become one of the most widely used frameworks for

Python online applications.

Although it doesn't impose any dependencies or project layout, Flask makes recommendations. The developer is in charge of selecting the libraries and tools they choose to utilize. The community has produced a large number of extensions that make it simple to add new functionality. [45]

Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, Python/IPython shells, web application servers, and various graphical user interface toolkits.[54]

3. Experiments, discussion the obtained results

In this section, we will experiment with several optimization techniques and to further enhance the performance of our model.

Before we discuss the results, it is essential to understand them. We plotted various metrics, including loss, accuracy, mean squared error (MSE), and intersection over union (IoU), as mentioned in Chapter 2. The system will generate the following plots:

- For localizing: Bounding Box Regression Model MSE and Bounding Box Regression IoU.
- For classification: Classification Model accuracy and Classification Model loss.
- For the total system: Total model accuracy and Total model loss.

Chapter 03: Results and Implementation work

a. Adam:

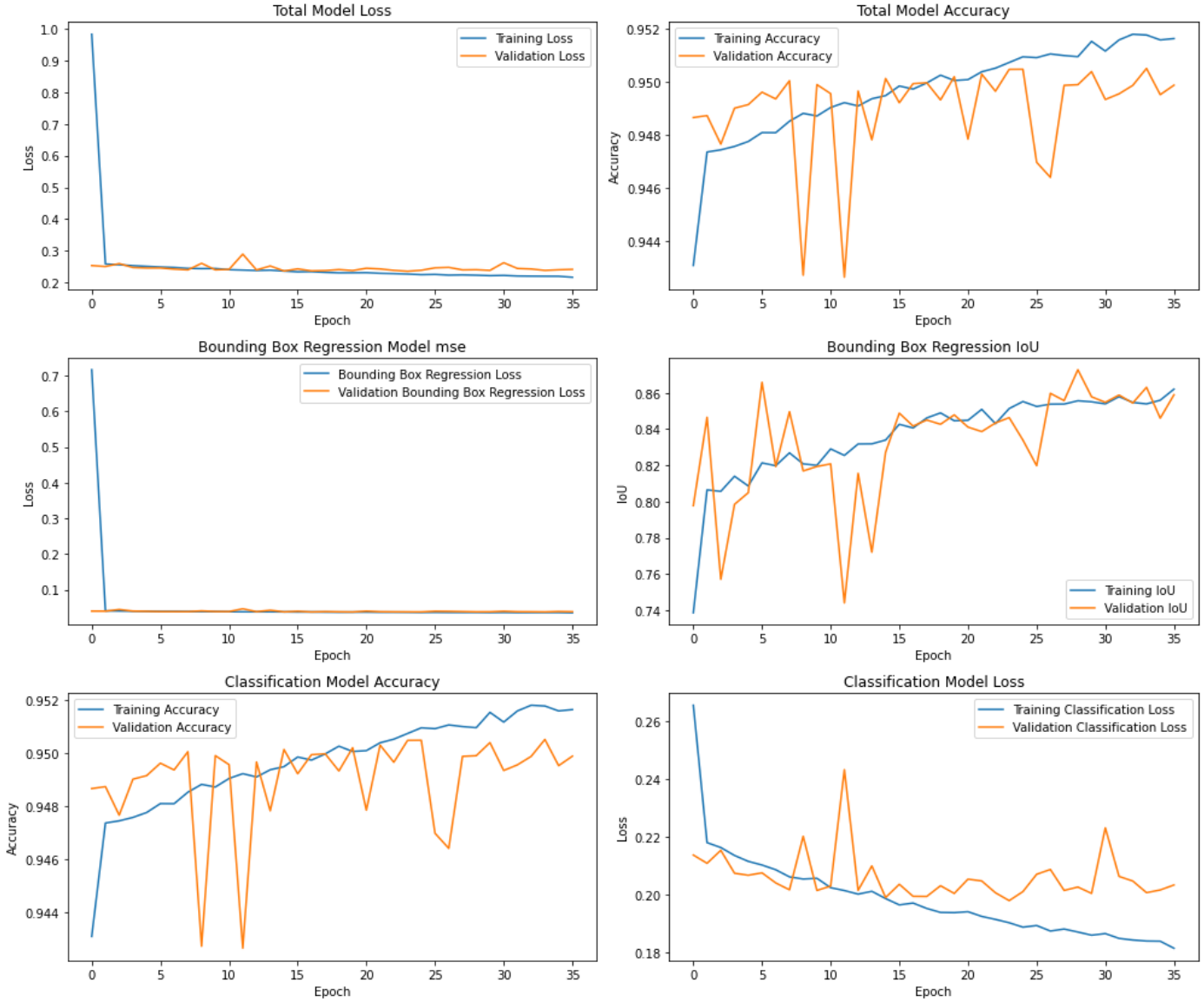

Figure 20: Charts of adem's results.

Great to hear that our model achieved a high accuracy of 0.9518 in Epoch 36 with a low loss of 0.2170 in the training set. Furthermore, our validation accuracy is also high at 0.9499 with a low validation loss of 0.2415, indicating that our model is performing well on both the training and validation data. This is a good sign that our model is generalizing well to unseen data and can be used for predicting new data. The classification model accuracy and validation accuracy are similarly high at 0.9518 and 0.9499, respectively. The Mean Squared Error (MSE) and validation MSE are low at 0.0351 and 0.0381, respectively, while the Intersection over Union (IoU) and validation IoU are also high at 0.8611 and 0.8590, respectively, further confirming the model's robustness.

b. Adadelta:

