A HYBRID DEEP LEARNING ARCHITECTURE FOR VIDEO-BASED AUTOMATIC VEHICLE DETECTION

2023
MASTER THESIS
COMPUTER ENGINEERING

Mohammed Abduljabbar ZAID

Thesis Advisor
Assist. Prof. Dr. Muhammet ÇAKMAK

A HYBRID DEEP LEARNING ARCHITECTURE FOR VIDEO-BASED AUTOMATIC VEHICLE DETECTION

Mohammed Abduljabbar ZAID

Thesis Advisor
Assist. Prof. Dr. Muhammet ÇAKMAK

T.C.
Karabuk University
Institute of Graduate Programs
Department of Computer Engineering
Prepared as
Master Thesis

KARABUK
October 2023

I certify that in my opinion the thesis submitted by Mohammed Abduljabbar ZAID titled “A HYBRID DEEP LEARNING ARCHITECTURE FOR VIDEO-BASED AUTOMATIC VEHICLE DETECTION” is fully adequate in scope and in quality as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Muhammet ÇAKMAK .....
Thesis Advisor, Department of Computer Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a Master of Science thesis. October 18, 2023

Examining Committee Members (Institutions)	Signature
Chairman : Assoc. Prof. Dr. Zafer ALBAYRAK (SUBU)	.....
Member : Assist. Prof. Dr. Muhammet ÇAKMAK (SNÜ)	.....
Member : Assist. Prof. Dr. Sait DEMİR (KBU)	.....

The degree of Master of Science by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Assoc. Prof. Dr. Zeynep ÖZCAN .....
Director of the Institute of Graduate Programs

"I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well."

Mohammed Abduljabbar ZAID

ABSTRACT

M. Sc. Thesis

A HYBRID DEEP LEARNING ARCHITECTURE FOR VIDEO-BASED AUTOMATIC VEHICLE DETECTION

Mohammed Abduljabbar ZAID

Karabük University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Muhammet ÇAKMAK
October 2023, 72 pages

Detecting vehicles in Intelligent Transportation Systems (ITS) is important to maintaining road safety, monitoring vehicle flow, identifying illegal vehicle types, detecting incidents, and estimating vehicle speeds. Despite the increasing prevalence of scholarly inquiry, the issue at hand continues to be a formidable obstacle that needs resolution. Proposed hardware-based alternatives, such as Radars and LIDAR (Light Detection and Ranging) remote sensing, have been deemed impractical due to their high cost of maintenance and limited provision of relevant information to human operators in Surveillance systems.

To effectively monitor transportation systems, it is imperative to implement two crucial measures for surveillance: vehicle detection and license plate recognition. The current methodologies for vehicle detection utilize feed-forward convolutional neural

networks (CNNs) as backbone architectures. However, these are scale-sensitive and cannot handle variations in vehicles' scales in sequential video frames.

To address these issues of vehicle detection in road surveillance, such as variations in illumination, vehicle scale variations, and occlusions. we introduce the Scale Invariant Hybrid Convolutional Neural Network (SIH-CNN) in conjunction with YOLO (You only look once) for real-time vehicle detection. The SIH-CNN is meticulously crafted to be resilient against size variations and adept at accommodating vehicles of varied sizes within sequential video frames, ensuring consistent performance regardless of the vehicle's position relative to the camera. By leveraging YOLO's rapid and accurate object detection capabilities, our hybrid model excels in identifying and classifying vehicles of all shapes and sizes across a diverse range of scenarios, setting a new benchmark in adaptability and efficiency for vehicle detection systems in practical applications.

Results show that the proposed SIH-CNN model achieved a mean average precision (mAP) of 77.76% on the UA-DETRAC benchmark, which is 3.94% higher than the baseline detector with real-time performance of 48.4 frames per seconds.

Key Words : Intelligent Transportation Systems, Automated surveillance, Vehicle detection, Convolutional neural network, Scale invariant.

