PEOPLE'S DECMOCRATIC REPUBLIC OF ALGERIA
Ministry of higher Education and Scientific Research
University Mohamed Khider – BISKRA
Faculty of Exact Sciences, Natural and Life Sciences
Computer science department

Order N° : …./M2/2023

Dissertation

Presented to obtain the academic master's degree in

Computer Science

Option: system information optimization and decision **(SIOD)**

Identifying and tracking vehicles from a front facing camera using deep learning

---

By :
REZIG LOKMANE

CHIGHOUB Fouzia	MAA	President
BABAHENINI Mohamed Chaouki	PR	Supervisor
NAIDJI Ilyes	MCB	Examiner

Academic year 2022-2023

Thanks

First of all, I want to express my gratitude to Allah Almighty. Without his support, he It would have been very difficult for me to complete this research on my own.

Second, I would like to thank my parents for believing in me and supporting me in all difficult situations, especially my mother, And I hope that she sees what I have achieved now.

Then, I would like to thank my supervisor, Babahenini Mohamed Chaouki, for guiding me in the right direction throughout the project.

Finally, I would like to thank all the other members of the department, my colleagues, and some of my friends. They always kept me motivated and helped me by giving me mental support and strength to finish my thesis.

abstract

The identification and tracking of vehicles play a crucial role in various applications, including traffic management, surveillance systems, and autonomous driving. Traditional methods for vehicle detection and tracking often rely on handcrafted features and heuristics, which are limited in their ability to handle complex and diverse real-world scenarios. In recent years, deep learning approaches have emerged as a promising solution to overcome these limitations, leveraging the power of neural networks to learn complex representations directly from raw data.

This research focuses on the development of an efficient and accurate system for identifying and tracking vehicles from a front-facing camera using deep learning techniques. The proposed system consists of two main components: vehicle detection and vehicle tracking.

For vehicle detection, a deep convolutional neural network (CNN) is trained on a large dataset of annotated vehicle images to learn discriminative features. The trained CNN is capable of accurately localizing and classifying vehicles in real-time from the input video frames. This enables efficient detection of multiple vehicles in various environmental conditions, including challenging scenarios such as occlusion and varying lighting conditions.

For vehicle tracking, a combination of object tracking algorithms and deep learning-based techniques is employed. The system maintains a set of vehicle tracks over consecutive frames, utilizing methods such as Kalman filtering, optical flow, and deep feature matching. Deep learning-based methods aid in handling challenging situations like abrupt motion changes, occlusion, and temporary object disappearance.

The proposed system is evaluated on a benchmark dataset and compared against state-of-the-art methods. The results demonstrate the effectiveness of the deep learning-based approach in accurately identifying and tracking vehicles from a front-facing camera. The system achieves high detection and tracking accuracy, robustness to challenging scenarios, and real-time performance.

In conclusion, this research presents an effective solution for identifying and tracking vehicles from a front-facing camera using deep learning techniques. The proposed system offers improved accuracy and robustness compared to traditional methods, enabling its po-

tential applications in traffic management, surveillance systems, and autonomous driving, ultimately contributing to safer and more efficient transportation systems.

July 9, 2023

List of Figures

Contents

Chapter 1

General introduction

Tracking an object is the process of continuously keeping track of its location and movement. It entails using a variety of methods, such as computer vision, sensor technologies, or GPS, to continually estimate the position, motion, and orientation of the vehicle or vehicles being tracked[10].

Vehicle tracking can be used in a variety of contexts, including security systems, self-driving cars, robotics, augmented reality, and sports analysis. Accurate vehicle tracking enables the extraction of useful data, the generation of forecasts, the detection of abnormalities, and the facilitation of interactive experiences[19]. The major vehicle of developing tracking vehicle software is to offer a reliable and effective method of keeping track of items, enabling a variety of uses in industries like security, transportation, entertainment, and research.

1.1 Problem statement

A common problem with vehicle tracking is the accuracy of vehicle detection and tracking. This could be due to lighting variations, similar vehicles in the scene, temporary occlusions, or rapidly moving vehicles, and could also be the AI tools that we are using (CNN, YOLO, etc). Also is try to create applications that allow us to define and help people when they drive we had a lot of problems some of them try the application in real life one the problem that we had is that we get problems with the authority that didn't let as to try

the application in the public road etc, Accurate detection and tracking are essential to ensure reliable results and efficient use of the tracking vehicle in various applications.

1.2 Proposed method

We are going through some of the primary methods used in tracking vehicles(both of these methods are based on AI):

- **ByteTrack**

  Modern vehicle tracking technology known as ByteTrack combines the benefits of anchor-based and anchor-free methods.

  By using a cutting-edge technology that links each detection box with a related vehicle, the ByteTrack algorithm circumvents the drawbacks of conventional tracking techniques. It utilizes a two-head architecture, where one head forecasts the bounding boxes, and the second head forecasts keypoint heatmaps. ByteTrack is able to precisely locate and track things because of this design, which also records fine-grained spatial data[1].

This is done by comparing the detections to the tracks that already exist based on a variety of criteria, including motion data, spatial overlap, and appearance resemblance. Real-time tracking applications are now possible in many different domains, such as surveillance, autonomous driving, and human-computer interaction, thanks to ByteTrack ability to follow multiple vehicles across successive frames with effectiveness.

1.3 Description of the dissertation

We will discuss tracking vehicles generally and go over the available options. The study focuses on using deep learning to create a reliable and effective vehicle tracking system. This study tries to address difficulties caused by occlusions, scale changes frequently seen in vehicle tracking, by utilizing the capabilities of CNNs. The dissertation offers a thorough analysis of many tracking-related topics, such as feature extraction, network architecture

design, data preparation, and tracking methods. The proposed CNN-based tracking system's performance, robustness, and scalability are evaluated experimentally using a variety of datasets. The findings of this study have implications for a variety of applications, including surveillance, traffic control, and intelligent transportation systems.

Chapter 2

Tracking vehicles based on AI

2.1 Introduction of vehicle Tracking

Tracking vehicles offers numerous benefits across various domains. In fleet management, it enables efficient monitoring and optimization of driving safely, including route planning.

Moreover, vehicle tracking systems enhance vehicle security, and Personal vehicle tracking allows individuals to monitor their vehicles for various purposes like other drivers on road, such as ensuring the safety of family members.

In numerous industries and applications, vehicle tracking is essential for gaining insightful information, allowing automation, and improving user experience. Here are a few crucial features and uses of vehicle tracking :

- **Autonomous cars**: In order for autonomous cars to perceive and comprehend their immediate environment, vehicle tracking is a crucial component. In order to provide safe navigation and collision avoidance, it enables vehicles to recognize and track other vehicles, pedestrians, cyclists, and obstacles[4].

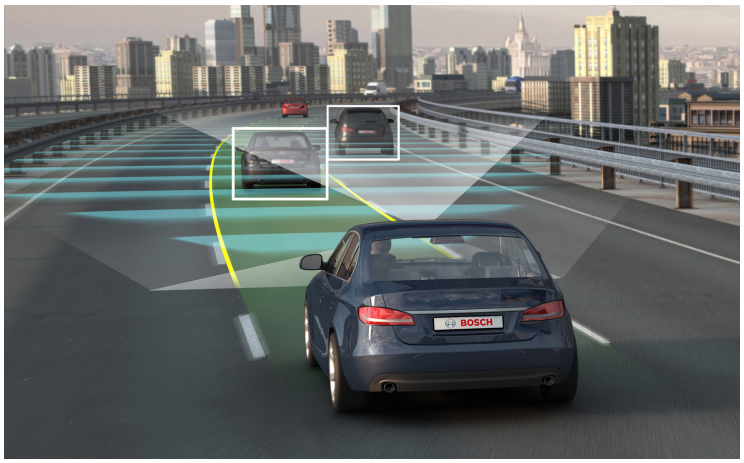

Figure 2.1: Autonomous cars

- **Sports analysis**: Tracking players, balls, and other objects during games is a common application of vehicle tracking in sports analysis. It makes it possible for coaches, broadcasters, and sports fans to generate visualizations and statistics as well as track players, predict their trajectories, and analyze their performance in depth[26].

Figure 2.2: Sports analysis

but in the dissertation, we are going to focus on the tracking of the vehicle and how it works in the pov of driving cars.

Types of Tracking The two different kinds of vehicle tracking are video tracking and picture tracking :

- **Image tracking**: The automatic identification and tracking of images is known as image tracking. Its main use is in augmented reality (AR), where an algorithm might examine a two-dimensional image taken by a camera to identify planar images. The three-dimensional graphic is then placed on top of these planar images to produce an augmented reality experience[22].

Users are able to move their cameras while still retaining the tracking of the 2D surface and the graphic shown on it by superimposing the 3D graphic onto the detected planar surface. Companies like Apple and Ikea use this technology to provide customers with a virtual experience so they can see how things might look in their own homes.

- **Video Tracking**: The process of video tracking is finding a link or correlation between the target items seen in successive video frames. Its goal is to examine the successive frames of a video and, using prediction and bounding box delineation, relate the past and current positions of a vehicle. This method is widely used in traffic monitoring, self-driving cars, and security systems because it permits in-the-moment video analysis[22].

2.2 Machine learning for tracking vehicles

A branch of artificial intelligence called machine learning is concerned with creating algorithms that can learn from data and make predictions or decisions based on that data. Without the need for explicit programming, machine learning aims to automate the process of creating models that can make predictions based on incoming data[11], In literature, Tom Mitchell defined ML as "A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in T, as measured by P, improves with experience E" [28]. The majority of technologies now incorporate machine learning, including self-driving cars, recommendation engines that propose movies and other TV shows to users, facial recognition technology for tagging individuals on social media, and OCR, which turns text images into editable forms. As a result, machine learning is a science that is continually evolving and growing, and there are some known problems that need to be solved.