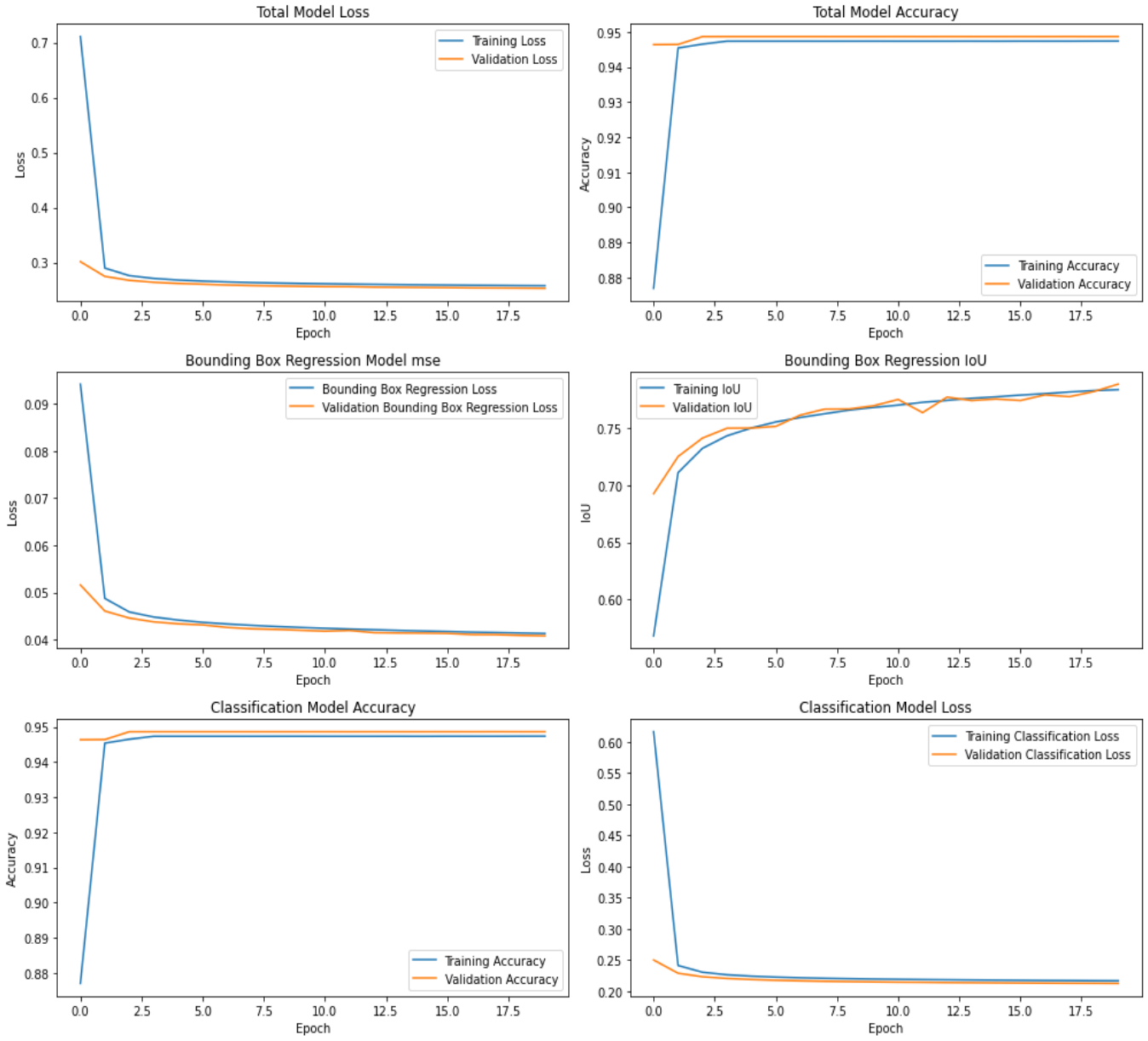

Figure 21: Charts of Adadelta's results.

Great to hear that the model achieved a high accuracy of 0.9469 in Epoch 20 with a low loss of 0.2604 in the training set. Furthermore, the validation accuracy is also high at 0.9487 with a validation loss of 0.2530, indicating that the model is performing well on both the training and validation data. This is a good sign that the model is generalizing well to unseen data and can be used for predicting new data.

c. RMSprop:

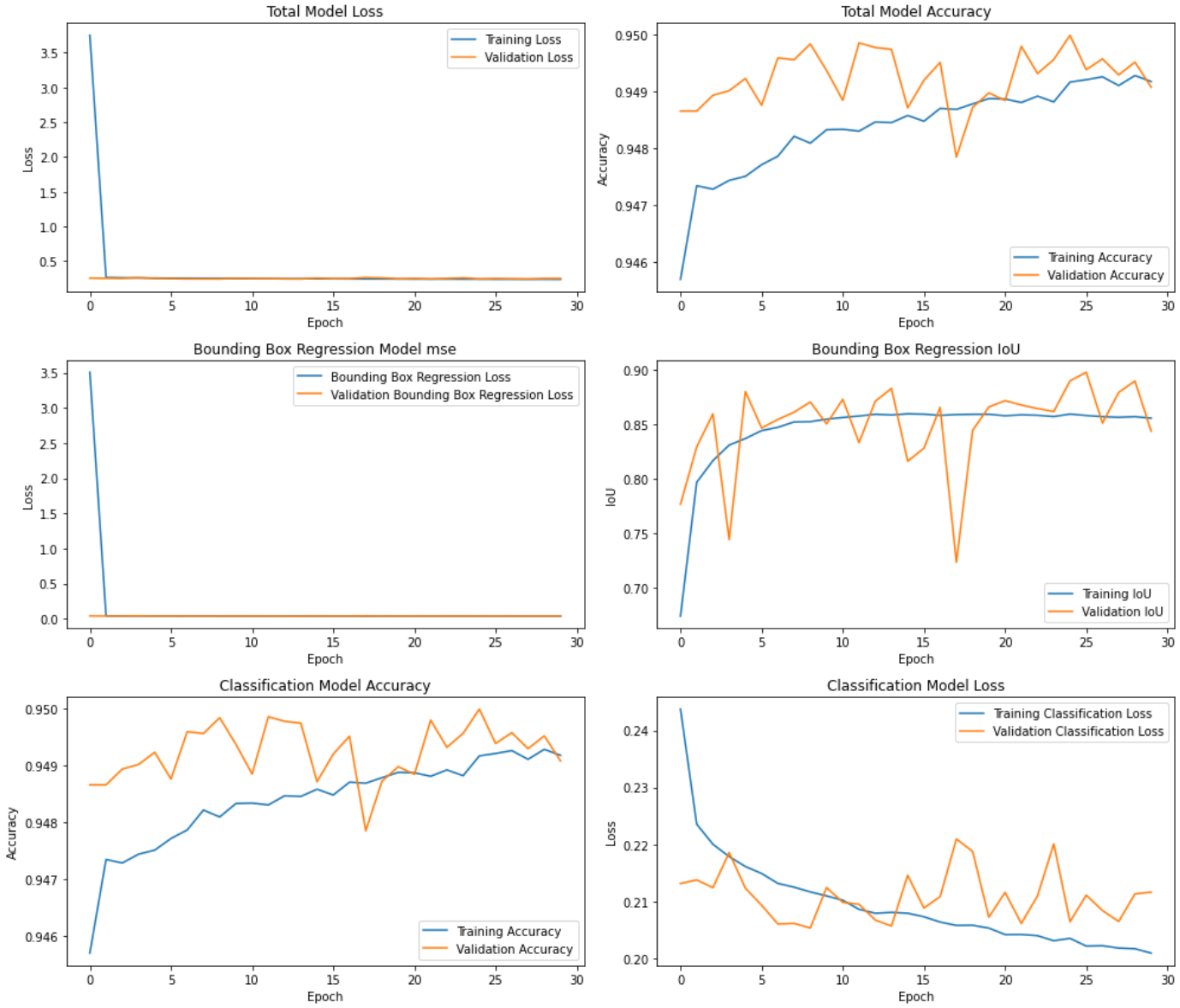

Figure 22: Charts RMSprop of results.

Great to hear that the model achieved a high accuracy of 0.9493 in Epoch 30 with a low loss of 0.2361 in the training set. Furthermore, the validation accuracy is also high at 0.9491 with a validation loss of 0.2500, indicating that the model is performing well on both the training and validation data. This is a good sign that the model is generalizing well to unseen data and can be used for predicting new data.

d. SGD:

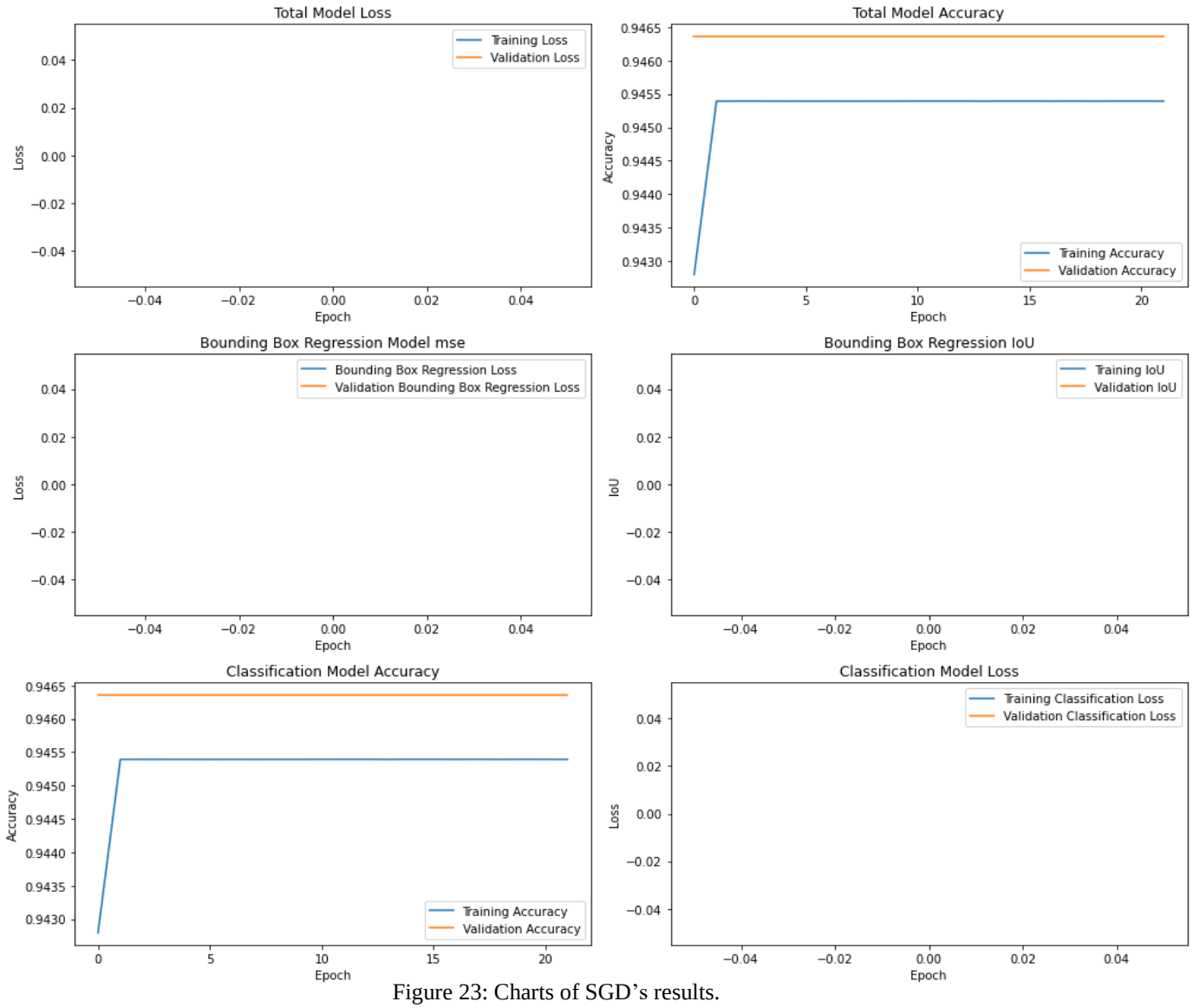

Figure 23: Charts of SGD's results.

It seems like the model did not perform well when using the SGD optimizer. The training accuracy was 0.9450 in Epoch 22, but there was no validation accuracy recorded, indicating a potential issue during the training process. Both training and validation losses are not available (nan), suggesting that the training might not have completed properly or there were issues with the data or the model configuration. This indicates that the model did not converge and further investigation is needed to identify and resolve the underlying issues.

e. Fltr:

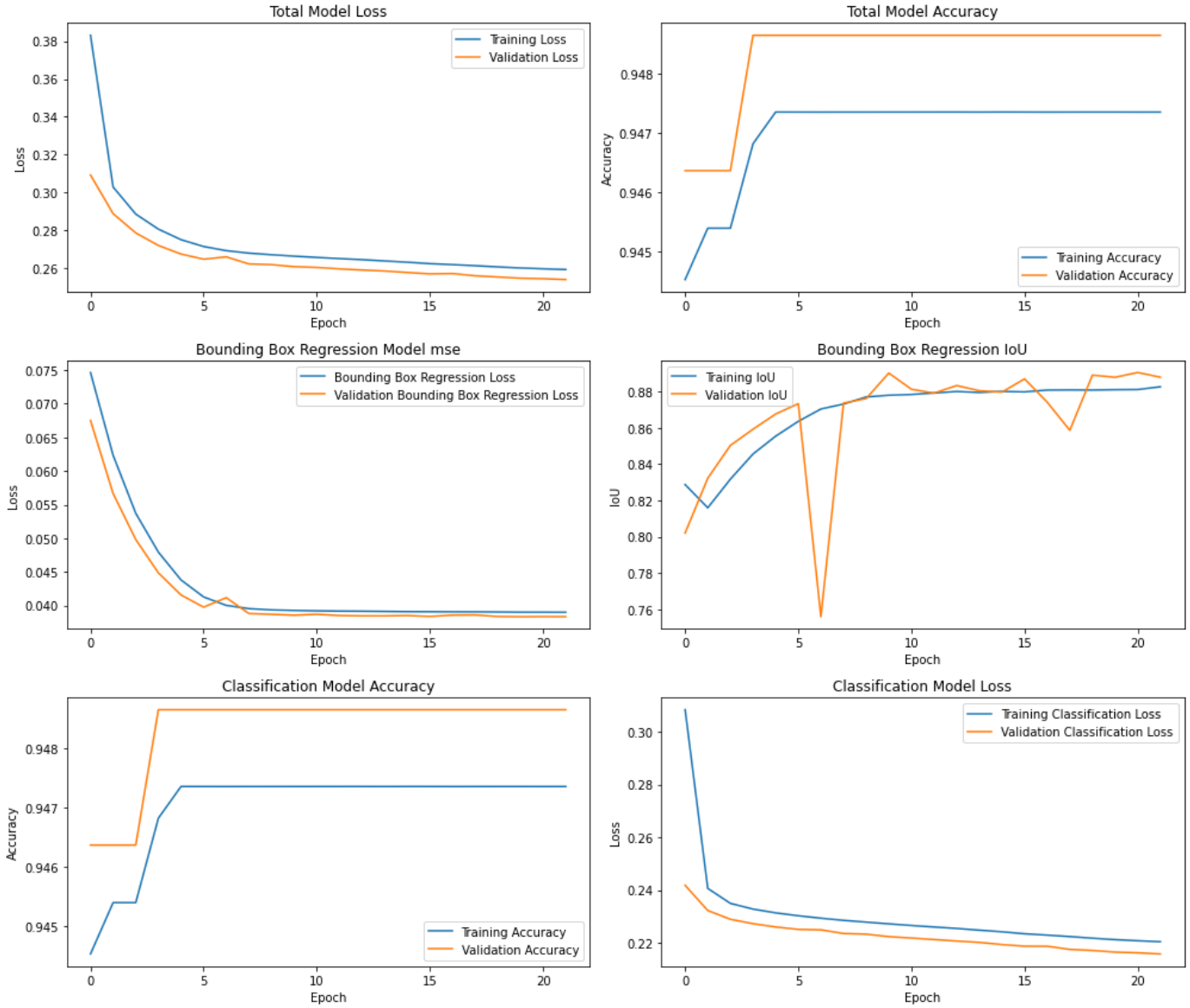

Figure 24: Charts of FLTR's results.

the model achieved a high accuracy of 0.9487 in Epoch 22 with a low loss of 0.2540 in the validation set. Furthermore, the training accuracy is also high at 0.9478 with a training loss of 0.2549, indicating that the model is performing well on both the training and validation data.

Additionally, the bounding box regression model shows impressive performance with a low training MSE of 0.0387 and validation MSE of 0.0383. The IoU metrics are also strong, with a training IoU of 0.8833 and a validation IoU of 0.8879, suggesting the model's predictions for bounding boxes are precise.