Science Code : 92431

ÖZET

Yüksek Lisans Tezi

VİDEO TABANLI OTOMATİK ARAÇ TESPİTİ İÇİN HİBRİT BİR DERİN ÖĞRENME MİMARİSİ

Mohammed Abduljabbar ZAID

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Muhammet ÇAKMAK
Ekim 2023, 72 sayfa

Akıllı Ulaşım Sistemlerinde (ITS) araçların tespiti yol güvenliğinin sağlanmasında hayati öneme sahiptir. Araç akışının izlenmesi suça karışan araçların belirlenmesi, tehlike oluşturan olayların tespit edilmesi ve araç hızlarının tahmin edilmesi için kullanılmaktadır. Her ne kadar araç akışlarının izlenmesi ile ilgili çalışmalar olmasına rağmen gerçek zamanlı ve data etkin tespit yapan sistemlere ihtiyaç duyulmaktadır. Diğer taraftan pratikte kullanılmakta olan Radar ve LIDAR gibi donanım tabanlı sistemler yüksek bakım maliyeti ve uzman personele ihtiyaç gerektirmektedir.

Ulaşım sistemlerini etkili bir şekilde izlemek için araç tespiti ve plaka tanıma sistemlerinin beraber çalışması gerekir. Araç tespiti için uygulanan metodolojiler, ağırlıklı temel mimari çerçeve olarak ileri beslemeli evrişimli sinir ağlarını (CNN) kullanır. Ancak, bu yöntemler ölçek değişikliklerine duyarlı olduğu ve çeşitli boyutlardaki araçları birden fazla video karesi içinde etkili bir şekilde gösterme

sorunları yaşamaktadır. Bu tezde, gerçek zamanlı araç algılama sorunlarına çözüm olarak YOLO (You Only Look Once) (Yanlızca bir kere bakar) tabanlı SIH-CNN (Scale Invariant Hybrid Convolutional Neural Network Convolutional Neural Network) (Ölçek Değişmez Hibrit Evrişimli Sinir Ağı) altyapısını kullanan hibrit bir tespit sistemi önerilmiştir. SIH-CNN boyut değişikliklerine karşı hassastır. Ayrıca sıralı video karelerinde farklı boyutlardaki araçların görüntülerini daha iyi alır ve aracın kameraya göre konumundan bağımsız olarak tutarlı sonuçlar üretir. YOLO ve SIH-CNN'nin hızlı ve doğru nesne algılama özellikleri kullanan hibrit model, çeşitli senaryolarda tüm şekil ve boyutlardaki araçları tanımlama ve sınıflandırma konusunda üstünlük sağlamıştır.

Bu tezde önerilen SIH-CNN modeli kamuya açık UA-DETRAC kıyaslama ölçütü kullanılarak değerlendirilmiştir. Deney sonuçları, sunulan SIH-CNN modelinin UA-DETRAC kıyaslama veri kümesi üzerinde değerlendirildiğinde %77,76'lık bir ortalama hassasiyet (mAP) elde ettiğini göstermektedir. Performans ölçütü, temel dedektöre göre %3,94'lük bir üstünlük sergilemektedir.

Anahtar Kelimeler : Akıllı Ulaşım Sistemleri, Otomatik gözetim, Araç algılama, Evrişimsel sinir ağı, Ölçek değişmezliği.

Bilim Kodu **:** 92431

ACKNOWLEDGMENT

I want to express my sincere gratitude to everyone who made it possible for me to complete this thesis. First, I would like to thank my thesis advisor, Asist. Prof. Dr. Muhammet ÇAKMAK, for his help in organizing my project and writing this thesis. He gave me promising ideas and pushed me to do my best. I am grateful to the Department of Computer Engineering for allowing me to complete this thesis. I would also like to thank Karabuk University for their support. I would also like to acknowledge with much appreciation the crucial role of the Karabük University Kamil Güleç Library staff, who permitted the use of all necessary resources to complete the task. I sincerely thank my parents and family, who provided me with endless inspiration, continuous encouragement, and support throughout my studies. Finally, I am indebted to all my friends who supported me in any way during the completion of the project.