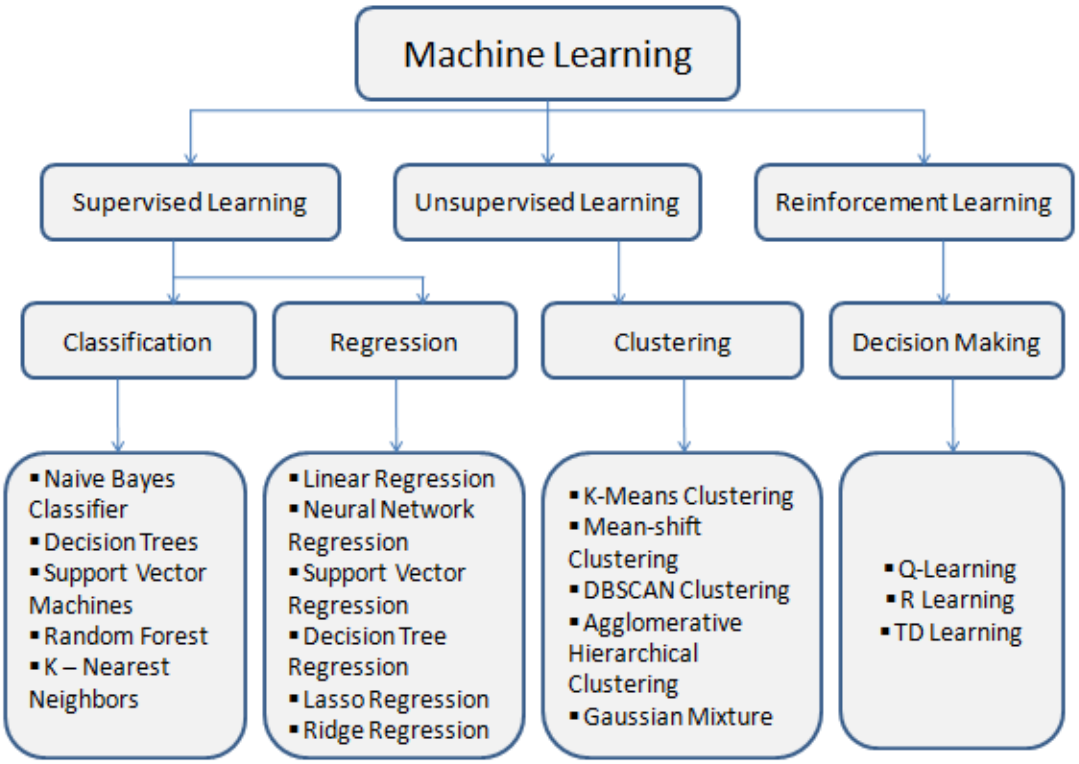

Figure 2.3: machine learning

2.2.1 Supervised learning

The most prevalent kind of machine learning is supervised learning. In supervised learning, the goal is to teach the computer a mapping function (also known as a model) that can correctly predict the output given a new input. The machine is fed labeled training data (i.e., input/output pairs). Email spam filtering, audio recognition, and image categorization are a few examples of supervised learning tasks, A machine learning technique called supervised learning uses labeled data, which has been annotated with the appropriate output values for the inputs. In order for the algorithm to properly predict new, unplanned inputs, supervised learning aims to develop a mapping from inputs to outputs [6]. Labeled data is divided into a training set and a test set for supervised learning. The model is developed using the training set, and its performance is subsequently tested using the test set. Up until it can properly predict the test set, the model is iteratively improved [14]. Predictive modeling, speech recognition, natural language processing, image classification, and other

fields all make major use of supervised learning. Decision trees, logistic regression, neural networks, and linear regression are a few examples of supervised learning algorithms.

2.2.2 Unsupervised learning

Unsupervised learning, in opposed to supervised learning, focuses on input that has not been labeled Xtrain = Xu = x1,..., xu and does not know in advance what the result will be. The primary duty of the machine learner is to resolve issues on its own[15]. This is comparable to giving a pupil a set of patterns and asking them to pinpoint the underlying themes that gave rise to the patterns. This translates to machine learning independently without human guidance[11].

2.2.3 Semi-supervised Learning

For applications where there is a limited amount of labeled data, a hybrid strategy combining supervised and unsupervised learning can be used. Using labeled and unlabeled data inputs, this strategy seeks to produce a mapping function. The semi-supervised learning system makes an effort to categorize part of the unlabeled data using the labeled data. To fully make use of the advantages of semi-supervised learning, the unlabeled dataset must be substantially larger than the labeled data. Otherwise, fixing the issue might be better accomplished using conventional supervised methods.

Applications in the real world that benefit from semi-supervised learning include the classification of protein sequences, the categorization of online material, and speech analysis, where the labor-intensive and human-intensive process of categorizing audio recordings takes place[12].

Semi-supervised learning may be classified into the following categories [44] :

- A type of semi-supervised learning called self-training involves the model learning from its own predictions.

- Using multi-view data and the co-training framework, co-training is a subset of semi-supervised learning that enables models to gain knowledge from their own predictions.

- learning involves the student actively choosing which data points to ask a subject matter expert or instructor to label, It is a form of semi-supervised learning.

2.3 Deep learning for tracking vehicles

Deep learning is a branch of machine learning that focuses on using artificial neural network techniques that are motivated by the composition and operation of the human brain. It includes a range of learning strategies that include training models using datasets, including unsupervised, semi-supervised, and supervised learning[30].

2.3.1 Terminology

biological neuron : Over 100 billion neurons make up the enormous network that makes up the human brain, and each neuron in this network receives information from other neurons. As a result of their interaction and convergence, these signals might cancel out unwanted or excessive signals before they spread further.

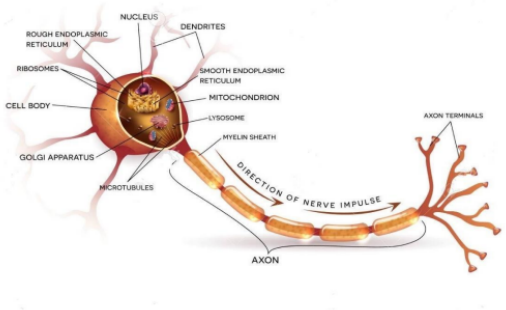

Figure 2.4: Neuron biologic

According to Figure (2.4), dendrites, cell bodies, and axons are the three primary parts of a neuron. A network of receptors known as dendrites is responsible for receiving electrical impulses from neighboring neurons and transmitting them to the cell body. The dendrites assemble charges and function as integrators. When the total charge reaches a specific level, the neuron creates an electrical potential through an electrochemical mechanism. This electrical potential then spreads along the neuron's axon and activates nearby neurons.

Artificial neuron: Biological neurons inspired artificial neurons, defined the latter as a function that learns to manipulate characteristics on which it will apply a function, and will bring out a new value that will spread to other neurons.

In following the figure (Figure 2.5) the explanation of the components of an artificial neuron and we explain their operation :

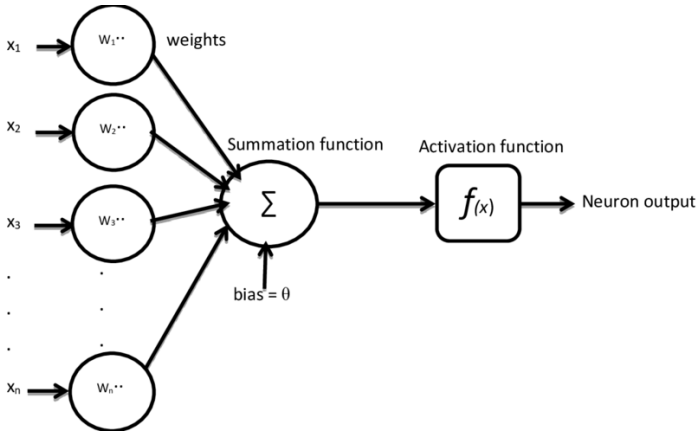

Figure 2.5: artificial neuron[41]

We have the values entered (x i) are the characteristics, and the combination function calculates the weighted sum of weights (w i) and these entered The result n of the weighted sum is called the activation level of the neuron, When the level d activation reaches or exceeds the threshold, then the argument of becomes positive (or null). Otherwise, it is negative[37].

The weights: The weights are the coefficients of the equation you are trying to solve. Negative weights reduce the value of an output. When a neural network is trained on the training set, it is initialized with a set of weights. These weights are then optimized during the learning period and the optimal weights are produced[14]. The bias is simply a constant value (or a constant vector) that is added to the product of the inputs and the weights. The bias is used to compensate for the result[16].

2.3.2 Convolutional Neural Networks :

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm that is particularly well-suited for image processing and analysis tasks. In this response, we will

explain how CNNs work in more detail, At a high level, a CNN is a type of neural network that consists of multiple convolutional layers, followed by one or more fully connected layers. Each convolutional layer is made up of a set of filters, which are applied to the input image in a sliding window fashion.

The output of the convolution operation is a feature map, which contains information about the presence of certain visual features in the input image[4], The convolution operation is designed to identify patterns and structures in the input image, regardless of their location in the image.

This is achieved by sharing weights between the different locations in the image, which reduces the number of parameters that need to be learned and makes the network more efficient[23], After passing through the convolutional layers, the output of the network is typically passed through one or more fully connected layers, which produce the final prediction or decision based on the input image and the learned weights.

The weights of the network are typically learned using backpropagation, as in other types of neural networks[24], One of the key advantages of CNNs is their ability to learn and identify complex visual features in an input image. This makes them well-suited for tasks such as vehicle recognition, image classification, and image segmentation. CNNs have been successfully applied in many domains, including healthcare, transportation, and security[8].