Overall, these results indicate that the model is generalizing well to unseen data and can be confidently used for predicting new data.

f. Adafactor:

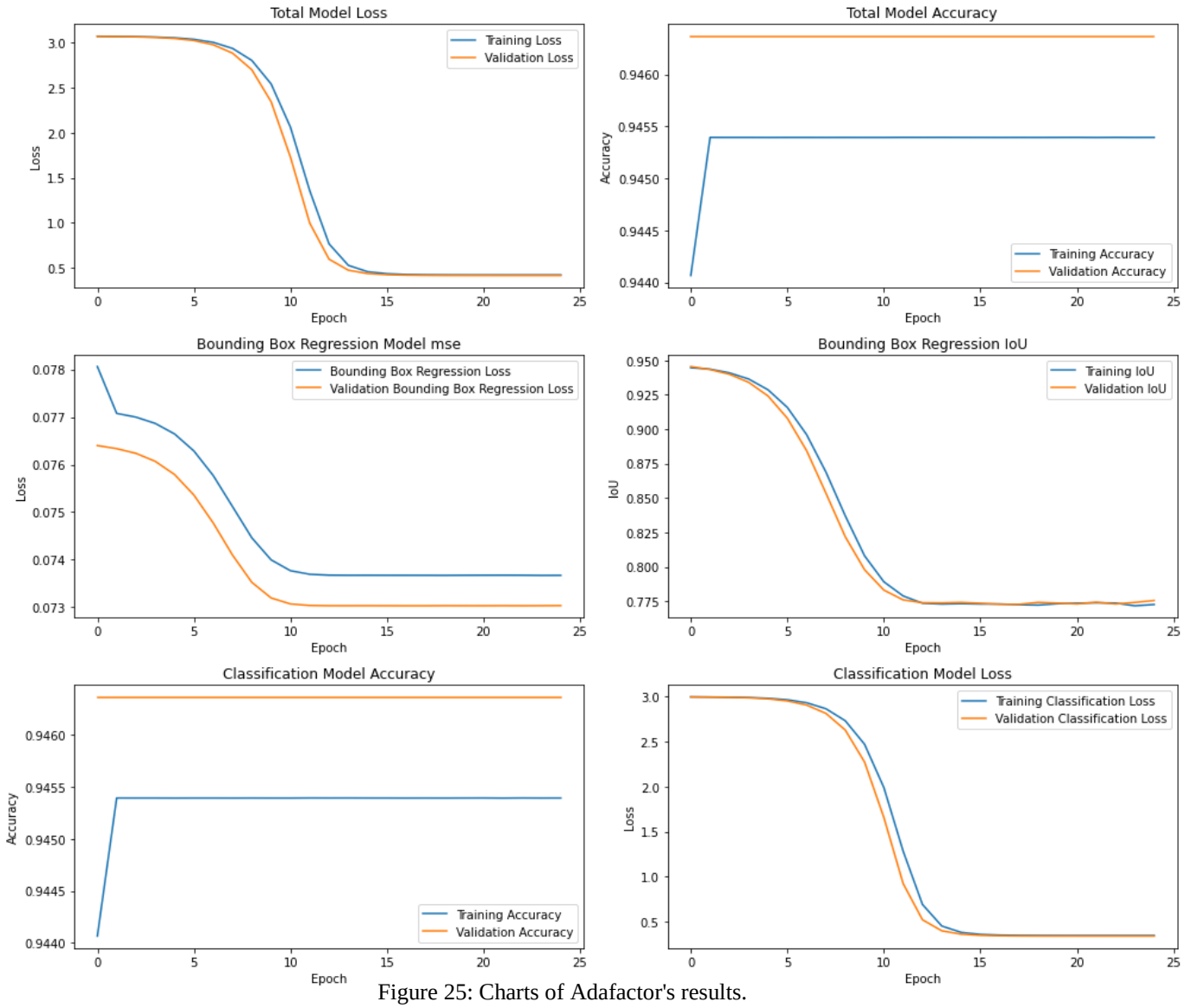

Figure 25: Charts of Adafactor's results.

It looks like the model did not perform very well in this case, with low accuracy and high loss values both for the training and validation sets. While the total model loss decreased significantly, the accuracy for both training and validation remained relatively low and flat throughout the training process.

The accuracy and validation accuracy are both around 0.9469 and 0.9487, respectively, which are not very high. The losses for both training and validation are also high, indicating the model is not fitting the data well. The Bounding Box Regression Model's IoU decreased, further suggesting poor model performance. The training and validation losses show a downward trend, but they are still high, implying potential issues with the model's architecture, data preprocessing, or insufficient training data.

Overall, it seems like the model is not performing well with low accuracy and high loss values.

g. Nadam:

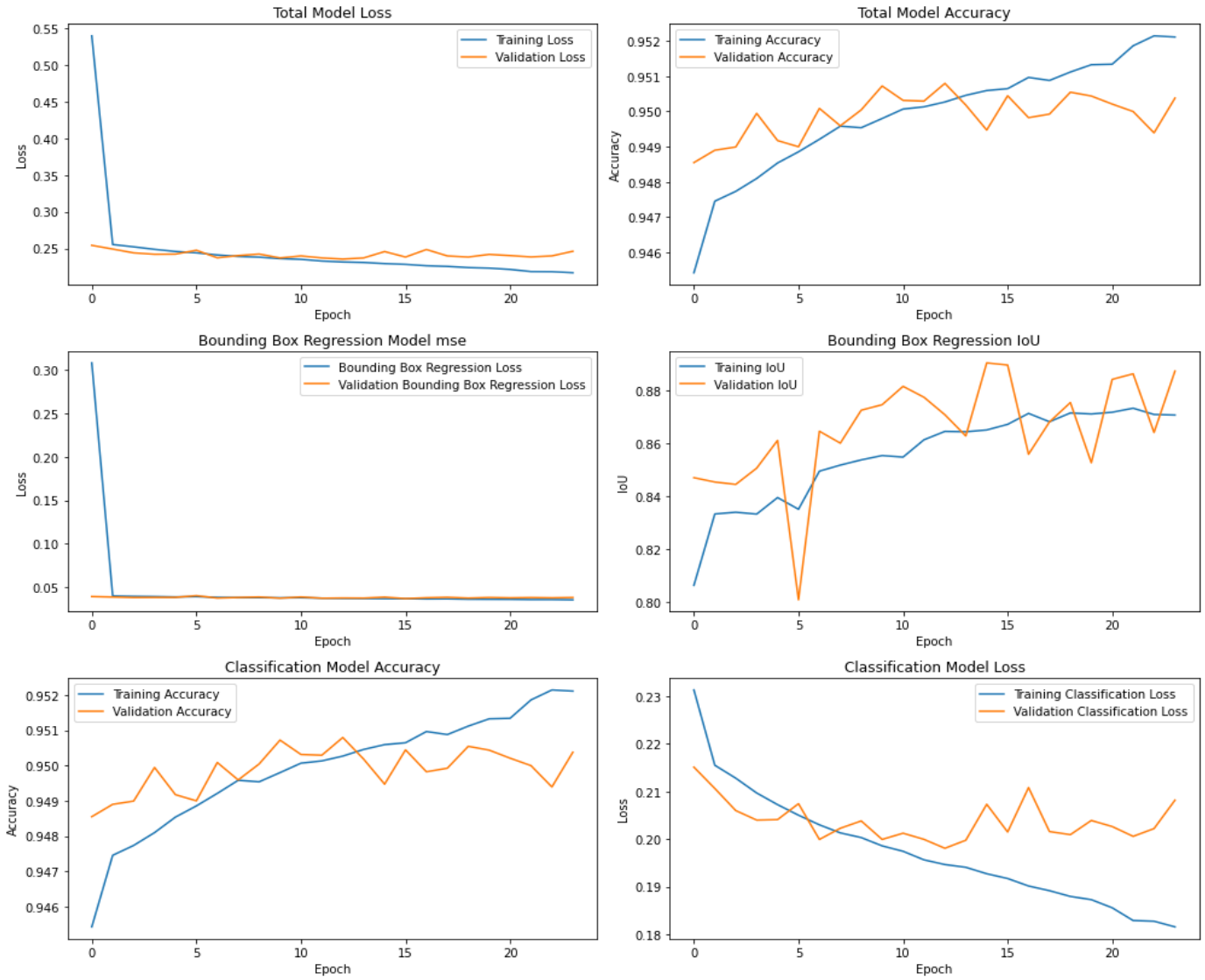

Figure 26: Charts of Nadam's results.

the model achieved a high accuracy of 0.952 in Epoch 20 with a low loss of 0.216 in the training set. Furthermore, your validation accuracy is also high at 0.950 with a validation loss of 0.246, indicating that the model is performing well on both the training and validation data.