CONTENTS

Page

Page

Page

LIST OF FIGURES

Page

Page

LIST OF TABLES

Page

SYMBOLS AND ABBREVITIONS INDEX

SYMBOLS

∑ : weighted sum

Λcoord : Coord is Regularize for coordinates loss

σ : Sigma

∈ : is an element of

∫ : integral

ABBREVITIONS

AI : Artificial Intelligence

ML : Machine learning

CNN : Convolutional Neural Network

ITS : Intelligent Transportation Systems

SIH-CNN : Scale Invariant Hybrid Convolutional Neural Network

DCB : Dense connection block

ROI : Region 0f Interest

STVD : Swin Transformer-based Vehicle Detection

FPGA : Field Programmable Gate Array

DL : Deep Learning

TLU : Threshold Logic Unit

MLP : Multilayer Perception

IOU : Intersection of Union

TP : True Positive

FP : False Positive

TN : True Negative

FN : False Negative

TPR : True Positive Rate

FPR : False Positive Rate

AUC : Area under Curve

(mAP) : Mean Average Precision

DPM : Deformable Part Models

RPN : Region Proposal Network

FPN : Feature Pyramid Network

YOLO : You Only Look Once

BN : Batch Normalization

PART 1

INTRODUCTION

1.1. CONTEXT AND MOTIVATION

The aim of computer vision, a field within artificial intelligence, is to enable computers to comprehend visual stimuli like humans do. Specifically, this entails equipping computers to extract perceptible insights from visual data. By doing so, robots can accurately delineate the characteristics of objects, recognize them, and comprehend object motion patterns [1,2].

In recent years, there has been a significant increase in accessible visual data due to the proliferation of smart devices, cameras, and sensors [3]. As of 2021, the worldwide number of connected devices has reached 34 billion, which means an average of around 6 gadgets per person [4]. According to the cited source, visual data is being generated at around one million video minutes per second on the internet [5]. Relying on manual human efforts to comprehend and extract information from videos and pictures has become infeasible. Therefore, it is essential to prioritize the development of algorithms that allow computers to perceive and interpret the visual world. Integrating various computer vision methodologies with deep learning capabilities facilitates the advanced processing of visual data [6]. This approach simplifies depicting prominent objects and their movement in video footage.

Computer vision is a cross-disciplinary field with multiple applications, including healthcare, engineering, and transportation. In contemporary traffic monitoring infrastructures, surveillance cameras are extensively utilized and available, serving as a crucial element of such systems [7]. The cameras provide significant data for various transportation applications, such as parking management, identifying and detecting vehicles, counting vehicles and pedestrians, and identifying license plates. The

inclination towards creating smarter cities is increasing, and with this goal, using multiple technologies to enhance transportation planning, optimize traffic flow, and improve road user safety is necessary [8].

Within the realm of computer vision, the primary purpose of object detection involves accurately identifying and localizing specific objects present in an image. In contrast to objectives such as object segmentation or semantic segmentation where the goal is to group pixels homogeneously based on semantic significance and other criteria [9], object detection often involves providing the coordinates of a bounding box around the object to localize it.

The goal is to precisely identify and categorize objects in an image with a methodology miming human visual perception. Different techniques exist for creating a bounding box around an object in an image. It should be noted that the definition of a bounding box may differ in cases of vehicle occlusion. A bounding box may only encompass the visible segment of a vehicle when the vehicle is partially obscured. Differs from the conventional approach utilized by most datasets, which usually generates a bounding box that includes the entire vehicle, even its obscured sections [10].

1.2. AIMS

This thesis aims to:

- Develop a model capable of efficiently meeting the real-time performance demands of vehicle recognition applications.
- This study aims to design a system that can analyze video frames captured by surveillance cameras situated in urban areas, such as parking lots and roads. This investigation focuses on various vehicle types, including cars, buses, vans, and trucks. The footage captured from these scenes presents numerous challenges, such as changes in light and sound due to varying weather conditions, obstructions that wholly or partially obstruct the view, and fluctuations in vehicle velocity.