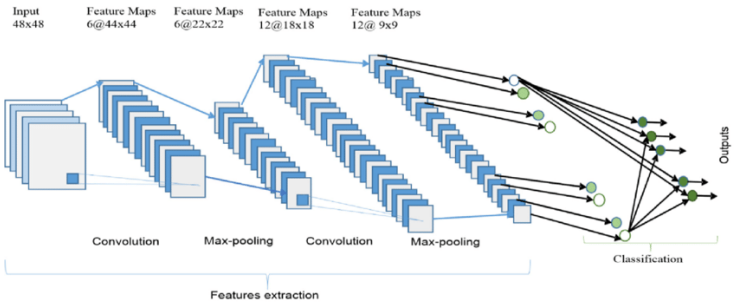

Figure 2.6: architecture of the Convolutional Neural Network

Steps of tracking baesd on cnn vehicle tracking utilizing Convolutional Neural Networks (CNNs) uses the capabilities of deep learning to detect and track vehicles in video sequences. CNNs are particularly ideal for this purpose since they can successfully

learn detailed spatial patterns and characteristics straight from the raw picture input[35]. The method of vehicle tracking using CNNs generally entails the following phases:

- **Detection**: Initially, a CNN-based vehicle identification method, such as Single Shot MultiBox Detector (SSD) or You Only Look Once (YOLO), is applied to recognize items of interest in the first frame of the video series. These methods produce bounding box predictions and class probabilities for the observed [25].

- **Feature Extraction**: Once the vehicles are spotted, a pre-trained CNN model, such as VGGNet, ResNet, or MobileNet, is used to extract high-level features from the regions of interest (ROI) inside the observed bounding boxes. The CNN model examines the ROI and outputs a feature vector that encapsulates the vehicle's visual appearance[39].

- **Matching and Tracking**: The retrieved features are then utilized to match and track the item in consecutive frames. This may be performed using numerous strategies, such as correlation filters, optical flow, or deep metric learning. The aim is to select the most comparable item in each frame based on the learned attributes and preserve the continuity of the vehicle's motion[25].

- **Update and Adaptation**:To handle variations in appearance, scale, or occlusion, the CNN-based tracker may incorporate mechanisms for online model update and adaptation. This involves continuously updating the CNN model using new examples of the tracked vehicle and fine-tuning its parameters to improve tracking performance.

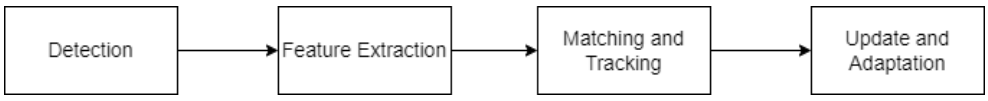

Figure 2.7: steps of tracking based on cnn

2.3.3 Issues and Solutions in CNN-based vehicle Tracking :

The primary difficulties typically result from flaws in the image that prevent vehicle tracking models from efficiently performing detections on the images. Here, we'll go over the few most typical problems that come up when tracking vehicles and how to avoid or resolve them[27].

- **Occlusion**: Occlusion has many Definition but it is the same in deep learning, Occlusion happen when two vehicles or more get closer to each other, that case problem for the algorithm to track vehicles in the same time because the occluded vehicle can be seen as one, the system can get confused for identify the initially tracked vehicle as a new vehicle[9].

Figure 2.8: ome situations of occluded

- **Object Recognition and Localization** Vehicle classification and localization The first significant difficulty with vehicle detection is its additional goal: in addition to classifying picture vehicles, we also want to establish the locations of the items, or what is known as the "vehicle localization task." Most often, a multi-task loss function is used by researchers to solve this problem, penalizing both classification and localization failures[17].

  One well-liked category of vehicle detection frameworks is regional CNNs. In these techniques, region recommendations for where items are most likely to be placed are created, and then CNN processing is used to categorize and further specify vehicle placements. To enhance the performance of R-CNN, Ross Girshick, and colleagues created Fast R-CNN. Because the classification and localization tasks are optimized

using a single unified multi-task loss function, Fast R-CNN not only dramatically speeds up computation times but also enhances accuracy. Each potential vehicle-containing candidate area is compared to the actual items in the picture. Penalties are then assessed to candidate areas for incorrect classifications and incorrect box alignment. Thus, there are two different types of terms in the loss function[36]: where the classification term imposes log loss on the projected probability of the true

$$\mathcal{L}(p, u, t^u, v) = \overbrace{\mathcal{L}_c(p, u)}^{classification} + \lambda \overbrace{[u \geq 1]\, \mathcal{L}_l(t^u, v)}^{localization},$$

Figure 2.9: loss function

vehicle class u and the localization term is a smooth L loss for the four positional components that compose the rectangle. Note that the localization penalty does not apply to the background class when no vehicle is present, (u=0). Also notice that the value (λ) may be modified to emphasize either classification or localization more strongly.

- **anchor box**: Instead of selective search, Faster R-CNN's enhanced region proposal network employs a narrow sliding window over the image's convolutional feature map to create candidate RoIs. Multiple RoIs may be expected at each place and are specified relative to reference anchor boxes. The shapes and sizes of these anchor boxes are deliberately designed to cover a variety of various scales and aspect ratios. This permits numerous sorts of vehicles to be recognized with the expectation that the bounding box coordinates need not be altered significantly throughout the localization operation. Other frameworks, notably single-shot detectors, also employ anchor boxes to initialize areas of interest[43].

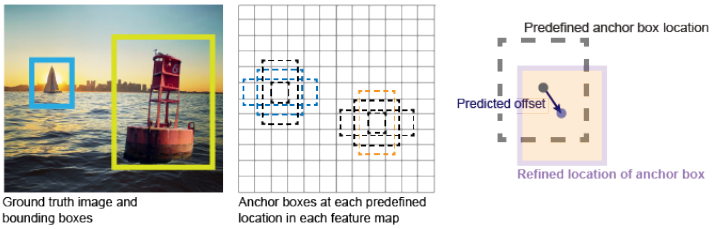

Figure 2.10: anchor box

- **The feature pyramid network**: The feature pyramid network (FPN) takes the notion of numerous feature maps one step further. Images initially transit via the standard CNN route, generating semantically rich final layers. Then to recover improved resolution, FPN develops a top-down route by upsampling this feature map. While the top-down route helps identify items of varied sizes, spatial placements may be biased. Lateral connections are created between the original feature maps and the appropriate reconstructed layers to enhance item localization. FPN now offers one of the leading approaches to identify vehicles at many scales, and YOLO was upgraded with this technology in its 3rd iteration[43].

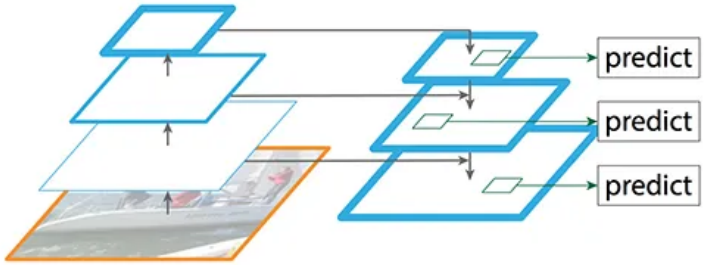

Figure 2.11: The feature pyramid network architect

2.4 Vehicles tracking techniques classification

2.4.1 Motivation

Convolutional Neural Networks (CNN) are employed because of their excellent capacity for processing and evaluating visual input. CNNs are exceptionally excellent at tasks like image classification, object recognition, and semantic segmentation because they are specifically created to identify difficult patterns and extract meaningful information from

photos. They automatically discover and represent visual information thanks of their hierarchical structure and ability to learn from large-scale datasets, **which increases accuracy and efficiency in computer vision applications.**

Vehicles tracking techniques classification :

Based on the technology and methods used to monitor and track vehicles, vehicle tracking technologies can be divided into a number of categories. These methods provide accurate and immediate vehicle location information by combining hardware, software, and communication technologies the figure(Figure 2.12) shows the different classifications.

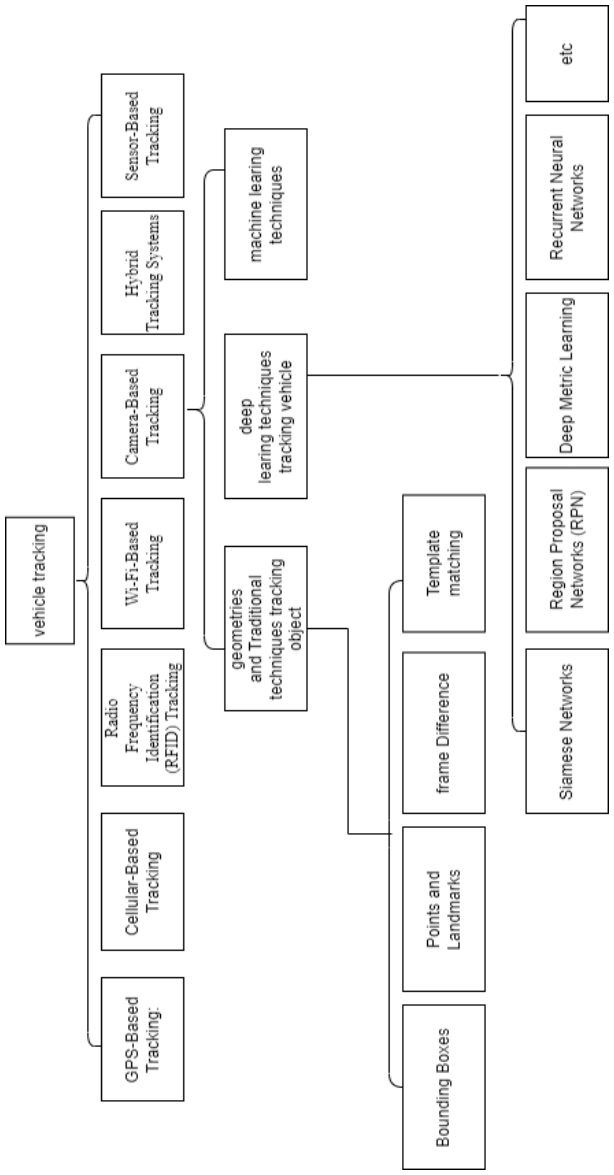

Figure 2.12: the different classification of tracking vehicles

Depending on the technology and methods employed, there are various subcategories of vehicle tracking techniques. Here are a few categories that are frequently used:

- **GPS**Vehicles are tracked using Global Positioning System (GPS) technology. GPS receivers fitted in vehicles receive signals from satellites to pinpoint their exact location. A central server receives the location information for tracking and analysis[38].

- **Cellular**Vehicle tracking is done through mobile networks. Vehicles come with cellular modems or other gadgets that connect to mobile towers. Cellular networks are used to transmit the location data to a central server[3].