Additionally, the bounding box regression model shows an impressive performance with a low training MSE of 0.0347 and validation MSE of 0.0380. The IoU metrics are also strong, with training IoU at 0.8741 and validation IoU at 0.8875, suggesting the model's predictions for bounding boxes are precise.

Overall, these results indicate that the model is generalizing well to unseen data and can be confidently used for predicting new data.

Optimizer	Loss	Val_loss	Accuracy	Val_accuracy	Class_Accuracy	Val_Class_Accuracy	MSE	Val_MSE	IoU	Val_IoU	epoch	Batch size
Adam	0.2170	0.2415	0.9518	0.9499	0.9518	0.9499	0.0351	0.0381	0.8611	0.8590	36	24
Adadelta	0.2604	0.2530	0.9469	0.9487	0.9469	0.9487	0.0421	0.0408	0.7823	0.7883	20	8
RMSprop	0.2361	0.2500	0.9493	0.9491	0.9493	0.9491	0.0369	0.0384	0.8575	0.8438	30	8
SGD	nan	nan	0.9450	nan	0.9450	nan	nan	nan	nan	nan	22	12
FTRL	0.2549	0.2540	0.9478	0.9487	0.9478	0.9487	0.0387	0.0383	0.8833	0.8879	22	8
Adafactor	0.4315	0.4155	0.9438	0.9464	0.9438	0.9464	0.0758	0.0730	0.7711	0.7755	25	8
Nadam	0.2162	0.2462	0.9521	0.9504	0.9521	0.9504	0.0347	0.0380	0.8741	0.8875	23	10

Table 4: Comparison of different optimizers.

This table shows the performance metrics (accuracy and loss) for different optimization algorithms along with the corresponding validation metrics and the number of epochs. The best-performing algorithm based on validation accuracy and loss is highlighted in green, while the worst-performing algorithm is highlighted in red. Based on the provided results, it seems like the best optimizer (model) used is Nadam. It has a high accuracy of 0.9521 and a low loss of 0.2162 on the training data, as well as a high validation accuracy of 0.9504 and a low validation loss of 0.2462. Additionally, it achieved these results in only 23 epochs.

Based on these results, the **Nadam optimizer** stands out as the most effective, achieving the highest accuracy and the lowest loss in both training and validation datasets, demonstrating excellent generalization and performance. Conversely, the **Adafactor optimizer** shows the poorest performance, with the highest loss and the lowest accuracy, indicating suboptimal model training and validation results.

The system's classification accuracy, when using the Nadam optimizer, was determined to be 95.21%. This high accuracy indicates the model's effectiveness in correctly identifying and categorizing various traffic objects such as cars, pedestrians, and bicycles.

In addition to classification accuracy, the system's low classification loss and MSE, coupled with a high IoU, suggest that the model is not only accurate in detecting objects but also reliable in localizing objects. The Nadam optimizer's performance highlights its suitability for this specific application, making it the preferred choice for optimizing the traffic object detection.

4. Comparison with Faster R-CNN

Metric	Our Model	Faster R-CNN[55]
Dataset	Pascal voc 2012	Pascal voc 2012
Optimizer	Nadam	SGD with momentum
Loss	0.2162	0.2 - 0.3
Validation Loss	0.2462	0.2 - 0.3
Accuracy	0.9521	Higher (mAP > 80%, indicating high accuracy)
Intersection over Union (IoU)	0.8741	0.6 - 0.8
Validation IoU	0.8875	0.6 - 0.8
Parameters	~30 million	~60 million

Table 5: comparison table based on our provided metrics and typical Faster R-CNN performance

Using the PASCAL VOC 2012 dataset, the comparison table shows the performance differences and similarities between your custom object identification model and Faster R-CNN. A number of important performance indicators were assessed, such as training features, IoU, and loss.

With Nadam optimization, our unique model attains competitive IoU values of 0.8741 and 0.8875, as well as a respectable accuracy of 95.21%. This suggests good performance in item identification inside images and high detection skills. On the other hand, because of its improved training regimen and network architecture, Faster R-CNN, which uses SGD with momentum and has a more intricate architecture with about 60 million parameters, usually produces lower loss values and maybe higher precision.

Both models show identical IoU and validation metrics on the PASCAL VOC 2012 dataset, indicating similar efficacy in object detection tasks. But our model's long training time per epoch more than 500 seconds highlights a possible disadvantage when contrasted with Faster R-CNN, which gains from better-optimized training protocols and possibly quicker convergence.

In summary:

Loss a Loss and Validation Loss: our model's loss is slightly lower than Faster R-CNN's typical range, which indicates that your model might be performing better on the training data. And with 95.21% accuracy

IoU and Validation IoU: our model's IoU (Intersection over Union) is higher than Faster R-CNN's typical range, which suggests that your model is more accurate in detecting objects.

Number of Parameters: Your model has fewer parameters than Faster R-CNN, which could make it more efficient and easier to train.

Training Time per Epoch: Your model takes significantly longer to train per epoch than Faster R-CNN, which could be due to differences in hardware, optimization techniques, or model complexity.

As primer vue, we see that our models more prefermence but we don't forgat that Faster R-CNN is trained on:

Hardware: NVIDIA Tesla V100 GPU, 12 GB RAM.

Software: CUDA-enabled GPU, Python with TensorFlow and PyTorch.

Our hardware and software are too modest comparing to faster's R-CNN

Then, considering the trade-off between accuracy and computational resources, I would say that our model has a slightly better performance than Faster R-CNN. However, the difference is relatively small, and both models are still performing well. In the end, the choice is based on particular application needs, such as the requirement for real-time processing, the effectiveness of the training process, and the difficulty of the detection task.

5. Representing our system interface

The system's interface display a page titled "Be Safe." The user interface on this page permits uploading images.

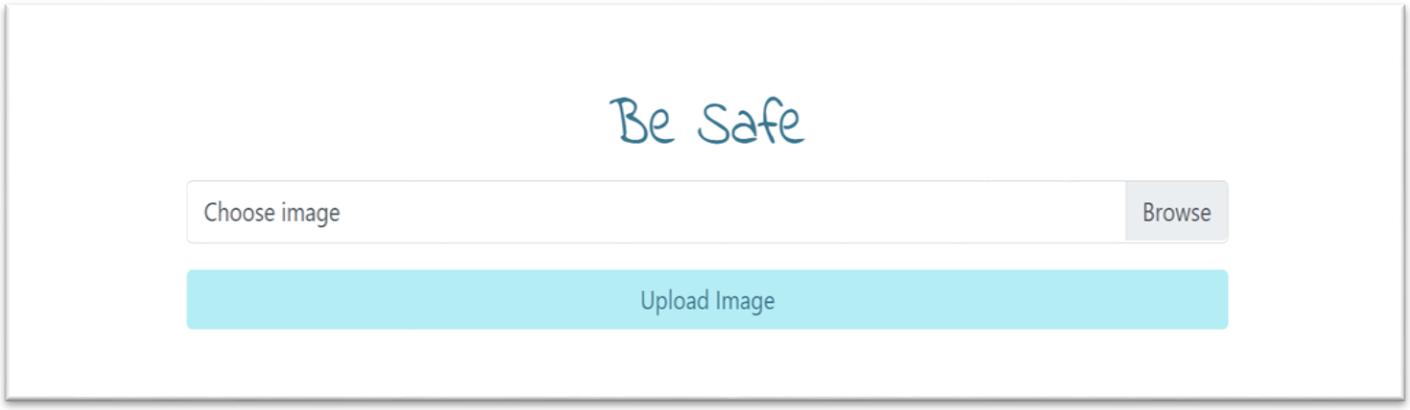

Figure 27: The system's interface.