1.3. IMPORTANCE AND CONTRIBUTION

This study contributes to the fields of surveillance and automated vehicle recognition; the importance and contribution of this work can be summarized as follows:

- The research supports the development of real-time surveillance systems essential for various applications, including security and law enforcement.
- Enhancement of Vehicle Detection Precision: This study introduces a new technique that incorporates multi-level features and inter-frame information to advance the precision and effectiveness of vehicle detection, particularly for distant and compact cars. The presented method represents a significant improvement in the field of vehicle detection.
- To tackle the problem of scale variation in moving vehicles, we've suggested a Scale Invariant Hybrid Convolutional Neural Network (SIH-CNN) design. This methodology enhances detection precision by competently handling the differing dimensions of vehicles in successive frames. The SIH-CNN model is created to be scale-invariant, enabling it to adjust to the fluctuating scales of vehicles, producing more dependable and precise vehicle detection.
- We have introduced a multi-level feature extraction block to address the problem of gradient vanishing and improve the model's capability to handle class variation. This component plays a critical role in enhancing the overall robustness of the detection system by efficiently extracting hierarchical features from the input data and capturing essential information at various abstraction levels, ultimately resulting in better detection performance.
- Our proposed scheme incorporates a multi-scale feature extraction block to improve the detection of small vehicles. That is crucial since traditional detection methods struggle with identifying small-sized vehicles. By extracting features at different scales, our model can more accurately detect and recognize these tiny vehicles, resulting in an overall enhancement in the performance of the vehicle detection system.
- Our proposed SIH-CNN operates at an impressive 48.4 frames per second, making it highly efficient and well-suited for real-time surveillance applications.

1.4. PROBLEM STATEMENT

- Vehicles come in various sizes, styles, colors, and shapes. Because of this variety, detection systems face a tremendous barrier in reliably and consistently identifying.
- Changes in Geometry of Cars: The position and orientation of vehicles in successive video frames may affect their geometry. These changes can also lead to additional complexities in recognizing and detecting vehicles.
- Environmental Variables: The vehicle detection and recognition process can be affected by environmental factors Such as dust, rain, and clouds, potentially compromising the accuracy of results.
- Many object detectors in vehicle surveillance models are designed for single-image detection, disregarding significant inter-frame information from video feeds.
- Grid Size Limitations: Using grid cells for detection in specific systems has limitations. A large grid size can detect small objects but at a high computational cost. Meanwhile, smaller grid sizes are computationally efficient but often fail to detect vehicles.
- Fixed-length grid cell systems may have difficulty predicting vehicles further away from the cameras. Some existing systems, like those that use DarkNet-19 as their backbone architecture, overlook multi-level features crucial for handling variations in vehicle classes.

1.5. HYPOTHESIS

The current research introduces the recognition system with the following hypotheses:

Hypothesis 1: A vehicle detection system utilizing the YOLO object detector in conjunction with the Scale Invariant Convolutional Neural Network (SICNN) backbone architecture will enhance vehicle localization and categorization accuracy.

Hypothesis 2: The implementation of the Scale Invariant Hybrid Convolutional Neural Network (SIH-CNN) architecture, with the inclusion of the Dense Connection

Block (DCB) and Inception Block, can effectively alleviate the problem of gradient vanishing during backpropagation and improve feature extraction, resulting in enhanced vehicle detection capabilities.

Hypothesis 3: Implementing the global pooling layer in (SIH-CNN) can decrease overfitting and reduce the number of parameters in the model, thus enhancing model performance when detecting vehicles.