- **Radio Frequency Identification (RFID)**Vehicles have RFID tags affixed to them, and RFID readers are scattered throughout the route being tracked An RFID reader detects and records each vehicle's specific identification number as it passes by. The sequence of RFID tag readings is used to track the movement of automobiles[31].

- **Camera**To visually track automobiles, surveillance systems or video cameras are used. To recognize and track automobiles, computer vision algorithms examine the video data that was taken. Identification of vehicles can be done using license plate and color and other detail recognition or other visual cues[18].

- **Sensor**Vehicle tracking systems employ a variety of sensors, including magnetometers, accelerometers, and gyroscopes. To determine the vehicle's position, speed, and direction of motion, sensor data is gathered and analyzed. For more precise tracking, sensor fusion techniques may be used to merge data from several sensors.

These divisions offer a summary of several vehicle tracking methods. The choice of a particular tracking technique is influenced by several elements, including the needed precision, coverage area, cost, and application requirements.

2.4.2 Geometries and Tradition techniques

There are various geometric models that may be used to model the shape, position, and velocity of an item when it comes to tracking. The strategies and systems used in vehicle

tracking heavily rely on these geometric representations. Here are a few typical vehicle tracking geometries :

- **Bounding Boxes :**is enclosed by rectangular sections. The top-left and bottom-right corners' coordinates serve as their definition. In situations when an item has a roughly rectangular form, bounding boxes offer a quick and effective way to represent the vehicle[29].

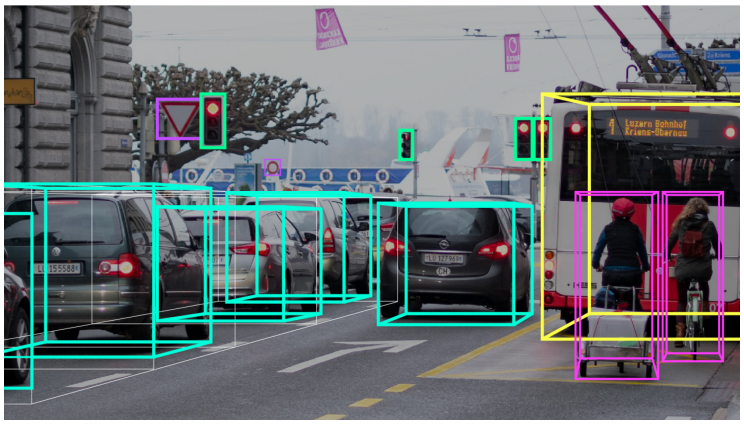

Figure 2.13: Bounding Boxes

- **Points and Landmarks :**Points and Landmarks are certain areas of the item or characteristics that are tracked throughout time. The corners or other distinguishing features of the item may be represented by these points. Point-based tracking algorithms frequently estimate the mobility of the points using methods like feature matching or optical flow[29]. here example of Points and Landmarks in the face of the person

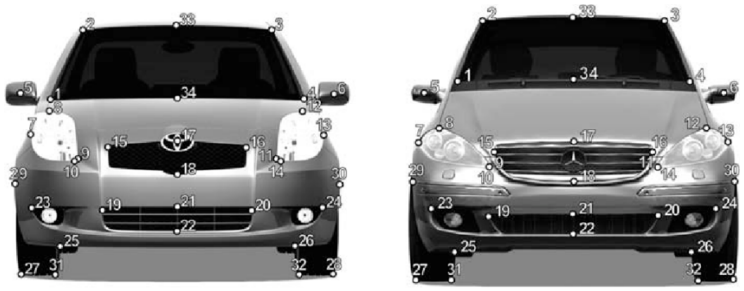

Figure 2.14: Points and Landmarks

- **Contours**: the borders or contours of the thing. They may be retrieved using edge detection or segmentation methods and serve to depict the form of the vehicle. The main goal of contour-based tracking techniques is to follow a vehicle's motion by matching and aligning the contours across frames[29].

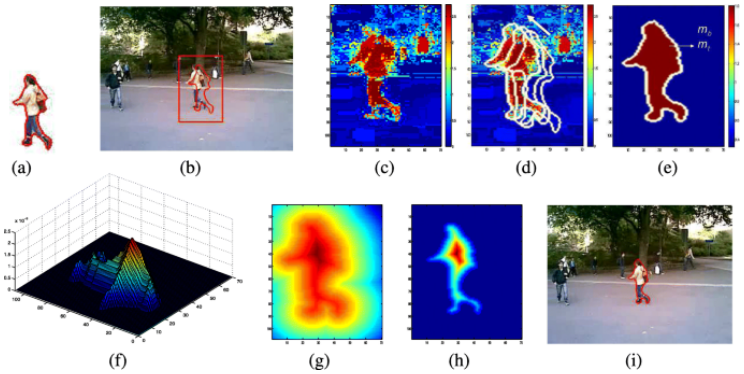

Figure 2.15: example of Contours

there are also some techniques do not called geometries but Traditional like

there a lot of vehicle tracking techniques (Traditional) and we go throw some of them

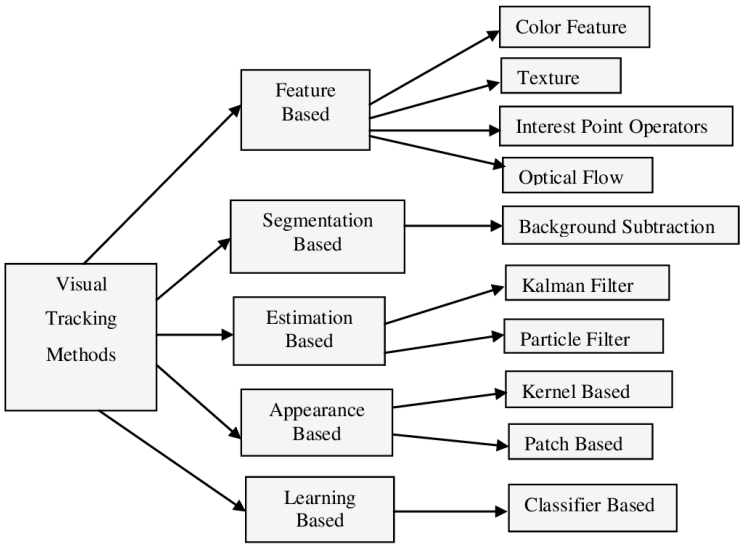

Figure 2.16: tree of Traditional techniques of tracking vehicle

- **Optical Flow :** this method takes part in analyzing the motion of pixel frame between successive frames in a video (could be live stream) sequence by examining the amplitude and direction of pixel movement, it can be easy to track and focus on the movement of cars in the scene[33].

Figure 2.17: Optical Flow

- **Background Subtraction :**In this approach, the foreground items in a video sequence(like a car) are separated from the environment. It is simple to identify moving vehicles and follow them throughout time by contrasting the current frame with a backdrop model[32].

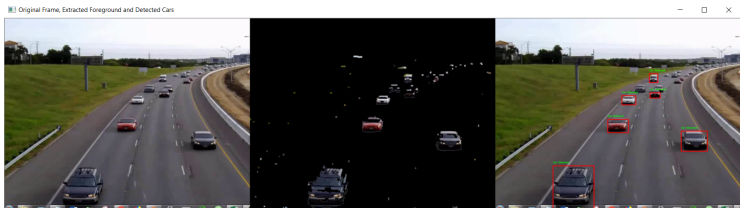

Figure 2.18: Background Subtraction

- **Frame Difference :** the technique is used for following the vehicle in the image we go and find the difference between two consecutive frames The method uses a reference background image for comparison purposes[32][34], Changes are detected in the areas and are identified as moving items.

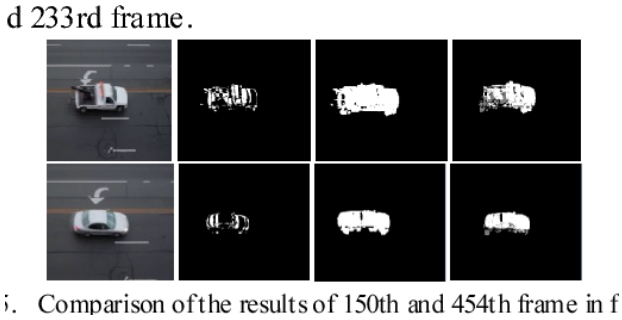

Figure 2.19: Frame Difference

- **Template matching :** Template matching is the process of moving the template over the entire image and calculating the similarity between the template and the covered window on the image. Template matching is implemented through two-dimensional convolution. In convolution, the value of an output pixel is computed by multiplying elements of two matrices and summing the results, One of these matrices represents the image itself, while the other matrix is the template, which is known as a convolution kernel [5].

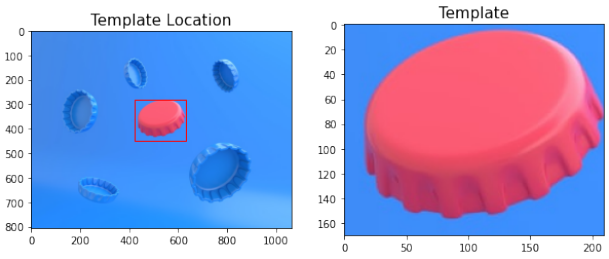

Figure 2.20: Template matching

- **Kalman :**For state estimation and prediction in many applications, including vehicle tracking, Kalman filtering is a frequently used method. An ideal recursive filter is used to estimate a system's state by fusing predictions from a mathematical model with data from sensors. When there is ambiguity or noise in the measurements, the Kalman filter is especially helpful.

  The two basic phases of the Kalman filter's operation are prediction and update. In the prediction phase, the filter makes a prediction about the system's state at the upcoming time step based on the dynamics model of the system. The error covariance, a measure of the uncertainty in the projected state, is provided along with this prediction[20].