We click on browse to choose an image.

Figure 28: Browse button.

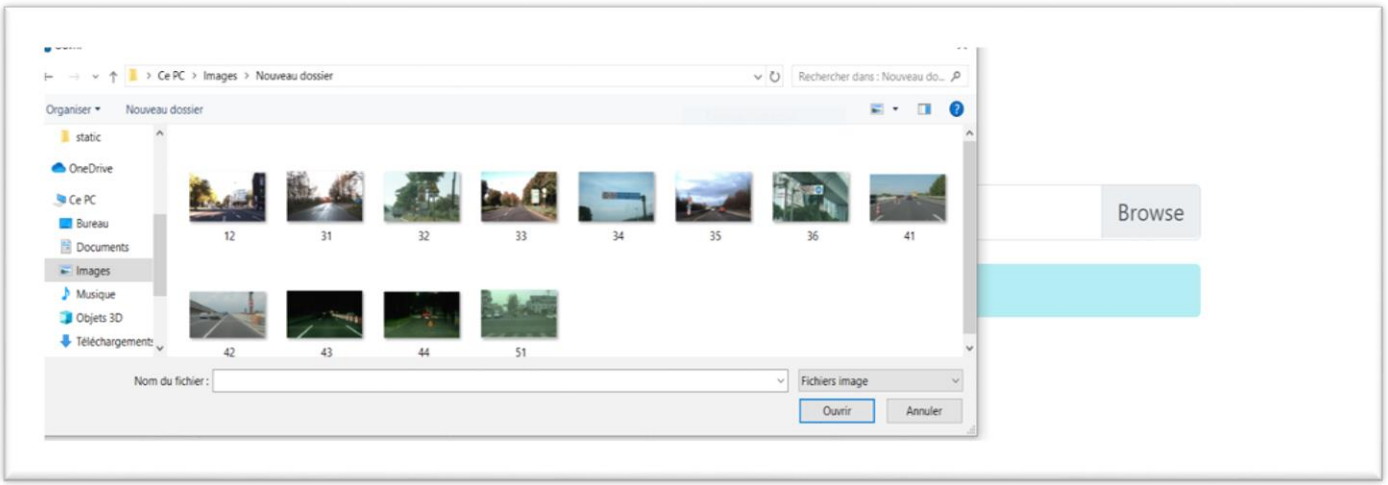

Figure 29: Choosing image.

Chapter 03: Results and Implementation work

After chosing an image we click in upload image.

Figure 30: Upload button.

And here is the results a section labeled "object detection" displays a picture of a road with several objects highlighted in green boxes that have been detected. In the top of the box we have the id of the detected object and the distance between the camera and this object in metres, for the distances between pairs of the detected items are listed in the "Distance Estimation" labeled section on the right. The distances are displayed with IDs and are given in meters.

For the figure below, the system detect only one object that's why we have no distance to display.

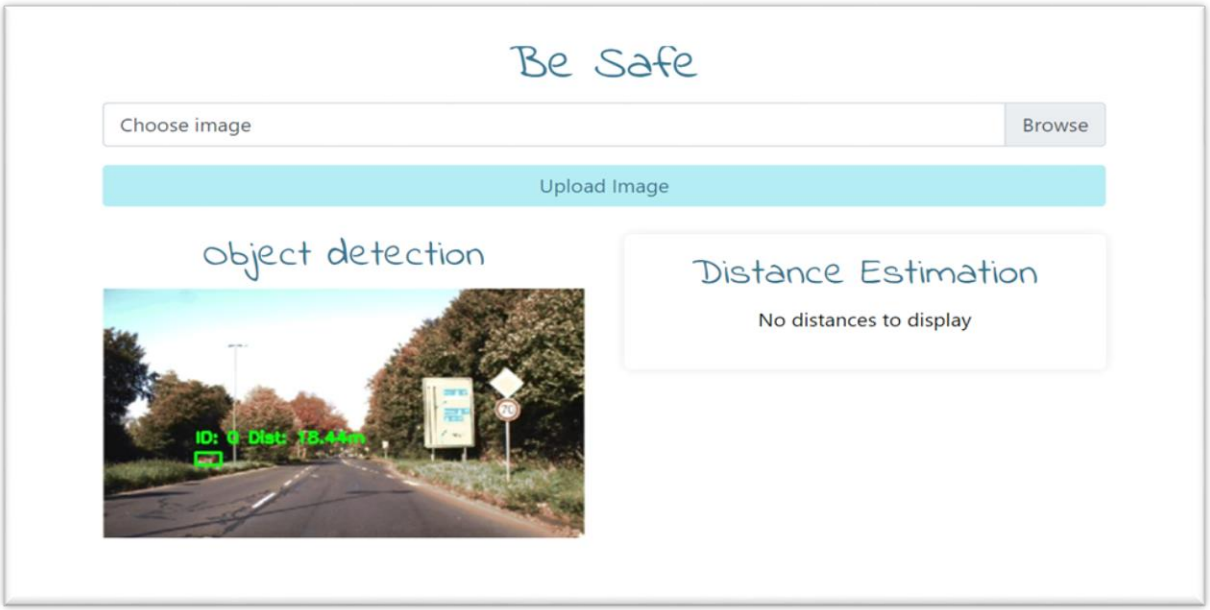

Figure 31: the results.

Chapter 03: Results and Implementation work

For the figure below, the system detect more than object that's why we have no distance to display.

On the left in the "Object detection" labeled section, the interface shows the image with detected objects and distance estimation between camera and objects.

On the right in the "Distance Estimation" labeled section the interface shows a list of distance between pairs of detected objects, for example: distance between ID 0 and ID 1 is 1.19855meters, the distance between ID 2 and ID 4 is 0.248399 meters.

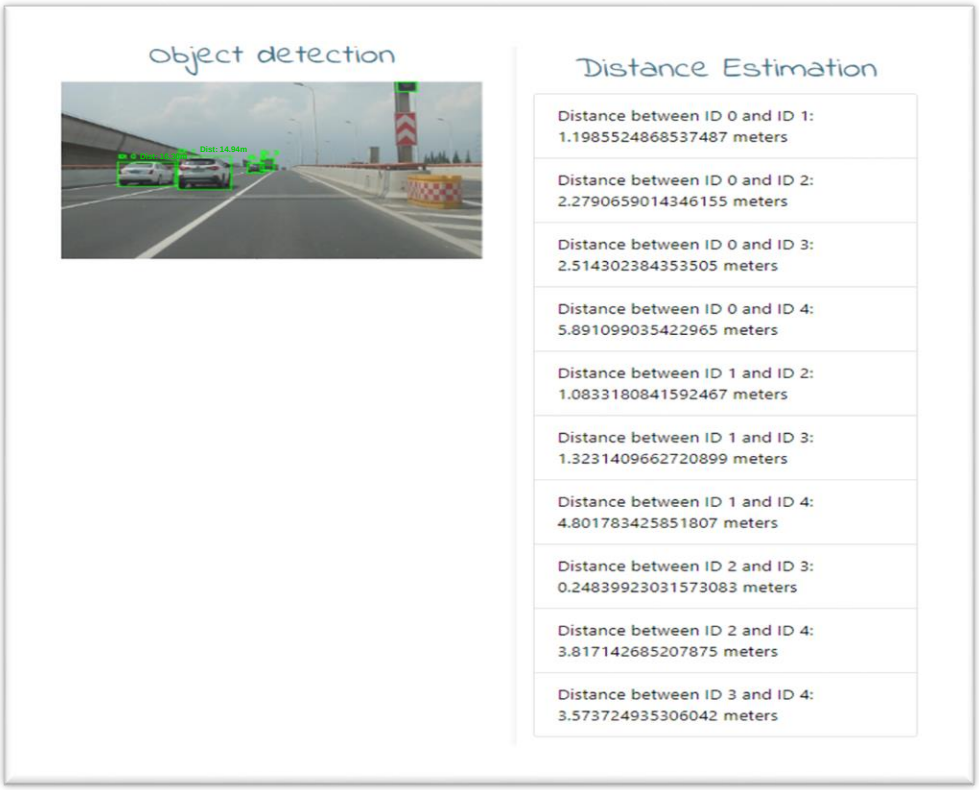

Figure 32: the results 2.