1.6. BLOCK DIAGRAM FOR VEHICLE DETECTION PROCESS

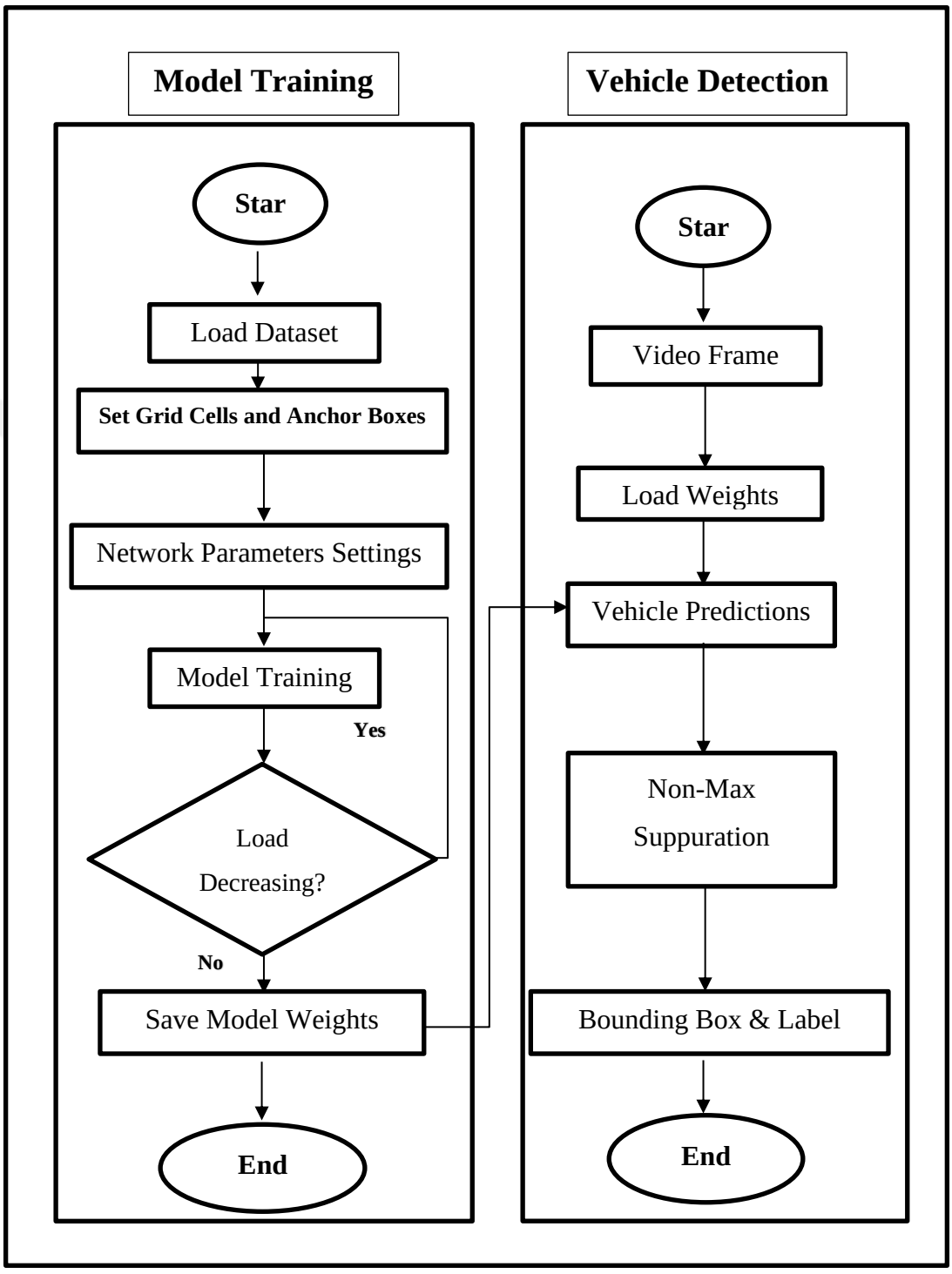

Figure 1.1. Vehicle Detection Process.

PART 2

LITERATURE REVIEW

Surveillance systems detecting moving vehicles are crucial for intelligent transportation, traffic management, and autonomous vehicles. As innovative city development grows, various technologies are utilized to improve transportation planning, optimize traffic flow, and enhance road user safety. That necessitates using automated surveillance systems in these areas [11]. Due to the high cost and limited availability of human monitors, reliable and easily accessible automated surveillance systems are essential. In this review of the relevant literature, we examine and contrast various studies investigating vehicle detection methods' effectiveness in urban zones.