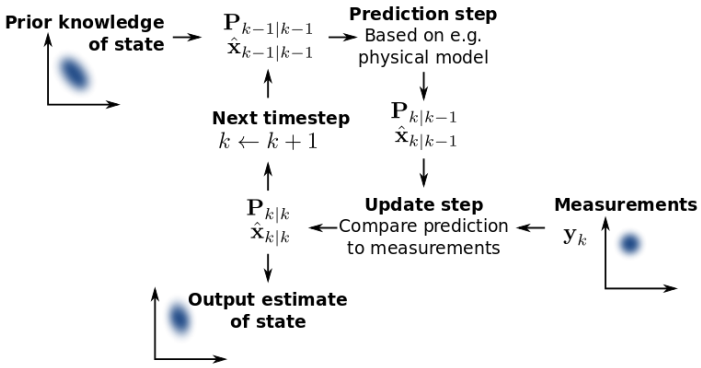

Figure 2.21: Basic concept of Kalman filtering

2.4.3 Tracking using SVM

Vehicle tracking systems employ Support Vector Machines (SVM), a well-liked machine learning technology. SVMs work well for binary classification jobs where the objective is to divide data points into two different classes. SVMs may be used to determine if an object

in a video frame belongs to a vehicle or not when it comes to tracking moving objects. A decision boundary that effectively distinguishes the vehicle class from the non-vehicle class is learned by the SVM method using a collection of characteristics taken from the input data[42].

Sliding window methodology is one typical method for tracking vehicles using SVM. The video frames are scanned by the sliding window at various sizes and locations to extract picture patches, which are then categorized by the SVM model. Each patch's presence or absence of a vehicle is determined by the SVM.[21].

2.4.4 Techniques based on CNN

There are numerous ways for tracking vehicle-based on Convolutional Neural Networks (CNNs) that have been developed in the area of computer vision. Here are some regularly used techniques:

- **Siamese Networks**: Siamese networks are a prominent method for optical vehicle tracking. They consist of two identical CNN branches that share weights and are trained to learn a similarity measure between the target vehicle and candidate areas in future frames. By comparing feature representations, the tracker may select the most comparable area to the target[2].

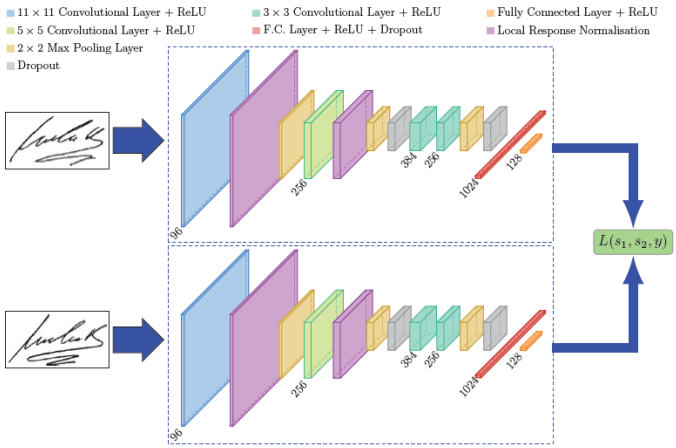

Figure 2.22: Siamese Networks

- **Region Proposal Networks (RPN)**: RPN is a method often employed in vehicle

detection. It creates a collection of possible vehicle areas in each frame based on CNN-based region recommendations. The tracker then finds the most relevant locations that match to the monitored item[36] as we see in this figure(2.23).

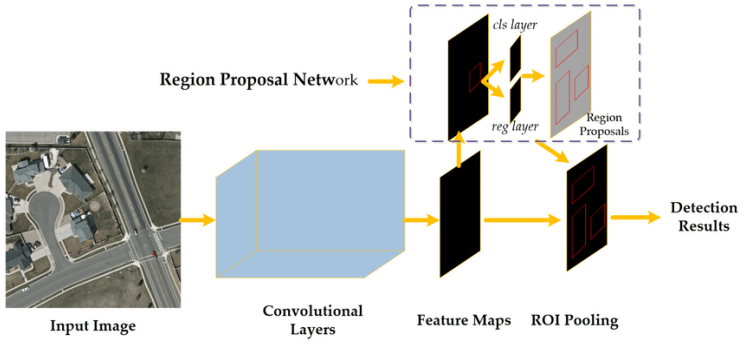

Figure 2.23: Region Proposal Networks (RPN)[7]

- **Deep Metric Learning**: Deep metric learning approaches seek to develop similarity metrics that can efficiently evaluate the similarity of feature representations of distinct frames. By training a CNN to integrate frame information into a metric space figure(2.24), the tracker can conduct robust matching and tracking based on similarity scores[40].

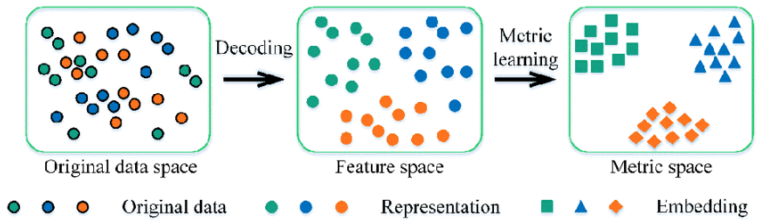

Figure 2.24: Deep Metric Learning

- **Recurrent Neural Networks**: In order to capture temporal connections between frames, RNNs, such as Long Short-Term Memory (LSTM) networks, can be employed in vehicle tracking. The tracker can learn to anticipate the vehicle's location using data from earlier frames by integrating recurrent connections[40].

2.5 Conclusion

In conclusion, tracking vehicles using AI-based techniques has revolutionized the field of transportation and surveillance. By harnessing the power of computer vision and deep learning algorithms, AI enables accurate and efficient tracking of vehicles in diverse scenarios. These techniques offer numerous benefits, including enhanced safety, improved traffic management, and effective resource allocation.

AI-based vehicle tracking systems provide real-time insights, anomaly detection, and predictive capabilities, enabling smarter decision-making and proactive measures. As AI technology advances further, we can anticipate even more sophisticated and intelligent vehicle tracking solutions that will continue to optimize transportation systems and contribute to safer and more efficient roads.

Chapter 3

The Design of software pipeline to identify vehicles in a video from a front-facing camera on a car

3.1 Introduction

The construction of a software pipeline to recognize automobiles in a video using Convolutional Neural Networks (CNN) requires many important components. Firstly, a collection of tagged vehicle photos is gathered and utilized to train a CNN model specialized for vehicle detection. The CNN model learns to extract important information and categorize whether an area of the picture includes a vehicle or not.

Overall, the software pipeline harnesses the capabilities of CNNs to recognize automobiles in the video by learning discriminative characteristics and conducting correct classification, resulting in robust and efficient vehicle recognition.

3.2 General architecture

Our identification model incorporates a varied variety of photos as inputs, including examples gathered from the vehicle and non-vehicle dataset. This strategy allows the model to effectively detect numerous kinds of vehicles. In the next part, we present a full overview

of our concept, which is specially intended to encourage safe driving on the road.

The model not only recognizes automobiles present in the frames but also identifies the lines that exist in the visual scene. By including these features, our model intends to increase overall road safety and provide a more secure driving experience.

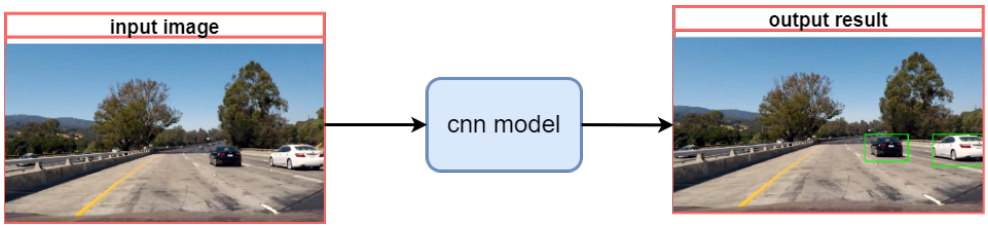

Figure 3.1: General architecture

3.3 Architectural details

In order to build a deep learning model for tracking vehicles with CNN that can classify accurately, we propose a CNN shown, in Fig.

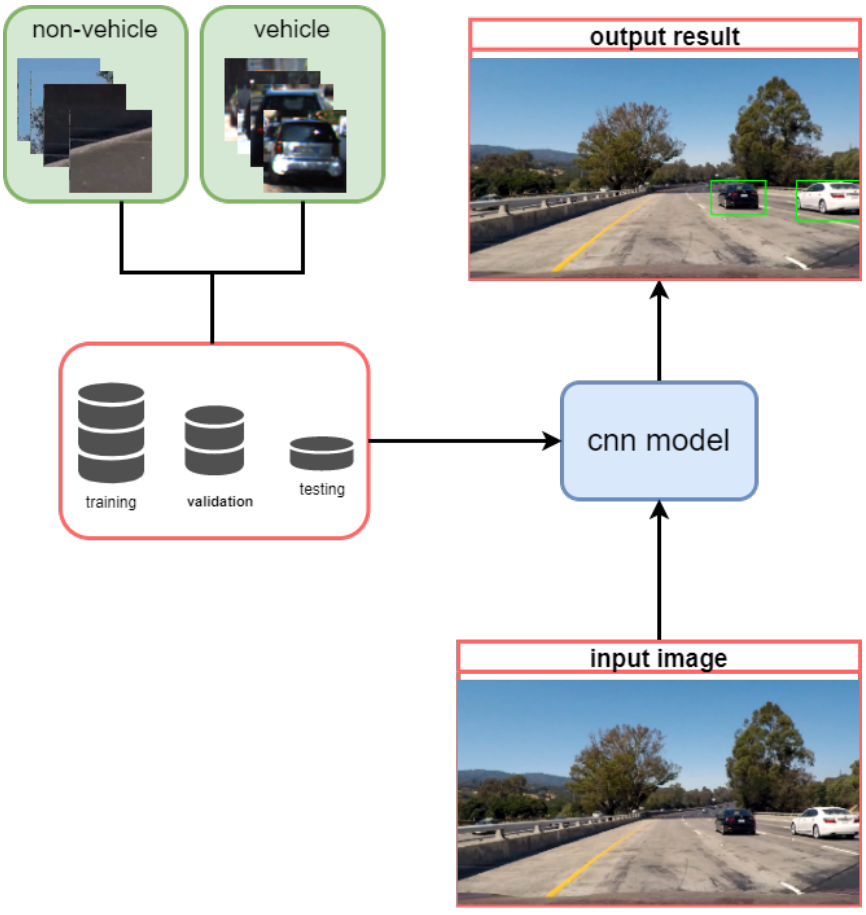

Figure 3.2: Architectural details

3.4 Data preparation

Data preparation is the first step to be applied before using it at CNN, It is divided into three stages.