For the name's id objects its affiched in the cmd not in the image detected, as shown in the image below.

```
0: 352x640 3 cars, 1 truck, 1 traffic light, 729.0ms
Speed: 6.0ms preprocess, 729.0ms inference, 938.4ms postprocess per image at shape (1, 3, 352, 640)
127.0.0.1 - - [23/May/2024 13:03:50] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [23/May/2024 13:03:50] "GET /static/output.jpg HTTP/1.1" 200 -
```

Figure 33: Objects names.

6. Conclusion

This chapter has presented the tools and technologies used in the development of the traffic object detection and distance estimation system, detailed the experiments conducted to evaluate the system's performance. The chosen tools and technologies, including specific programming languages, libraries, and frameworks, played a crucial role in the successful implementation of the system.

The evaluation metrics indicated that the system performed well, meeting the expectations and requirements set at the beginning of the project. The results demonstrated the system's capability to accurately detect and classify various road entities and estimate their distances effectively. We compered our method with the Two-stage detection algorithm; Faster R-CNN, which our model achived better performance

Moreover, we ended the chapter by presentation of the system's user interface, showcasing the test phase and demonstrating how the system operates in real-world scenarios.

General conclusion

In order to improve road traffic safety, this thesis investigated the use of computer vision and image processing algorithms for the identification and measurement of distances between different objects, such as vehicles, animals, and bicycles. This work's larger background is the growing dependence on artificial intelligence (AI) and digital technologies in daily life, especially when it comes to improving driving safety using cutting-edge detection systems. The project specifically sought to create an approximation method and a Convolutional Neural Network (CNN) based on the ResNet-50 architecture for precise distance computation.

Several challenges were encountered during this research. The first challenge was conceptualizing the best implementation idea, which took considerable time and effort. The second challenge involved finding a suitable dataset. Initially, the MSCOCO dataset, with over 80 classes, was considered ideal for object detection. However, its size of 26GB posed a barrier, making it impractical to train on a modest laptop. The third challenge was training the model: with 34 million parameters and a 4GB dataset, training for 20 epochs took over 6 hours, which was resource-intensive.

The solution involved successfully developing a robust CNN model based on ResNet-50 for object detection and a novel algorithm for distance estimation. The system was rigorously evaluated using metrics such as classification accuracy, classification loss, model accuracy, Mean Squared Error (MSE), and Intersection over Union (IoU). The results demonstrated that the system could accurately detect and classify road traffic objects and compute distances with high precision, outperforming some existing models like Faster R-CNN.

The key strengths of this work include:

- Accuracy and Efficiency: The developed CNN model and distance estimation algorithm achieved high accuracy and efficiency.
- Practical Application: The user-friendly interface showcased the system's potential for real-world deployment in enhancing driving safety.
- Comprehensive Evaluation: Extensive testing validated the system’s robustness and reliability.
- However, despite these strengths, there are limitations:
- Real-Time Performance: The system's performance in real-time applications needs further optimization to handle dynamic traffic environments effectively.
- Complex Scenarios: The accuracy of distance estimation in complex scenarios involving multiple objects and varying distances can be further improved.
- Hardware Constraints: The modest hardware used limited the extent of the experiments.

- **Future Perspectives**:

Building on the identified limitations, future work could focus on several enhancements and new research directions:

- Improve the system's performance for real-time applications by optimizing computational efficiency and response times.
- Conduct extensive testing in diverse and dynamic traffic environments to ensure robustness and reliability.
- Implement deep learning techniques for distance estimation to achieve better accuracy, particularly in complex scenarios with multiple objects.
- Integrate additional sensors to enhance detection capabilities.
- Experiment with alternative CNN architectures and advanced optimization techniques to enhance object detection performance, such as hybridizing YOLO and Faster R-CNN.

In conclusion, this research has made significant strides in enhancing road traffic safety through advanced computer vision techniques. The proposed solutions and the extensive evaluation demonstrate the system's potential and set a solid foundation for future improvements and applications. The insights gained and the challenges overcome provide valuable experience and knowledge, paving the way for further advancements in the field.

Bibliography

[1] Yanying Zhang, Xiaolin Zheng. Development of Image Processing Based on Deep, Learning Algorithm. 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC). ieee may 23th 2022.

[2] M Sandeli, « traitement d'images par des approches bio-inspirées application à la segmentation d'images », Constantine 2 university. 2014.

[3] BENFRIHA Sarra and HAMEL Asma. « Segmentation d'image par Coopration rgion contours ». PhD thesis, Kasdi Merbah-Ouargla University, 2016.

[4] Aldjia BOUCETTA. « Etude de l'effet des Transformées de Décorrélation en Compression des Images Couleurs RGB ». PhD thesis, Batna 2 umiversity, 2010.

[5] Nourria Keddar, Leila Ahmed Balkacem. « Détection et Reconnaissance Des Panneaux de signalisation Routière ». Master's thesis. Belhadj Bouchaib d'Aïn Témouchent university, 2019.

[6] Charleen; Cheryl Angelica; Hendrik Purnama; Fredy Purnomo . Impact of Computer Vision With Deep Learning Approach in Medical Imaging Diagnosis. November 24th, 2021. ieee 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)

[7] Anik Datta, Tamara Islam Meghla, Tania Khatun. Road Object Detection in Bangladesh using Faster R-CNN: A Deep Learning Approach. 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON- ECE). 12 April 12th, 2021 ieee.

[8] Yandan Kong, Kai Liu, Zhihong Liang, Tiancheng Liu. Research on small object detection methods based on deep learning. 2022 IEEE 4th International Conference on Power, Intelligent Computing and Systems (ICPICS). September 07th, 2022 ieee.

[9] C. Bhagya, A. Shyna . An Overview of Deep Learning Based Object Detection Techniques. 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT) . June 21th, 2019 ieee.

[10] Ross Girshick. Fast R-CNN. 30 Apr 2015 arXiv:1504.08083

[11] Jun Deng,a, Xiaojing Xuan, Weifeng Wang, Zhao, Hanwen , Zhiqiang,Wang. A review of research on object detection based on deep learning.

[12] Xinyi Zhou; Wei Gong; WenLong Fu; Fengtong Du. Application of deep learning in object detection. 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS). June 29th, 2017 ieee.