Hu et al. established a framework for visual vehicle detection in [12]. Their suggested approach used Region-of-Interest (RoI) pooling that considered the surrounding context to generate precise feature maps. For car categorization, they used a multi-branch decision network. The suggested system is trained using the VGG and PVANET CNN backbones. Their fundamental architectures are rudimentary feed-forward neural networks, notorious for missing detections and ignoring a wealth of practical semantic context.

A fast vehicle detection framework for traffic monitoring was proposed by Zhang et al. [13]. They introduced (connect and merge residual networks) to enhance classification accuracy. They developed a multi-scale prediction network to further fine-tune their vehicle-shape forecasting abilities. Redmon, Farhadi, and Darknet-19 [14] adopted the Darknet-19 backbone architecture in YOLO V2. Darknet-19 fails to detect small-scale vehicles because it disregards multi-level characteristics and pools in descending tiers.

Overfitting problems were avoided in the technique proposed by Haritha and Kumar [15]. A method for detecting vehicles using a cascaded neural network was proposed by Wu et al. They worked together to merge two distinct CNNs. The first convolutional neural network only deals with a limited amount of variation data. The second convolutional neural network was created to pick and choose features. Each CNN was first implemented independently, and then they were joined for optimal performance. Vehicles with a large inter-class variation and those too tiny for their planned network to detect.

Using Haar-like characteristics and the histogram of directed gradients, Alam et al. [16] offer a novel computer vision-based vehicle recognition system. The Gentle Adaptive Boosting algorithm handles hypothesis creation, while the support vector machine algorithm deals with false hypothesis screening. The approach successfully detects cars by using their outline and shape. However, Haar-like feature creation could identify erroneous vehicles.

The method presented by Gomaa et al. [17] is a hybrid of YOLOv2 and the work of Redmon and Farhadi. [14] In addition, we utilize point motion analysis to identify and tally moving cars precisely. Several convolutional neural networks are studied to determine which yields the best detection results. Important multilevel and multiscale vehicle detection aspects are overlooked in the suggested approach.

Deshmukh et al. [18] present a framework for unstructured traffic environment vehicle detection using Swin Transformers. The Swin transformer and a bidirectional feature pyramid network extract multi-scale feature effectively and robustly, respectively. With the STVD framework, detection accuracy is greatly improved over prior approaches. However, their suggested architecture does not consider the multilevel aspects crucial to mitigating the effects of the gradient vanishing problem in backpropagation.

Fast R-CNN-based day/night vehicle detection and classification is the focus of [19], authored by Arora et al. 2022. The proposed approach is effective even in little light, lengthy shadows, and heavy foot activity. It shows promise in terms of memory recall,

data correctness, and processing speed. However, the suggested architecture does not account for the multiscale and multilayer aspects necessary to deal with the varying sizes of vehicles in different video frames.

An enhanced version of the SSD [20] algorithm for rapid vehicle detection in traffic scenes is presented by Chen et al. in [21]. It uses MobileNet v2 as its central feature extraction network, which boosts the system's responsiveness in real-time. The technique achieves High average precision across multiple datasets, showcasing its enhanced inference speed and prediction accuracy. The trade-off between detection accuracy and computing resources is a weakness of their methodology.

Field Programmable Gate Array (FPGA) with YOLO CNN (Redmon & Farhadi, 2017) is proposed as a low-power, low-latency, high-precision, and adjustable vehicle detector (Zhai et al., 2023) [22]. The method can significantly shrink the model's size while maintaining accuracy by using dynamic threshold structured pruning and dynamic 16-bit fixed-point quantization for model compression. Their methods for optimizing hardware improve both efficiency and effectiveness in using available resources during computing. However, their proposed framework does not consider the multiscale properties essential for dealing with vehicle scale fluctuation in video frames.

Mittal et al. [23] present a hybrid model of Faster R-CNN and YOLO [23] with a majority voting classifier for vehicle detection and traffic density estimates. Methods such as transfer learning and data augmentation improve the effectiveness of models. The suggested model provides superior detection accuracy and traffic density estimation compared to YOLO and Faster R-CNN. However, their ensemble approach does not consider multilevel features that are essential for managing vehicle class variance.