3.4.1 The dataset used

We used three public dataset from KITTI vision benchmark suite which concerns vehicles, the dataset content tow type for vehicles the first that had cars on it, and the second does not have any Appear vehicles on the dataset [13].

3.4.2 General data center architecture

The dataset is separated into two parts the first part is the vehicle photo and the second photo without the vehicle, The data set needs to be separated into two types because the model cannot read the mixed-up data which makes the model confused and cannot train and learn from the data. The data had to be put into two different folders.

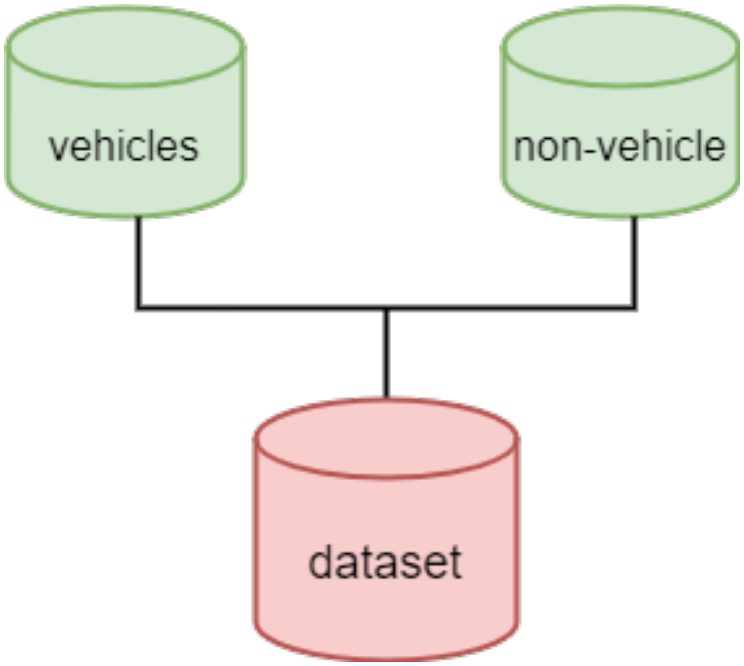

Figure 3.3: data center architecture

3.4.3 dataset Division

The most typical approach for dividing the dataset, in deep learning, is to divide the dataset into training, validation, and test sets. The best ratio of samples dispersed in each set differs for each task. However, as a rule of thumb, a split of 70 train, 20 validation, and 10 test split is typically employed.

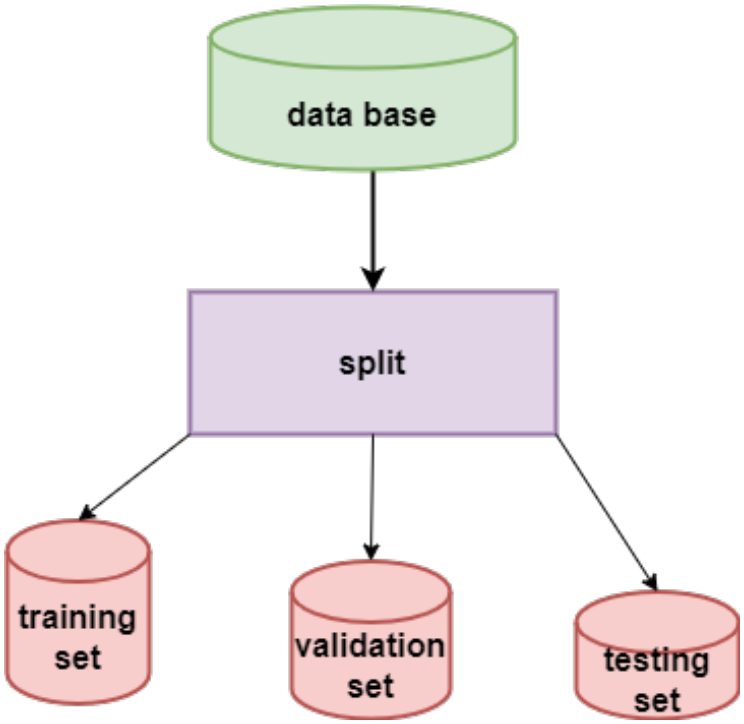

Figure 3.4: dataset Division

3.4.4 Training the model of learning

The learning strategy uses a stochastic gradient descent approach, with a fixed batch size: (a) a series of median corrected image corrections are introduced into the network over a string of periods, (b) a calculated error derivative, and (c) retro-spread across the network by updating network weights. The learning rate is reduced over time so that a local minimum is reached. The resulting learned weights (i.e. the model) are stored for later use at the time of the test. The purpose of the train component is to deduce a function that can well map the input images to their appropriate labels (whether there is a car or not) using the training set. The learning process is based on a convolutionary neural network that is a supervised deep learning neuronal network, and its structure (Figure 3.5)

includes layers of convolution, a layer of pooling, and fully connected layers, among which the convolution layer and the joint layer are essential elements for achieving the extraction of characteristics of CNN. The structure is first arranged alternately by convolution layer and grouping layer to the extraction and mapping of local characteristics from the learning set, and then successively arranged by several fully connected layers. The last layer of the network classifies these entity maps using a soft-max method and gives out scores corresponding to the "probability" that each image belongs to each of the classes learned. The initialization of the CNN settings has a significant impact on the overall accuracy of the prediction. We empirically initialized the hyperparameters and refined them during the training process

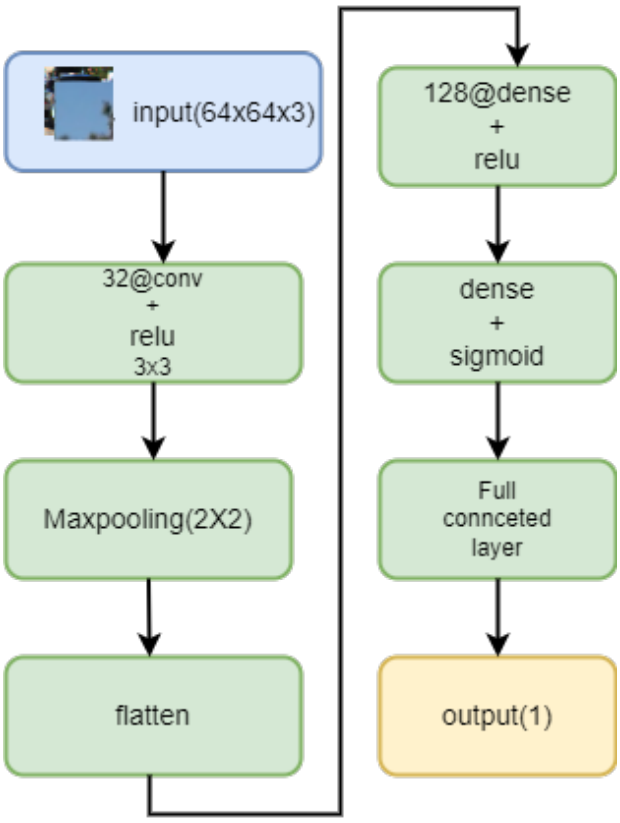

Figure 3.5: the model of learning

	Parameters of	Hyperparameters
Convolutional layer	cores	Size of cores, number of cores, stride, padding, function activation
Pooling layer	–	Pooling methods, filter size, stride, padding
Fully connected layer	Number of weights, function activation	Number of weights, activation function

Table 3.1: The hyperparameters used in the convolutional neural network

to verify our intermediate model, we deploy a second validation set. The accuracy or error rate of the model's predictions on the validation set acts as a feedback parameter for changing different hyperparameters. These hyperparameters include entity card names, convolutionary layer core sizes, layer filter sizes, and network reconfiguration. Through recurrent training-validation cycles, we tweak the hyperparameters and update the training dataset until the learning accuracy reaches a sufficient level. This procedure finally delivers the required architecture for our Convolutional Neural Network (CNN).

3.5 The model testing phase

By sending a picture to the network, of the same size used in the training, we obtain A class prediction based on the model learnt. A test set used once at the conclusion of the project to assess the performance of the final model that is improved and chosen on the learning process using learning and validation sets.

Overall predictions on the set of tests were assessed for true positive (V P), true negative (V N), false positive (F P), and false negative. (F P)These parameters have been used to construct a range of performance metrics, including accuracy (ACC), precision (PR), and F1-score (F1), according to the following equations :

- $$ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

- $$PR = \frac{TP}{TP + FP}$$

- $$F1 = \frac{2.TP}{2TP + FP + NP}$$

3.6 Conclusion

In this chapter, we discussed our suggested technique for creating a classification model, utilizing Deep Learning with a network of convolutive neurons

CNN is really thorough. The development environments and tools utilized in the execution of the suggested technique will be explained in the next chapter.

Chapter 4

Implementation of architecture

4.1 Introduction

In this chapter, we show the creation of a thorough design that brings our technique for developing a deep learning system for the tracking vehicle in a video stream with employing the CNN, presented in Chapter 3.

4.2 Development Environments and Tools

The construction of a learning model may be a resource-intensive job, frequently needing substantial computer capacity. To ease this strain, we harnessed the capabilities of CPUs and visual processors, which are well-suited for performing complicated computations required in machine learning activities. In order to access these computing capabilities, we elected to leverage the Google Colab cloud platform.

Google Colab, also known as Collaboratory, is an effort of Google focused on easing the spread of machine learning research and training. It offers a handy Jupyter Notebook environment that removes the need for local setup and functions fully in the cloud. By employing the Google Colab platform, we were able to leverage the power of distributed computing, enabling us to effectively train and fine-tune our learning model without being constrained by local hardware restrictions.