[13] Kaiming He Georgia Gkioxari Piotr Dollar Ross Girshick . Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision 2017

[14] https://www.v7labs.com/blog/yolo-object-detection 02/02/2024

[15] By ZHENGXIA ZOU , KEYAN CHEN , ZHENWEI SHI. Object Detection in 20 Years: A Survey. Proceedings of the IEEE March 2023

[16] Dong Wu Manwen Liao Weitian Zhang Xinggang Wang Xiang Bai Wenqing Cheng Wenyu Liu. YOLOP: You Only Look Once for Panoptic Driving Perception. arXiv:2108.11250

[17] Tsung-Yi Lin Priya Goyal Ross Girshick Kaiming He Piotr Dollar. Focal Loss for Dense Object Detection. arXiv:1708.02002

[18] https://www.scirp.org/journal/paperinformation?paperid=115011 27/01/2024

[19] Kaiwen Duan, Song Bai, Lingxi, Honggang, Qingming Huang, Qi Tian.CenterNet. Keypoint Triplets for Object Detection. arXiv:1904.08189

[20] A.Medjaoui , f.fares , « segmentation des images par contours actifs : application sur les Images Satellitaires à Haute Résolutions », Abou Bakr Belkaid university – tlemcen.2012

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks Sun. https://arxiv.org/pdf/1506.01497

[22] Muhammad Abdul Haseeb, Jianyu Guan, Danijela Ristić-Durrant, Axel Gräser. DisNet: A novel method for distance estimation from monocular camera. Institute of Automation, University of Bremen

[23] Mustafa M. Faisal, Mohammad S. Mohammed, Ali M. Abduljabar. Object Detection and Distance Measurement Using AI. 2021 14th International Conference on Developments in eSystems Engineering (DeSE) ieee Mars 2022

[24] K. Karthika; S. Adarsh; K.I. Ramachandran. Distance Estimation of Preceding Vehicle Based on Mono Vision Camera and Artificial Neural Networks. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). Octobre 2020 ieee

[25] Dongsheng Bao; Peikang Wang. Vehicle distance detection based on monocular vision. 2016 International Conference on Progress in Informatics and Computing (PIC), juin 2017 ieee

[26] Tzu-Yun Tseng, Jian-Jiun Ding. Vehicle Distance Estimation Method Based on Monocular Camera. 2020 International Symposium on Computer, Consumer and Control (IS3C), 08 Avril 2021

[27] Djenaihi . «Un système de détection des objets de la circulation routière et d'estimation de leur distance» Master's thesis. Mohamed Khider university - BIKSRA, 2019-2020

[28] KARAM Fouad, IMOULOUDENE. « Transfert sécurisé des données visuelles (images) dans un réseau intranet selon l'architecture client/serveur ». Master' thesis. Abou Bakr Belkaid university - tlemcen .2014-2015

[29] MEDJAOUI Amina, FARES Fadia. « Segmentation des Images par Contours Actifs : Application sur les Images Satellitaires à Haute Résolutions » . Master's thesis . Abou Bakr Belkaid university - Tlemcen, 2012.

[30] https://www.simplilearn.com/image-processing-article 01/02/2024

[31] Shree K. Nayar. Introduction to Computer Vision. Monograph FPCV-0-1, First Principles of Computer Vision, Columbia University, New York, Feb. 2022

[32] https://afrozchakure.medium.com/computer-vision-in-ai-and-its-different-types-%EF%B8%8F-d12ff54f603 06/02/2024

[33] https://www.jeveuxetredatascientist.fr/computer-vision/ 29/01/2024

[34] https://www.run.ai/guides/deep-learning-for-computer-vision#Uses-of-Deep-Learning-in-Computer-Vision 30/01/2024

[35]https://iask.ai/?mode=question&q=+Overview+of+Traffic+Object+Detection%0AImportance+of+Traffic+Object+Detection%0AChallenges+in+Traffic+Object+Detection 09/02/2024

[36] https://medium.com/@sapiensrobot/real-time-object-detection-705c2d26b6f7 10/03/2024

[37] https://utdrive.utdallas.edu/projects/image-based-techniques-in-driving- performance analysis/ 14/03/2024

[38] Asifullah Khan and others « A Survey of the Recent Architectures of Deep Convolutional Neural Networks ». Artificial Intelligence Review DOI, April 21st, 2020.

[39] Houssem Eddine's « Object Detection with OpenCV and Deep Learning». Master's Thesis in Electronics, Networks, and Telecommunications, Mohamed Khider University of Biskra, September 30, 2020.

[40] Oulmi Mehdi and Kaloune Salim's « Object Classification with Deep Learning». Master's Thesis in Computer Science, Akli Mohand Oulhadj University of Bouira, 2018.

[41] Louam Abdelhak Bilal. «Deep Learning based on reduction methods for face recognition» Master's Thesis in Telecommunication Sciences and Technologies, Networks and Telecommunication, Mohamed Khaider University of Biskra, 2019.

[42] https://www.python.org/doc/essays/blurb/ Accessed.03.2023 05/05/2024

[43] https://opencv.org/ 03/05/2024

[44] https://numpy.org/doc/stable/ 06/05/2024

[45] https://flask.palletsprojects.com/en/3.0.x/ 06/05/2024

[46] https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/ 10/03/2024

[47] Mesbah Fethia. « Détection d'objets par Deep Neural Network à l'aide du modèle YOLO en temps réel ». Master's Thesis in Telecommunication Sciences and Technologies, Networks and Telecommunication, 8 Mai 1945 University of Guelma.

[49] Mark Everingham · S. M. Ali Eslami · Luc Van Gool · Christopher K. I. Williams · John Winn · Andrew Zisserman. The PASCAL Visual Object Classes Challenge: A Retrospective. Springer Science +Business Media New York 2014. 25 June 2014.

[50] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6679565/ 30/05/2024

[51] https://www.v7labs.com/blog/intersection-over-union-guide 06/06/2024

[52] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.
https://doi.org/10.48550/arXiv.1512.03385

[53] Chollet, F. and Team, K. (2021). Keras documentation: Optimizers. https://keras.io/api/optimizers. Accessed on May 7th, 2023.

[54] https://github.com/matplotlib/matplotlib 10/06/2024

[55] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Advances in Neural Information Processing Systems (pp. 91-99).

مـلـخص

في السنوات الأخيرة، دفع تطور مجال رؤية الحاسوب صناعة السيارات إلى إنشاء أنظمة مساعدة للسائق يمكن أن تقلل بشكل كبير من حوادث الطرق. في هذه الأطروحة، قمنا بتقديم نموذج لكشف أجسام حركة المرور على الطريق وتقدير مسافتها.

يقوم هذا النموذج بتحديد واعتراف بالأشياء (مثل السيارات، الشاحنات، والمشاة) المتواجدة أمام المركبة. وذلك لتحديد طبيعة كل كائن بدقة باستخدام خوارزميات التعلم العميق. بعد ذلك، يقوم النموذج بتقدير المسافة بين الكاميرا والكائن المكتشف وبين أزواج الكائنات. يتيح دمج هذه المعلومات المختلفة للسائقين التركيز بشكل أفضل على الطريق وتحقيق تصور أفضل للعالم من حولهم.

يستخدم النظام شبكة عصبية تكرارية (CNN) استنادًا إلى هندسةResNet-50 ، معززة بطبقات تكرارية إضافية لتعزيز أداء الكشف. تم تدريب النموذج باستخدام مجموعة بياناتPascal VOC 2012 ، حيث حقق دقة بنسبة 95.21% وخسارة تصنيف دنيا باستخدام محسن Nadam .قمنا بمقارنة نموذجنا مع نموذج Faster R-CNN ، وحقق نموذجنا أداءً أفضل.

الكلمات الرئيسية: التعلم العميق، تقدير المسافة، رؤية الحاسوب، كشف الأجسام، ،ResNet-50

Abstract

In recent years, advancements in computer vision have driven the automotive industry to develop driver assistance systems capable of significantly reducing road accidents. This thesis introduces a model designed for detecting road traffic objects and accurately estimating their distances.

The proposed model identifies and classifies objects such as cars, trucks, and pedestrians located in front of the vehicle. Deep learning algorithms are employed to precisely determine the nature of each object. Subsequently, the model estimates both the distance between the camera and detected objects, as well as between pairs of objects. Integrating these insights enables drivers to maintain better focus on the road and gain a clearer perception of their surroundings.

The system utilizes a Convolutional Neural Network (CNN) based on the ResNet-50 architecture, augmented with additional convolutional layers to enhance detection performance. Training was conducted on the Pascal VOC 2012 dataset, achieving an accuracy of 95.21% and minimal classification loss using the Nadam optimizer. We compared our model with the Faster R-CNN model, and ours achieved better performance.

Keywords: Computer vision, Deep Learning, Distance estimation, Object detection, ResNet-50.