Python programming uses the integrated development environment,**PyCharm**. It fea-

Figure 4.1: google colab logo

tures a graphical debugger and enables code examination. Additionally, it supports Django web development, the maintenance of unit tests, and version control software integration. It is cross-platform software that can run on Windows, Mac OS X, and Linux and was

Figure 4.2: PyCharm logo

created by the Czech company JetBrains. It comes in a professional version published under a commercial license as well as a community edition distributed under an Apache license.

Python is a popular high-level programming language noted for its simplicity, flexibility, and dynamic nature. One of the unique advantages of Python is that it does not require a compilation process like many other programming languages. Instead, Python code is interpreted and run immediately, allowing for a shorter development cycle.

Figure 4.3: python logo

Python has developed considerable popularity among the developer and programming community due to its user-friendly syntax and simplicity of learning. Its readability and straightforwardness make it a perfect choice for beginners and expert programmers alike. This simplicity not only accelerates the development process but also leads to a considerable decrease in the cost of code maintenance over time.

Another advantage of Python is its wide selection of libraries and packages. These libraries contain modules and methods that extend the functionality of Python, promoting modularity and code reuse. With a huge ecosystem of libraries accessible, developers may tap into existing code and harness the work of others, saving time and effort in accomplishing basic tasks.

Keras A high-level neural network API called Keras was created in Python and can interact with either TensorFlow or Theano. Rapid experimentation was emphasized during its development. The secret to effective research is being able to move as quickly as possible from a concept to a result. Its primary author and maintainer is François Chollet, a Google engineer, and it was created as a component of the ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) project's research endeavor. In 2017, the Google TensorFlow team made the decision to include Keras support in the main TensorFlow library. According to Chollet, Keras was created as an interface rather than an all-encompassing learning system.

It offers a collection of simpler, more understandable higher-level abstractions that make it simple to set up neural networks without relying on a backend computer framework. Microsoft is also attempting to integrate a CNTK backend into Keras.

Figure 4.4: keras logo

NumPy A key component of the Python environment is the NumPy library, which supports massive and multidimensional tables and matrices as well as a wide variety of mathematical operations to manipulate and interpret them. The name "NumPy" means "Numerical Python", highlighting the program's focus on digital computing tasks.

Opencv : A feature-rich open-source program with a large selection of tools and functions for computer vision and image processing applications, OpenCV is also known as Open Source Computer Vision. It is a popular option across many businesses and scientific disciplines because of its versatility and wide range of capabilities.

Figure 4.5: opencv logo

With support for several programming languages including C++, Python, Java, and MATLAB, OpenCV caters to a sizable developer community. A broad variety of developers will have access thanks to the multi-language support. Users may effectively alter, examine, and extract useful data from photos and movies using OpenCV.

The program includes a wide variety of capabilities that allow users to carry out activities including altering photos, looking through visual content, and collecting pertinent information. Developers may deal with a variety of image and video datasets thanks to OpenCV's extensive range of tools and methods.

4.3 Preparation of the data

We are going to divide the data for distribution into two types: the first is for training, and the second is for testing (80% of the training data will be set aside for validation, while the remaining 20% will be used for the actual training). Before all this, we first read the data from the folder, as shown in figure.

```
    dataFile = 'dataset.p'

   if not os.path.isfile(dataFile):
       tryGenerateNew = aux.promptForInputCategorical(message='data file not found. Attempt to generate?',
                                                      options=['y', 'n']) == 'y'

       if tryGenerateNew:
           vehicleFolder = 'dataset\\vehicles\\cov'
           nonVehiclesFolder = 'dataset\\nonvehicles\\cov'
```

Using skLearn utils to split data to train and test sets

```
from sklearn.model_selection import train_test_split as trainTestSplit
xTrain, xTest, yTrain, yTest = trainTestSplit(imageSamplesFiles, y, test_size=0.2, random_state=42)
```

```
3
```

Further split train data to train and validation

```
from sklearn.model_selection import train_test_split as trainTestSplit
xTrain, xVal, yTrain, yVal = trainTestSplit(xTrain, yTrain, test_size
    =0.2, random_state=42)
```

function of	arguments
trainTestSplit	**xTrain, yTrain**: These variables represent the original training dataset, consisting of input data samples (xTrain) and their corresponding labels (yTrain). **test size** is the proportion of the dataset that will be allocated for testing. In this case, 0.2 indicates that 20% of the data will be used for testing, and the remaining 80% will be used for training. **random state** is an optional parameter that sets the random seed for reproducibility. It ensures that the same random splitting of data occurs every time you run the code, enabling consistent results.

Table 4.1: Divide of dataset parameter

4.4 Realization of the CNN model

To build deep neural networks with Keras, we first import the various libraries and modules as shown in Listing code and described in the following Table

```
from sklearn.model_selection import train_test_split as trainTestSplit
from keras.layers import Conv2D, Flatten, denes,Lambda, MaxPooling2D,
    Dropout
from keras.models import Model, Sequential
import pickle
import numpy as np
import os
import helper as aux
import glob
from keras.callbacks import ModelCheckpoint
```

Libraries/modules	Descriptions
Numpy	Is the basic library for scientific computing in Python. It provides a multidimensional array vehicle with high performance and tools to work with these paintings.
dense	Is used to instantiate a dense layer
Activation	Allows to add an activation function to the sequence of layers
Flatten	Converts the map of grouped features into a single column that is passed to the fully connected layer.
Maxpoling2D	Downsampled the input representation by taking the maximum value over the window defined by the size of the pool for each dimension along the feature axis
Adam	Optimizer that implements the Adam algorithm. Adam's optimization is a stochastic gradient descent method based on the adaptive estimation of the moments of the first and second
ModelCheckpoint	Reminder to save the Keras model or the weights of the pattern at a certain frequency

Table 4.2: Description of libraries and modules used

4.5 Parameter initialization CNN

Before we can start training the network, we need to initialize its parameters. The details of the hyperparameters, used in this study, are described in this the table

Hyperparameters	Descriptions	values
image width, height of the image	Input image size	64X64
batch size	is a hyperparameter that defines the number of samples to be processed before updating settings of the internal model	we use 64
Class num	the number of class	2

Table 4.3: DataSET loading

4.6 Network learning settings

Our deep learning model (CNN) was designed and configured utilizing a variety of human network setups and heuristics. We outline the specifications of the CNN architecture that has been presented.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_input (InputLayer)   [(None, 64, 64, 3)]       0

 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 128)               3936384

 dense_1 (Dense)             (None, 1)                 129

 flatten_1 (Flatten)         (None, 1)                 0

=================================================================
Total params: 3,937,409
Trainable params: 3,937,409
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.6: CNN settings

4.7 Results obtained

To visualize the performance of our deep learning CNN over time during training, we have created:

— a graph of "accuracy" on the train "acc" dataset on the training epochs (Figure 4.7).

— a graph of "loss" on the train dataset "loss" over the training epochs (Figure 4.8).

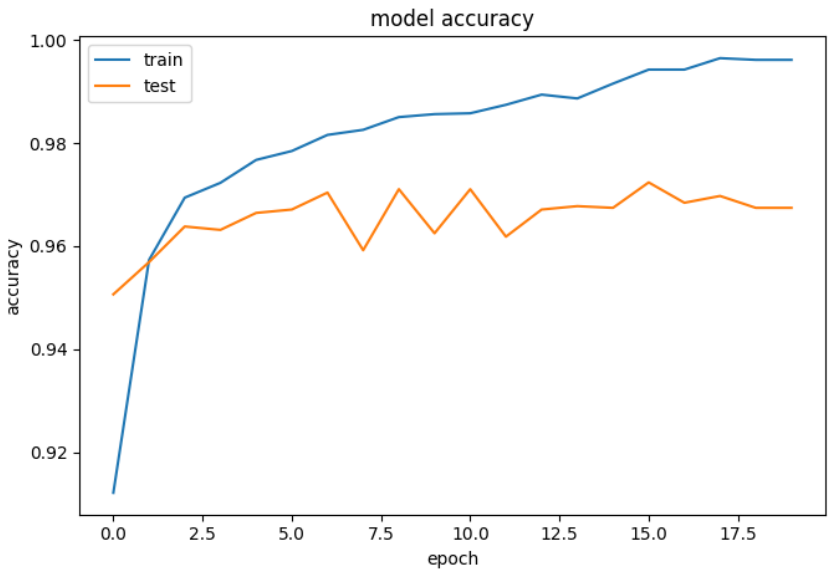

Figure 4.7: accuracy plot

In deep learning, the checkpoint is the stored weights of the model when there is an improvement in classification accuracy on the validation dataset, these weights can be used to make predictions as is, or used as a basis for continuing education. In our case, the best validation accuracy value is in epoch number 19. At that time, the values are :

— Training loss 'loss' is 0.22%

— Training accuracy 'accuracy' 99.72%

— Validation loss 'Val loss 2.41%

— Validation accuracy 'Val accuracy' 97.19%

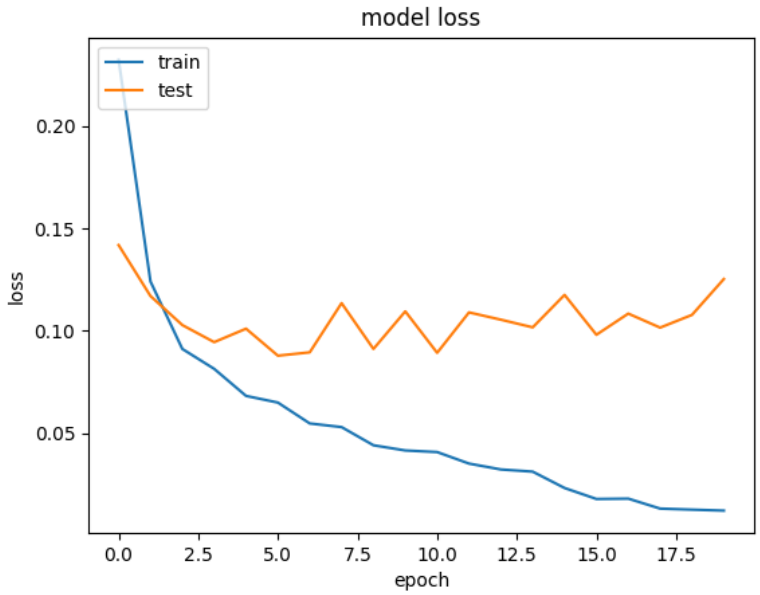

Figure 4.8: loss plot

4.8 Model evaluation on test data

```
this results in reel number
result2 : [[3712,160],
           [64,3664]]
```

Each row of the confusion matrix represents instances of an actual class and each column represents instances of a predicted class(The Matrix has this percentage number). The diagonal elements represent the percentage for which the predicted label is equal to the true label, while anything outside the diagonal was mislabeled by the system, Our model CNN system can perfectly predict: 3712 images of non-vehicle, 3664 vehicle We note that the model makes many errors(64 errors for the non-vehicle,160 for the vehicle). .Other ranking results are almost very good:

— F1 model score is: 97.01%

— PR model score is: 99.82%

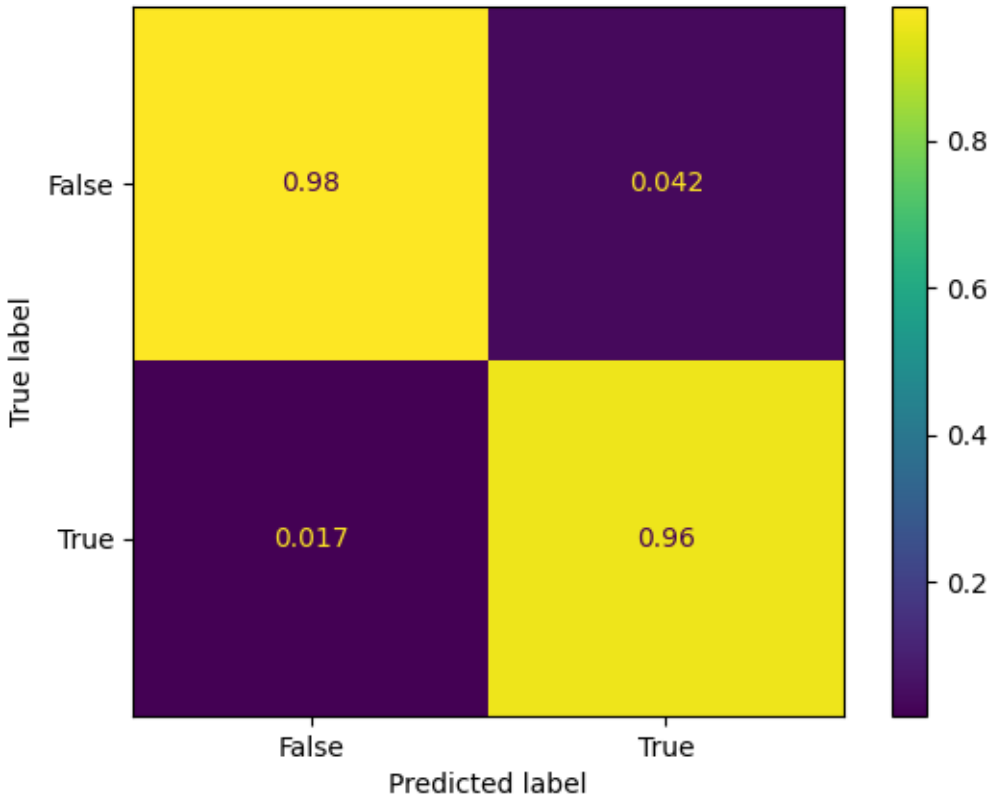

Figure 4.9: matrix confusion

4.9 screenshot of result

the video in been processing in detector.py by cute the video into frames the main(we use pov car driving) analyse the frames by import same function other file that help processed frames and use the model to define the car and draw the box on car .

the input frames :

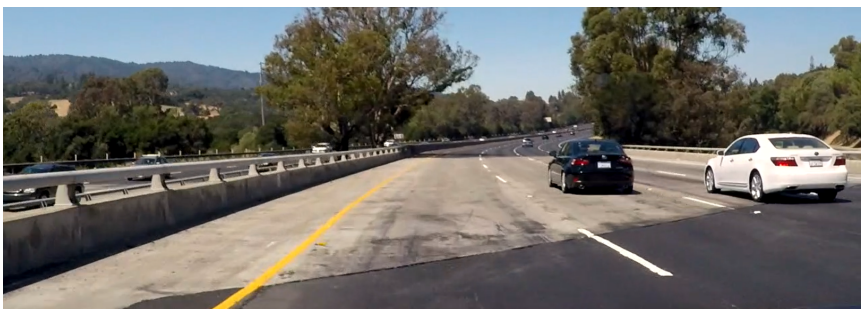

Figure 4.10: input frames before processing

output :

Figure 4.11: screenshot from videos of output

Figure 4.12: screenshot from videos of output

as a result to the we use the heat map to show the vehicle and not be confuse by the multi vehicles in the frame .

Figure 4.13: heat tmap

Chapter 5

Conclusion and perspectives

In this work, we have studied the use of deep neural networks, in particular convolutional neural networks, for the problem of Tracking and identifying vehicles

The deep learning CNN model proposed to classify the model to track the cars (CNN) excels in learning relevant features directly from datasets of a vehicle and non-vehicle

After successfully training the model, the "accuracy" of the test is calculated:

for 76000 images belonging to 2 classes, we reached 99.72% "testing accuracy" and 97.01% accuracy. From this assessment, we can conclude that:

— As the number of samples increases, the performance of the CNN model increase considerably. However, there are some areas where this research might be strengthened that we believe offer important insights for our work :

- make the system notify the driver of any cars coming close to him on the road.
- make a model to classify the passage.
- Use of another dataset for training, validation, and testing.
- Optimize the system for live detection.

Bibliography

[1] Object Tracking Methods. = https://medium.com/augmented-startups/top-5-object-tracking-methods-92f1643f8435. Accessed: 2021-11-08.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*. Springer, 2016.

[3] Dinesh Suresh Bhadane, Pritam B Bharati, Sanjeev A Shukla, Monali D Wani, and Kishor K Ambekar. A review on gsm and gps based vehicle tracking system. *International Journal of Engineering Research and General Science*, 3(2), 2015.

[4] Abhay Singh Bhadoriya, Vamsi Vegamoor, and Sivakumar Rathinam. Vehicle detection and tracking using thermal cameras in adverse visibility conditions. *Sensors*, 22 (12):4567, 2022.

[5] X Binjie and Jinlian Hu. Fabric appearance testing. In *Fabric testing*, pages 148–188. Elsevier, 2008.

[6] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[7] Wei Chen. Multiple-oriented and small object detection with convolutional neural networks for aerial image, September 2019. URL `http://https://www.researchgate.net/publication/335918494_Multiple-Oriented_and_Small_Object_Detection_with_Convolutional_Neural_Networks_for_Aerial_Image`.

[8] Francois Chollet. *Deep learning mit python und keras: das praxis-handbuch vom entwickler der keras-bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[9] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[10] Tiziana D'Orazio and Grazia Cicirelli. People re-identification and tracking from multiple cameras: A review. In *2012 19th IEEE International Conference on Image Processing*. IEEE, 2012.

[11] Issam El Naqa and Martin J Murphy. *What is machine learning?* Springer, 2015.

[12] S Gangadhar and R Shanta. Chapter 8-machine learning. *Handbook of Statistics*, 38: 197–228, 2018.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[15] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[16] Simon Haykin. *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Earnest Paul Ijjina, Dhananjai Chand, Savyasachi Gupta, and K Goutham. Computer vision-based accident detection in traffic surveillance. In *2019 10th International conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2019.

[19] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE access*, 7:128837–128868, 2019.

[20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[21] John D Kelleher. *Deep learning*. MIT press, 2019.

[22] Nico Klingler. Object tracking in computer vision (2023 guide). = https://viso.ai/deep-learning/object-tracking/. Accessed: 2021-11-08.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 1998.

[24] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[26] R Manikandan, R Ramakrishnan, and R Scholar. Human object detection and tracking using background subtraction for sports applications. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(10):4077–4080, 2013.

[27] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[28] Tom Michael Mitchell et al. *Machine learning*, volume 1. McGraw-hill New York, 2007.

[29] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 445–461. Springer, 2016.

[30] Donald J Norris. *Machine Learning with the Raspberry Pi*. Springer, 2020.

[31] Edward B Panganiban and Jennifer C Dela Cruz. Rfid-based vehicle monitoring system. In *2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. IEEE, 2017.

[32] Himani Parekh, Darshan Thakore, and Udesang Jaliya. A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering*, page 9, February 2014.

[33] Himani S. Parekh, Darshak G. Thakore, and Udesang K. Jaliya. A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering*, 2:2970–2978, 2014.

[34] Robert Pless, Tomas Brodsky, and Yiannis Aloimonos. Detecting independent motion: The statistics of temporal continuity. *IEEE transactions on pattern analysis and machine intelligence*, 22(8):768–773, 2000.

[35] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[36] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[37] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[38] Ni Ni San Hlaing, Ma Naing, and San San Naing. Gps and gsm based vehicle tracking system. *International Journal of Trend in Scientific Research and Development (IJTSRD)*, pages 271–275, 2019.

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[40] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. *Advances in neural information processing systems*, 26, 2013.

[41] Joseph Awoamim Yacim. Impact of artificial neural networks training algorithms on accurate prediction of property values, Nov 2018. URL `https://www.researchgate.net/figure/The-structure-of-the-artificial-neuron_fig2_328733599`.

[42] Kaihua Zhang, Lei Zhang, Qingshan Liu, David Zhang, and Ming-Hsuan Yang. Fast visual tracking via dense spatio-temporal context learning. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014.

[43] Kaihua Zhang, Qingshan Liu, Yi Wu, and Ming-Hsuan Yang. Robust visual tracking via convolutional networks without training. *IEEE Transactions on Image Processing*, 25(4), 2016.

[44] Xian-Da Zhang. A matrix algebra approach to artificial intelligence. 2020.