An enhanced version of YOLO [14] for vehicle detection is presented by Ghosh. in [24]. It predicts class probabilities with a single pass and achieves great accuracy with a small number of convolutional layers. The proposed method achieves better results on publicly available datasets than competing methods. The key problems with this

technology for application in Transportation systems are its limited focus on identifying moving vehicles and its possible generalizability.

However, their proposed network failed to detect extremely small vehicles and vehicles with a high inter-class variation. Table 2.1. critically analyses and briefly describes various vision-based detectors.

Table 2.1. Analyses and briefly describes various vision-based detectors.

	Detection Algorithm	Contribution	Limitations	Accuracy mAP %	Sensitivity	Complexity (Time / FPS)
Two-stage detectors	R-CNN [25]	Bypass the issue of selecting large number of regions and utilize Selective Search Algorithm (SSA) for region proposal then apply CNN based classifier to classify these regions.	Selective Search results in bottleneck. The detection speed is slow 47 seconds per image. Not suitable for real time applications.	Low 66	Low	Medium
	Fast-RCNN [26]	Introduce the convolutional feature map instead of SSA (i.e., used in R-CNN).	Not appropriate for real time applications. The detection speed is time-consuming only 5-frames per seconds.	Low 66.9	Low	Medium
	Faster-RCNN [27]	Introduce Region-Proposal-Network (RPN) that replace SSA for region proposal.	Proposed detector is position sensitive and translation invariant. And detection speed not good as single stage detectors.	Medium 66.9	Medium	Medium
	R-FCN [28]	Introduce the position-sensitive score maps to overcome the problems of Faster -RCNN.	Proposed detector is not suitable for real time detection problem.	High 82	Low	High
Single-stage detectors	YOLO [29]	YOLO is first single stage detector, that introduced regression-based detection. Proposed detectors classify and localize objects simultaneously.	The detection accuracy is very low. Large grid cells result in miss detection of small-scale vehicles. Poor localization.	Low 57.9	Medium	Low
	SSD [20]	Introduced more anchor points to precise localization. Utilize multi scale features to enhance detection accuracy for small scale vehicles.	Poor detection accuracy specially for small scales vehicles when vehicles are away from the cameras.	Low 70	Medium	Low
	YOLO V2 [14]	Introduced multi-scale training, adoptive anchor boxes and proposed Darknet19 a CNN based backbone architecture.	Proposed model cannot handle scale variation of vehicles in consecutive video frames.	Low 76.8	High	Low

Retina-Net [30]	Proposed a single stage detector with focal loss, to resolve the issue of class imbalance problem during training.	Detection speed is slow as compared to other single stage detectors.	Medium 80.37	Medium	High

The literature presents diverse techniques for vehicle detection in surveillance systems. Each paper proposes unique approaches with their strengths and limitations. Based on the literature's limitations, the proposed framework addresses the identified limitations to develop robust and efficient vehicle detection systems in urban zones.

PART 3

THEORETICAL BACKGROUND

3.1 MACHINE LEARNING DEFINITION

Automatic learning or machine learning techniques are frequently employed in object recognition. This computational discipline aims at developing autonomous systems capable of autonomous learning, ultimately leading towards the advancement of artificial intelligence. An inherent benefit is that complex tasks can be accomplished automatically, thereby minimizing human intervention.

The system learns, meaning it can recognize a complex set of patterns from data. All of this is done using a learning algorithm because a set of data is input and hypotheses can be constructed using a learning algorithm, allowing it to make predictions when new inputs are presented [31].

There are different types of machine learning:

- **Supervised Learning**: This type of learning involves feeding labelled training data into the system. Once training is complete, new data can be provided into the system in an unlabeled manner, as the system can recognize patterns during training to identify appropriate data. This type of learning is suitable for classification and regression problems [32,66].
- **Unsupervised Learning:** This machine learning form operates without needing labelled data as input. One prevalent challenge this method faces is grouping or clustering, whereby the system categories data into distinct groups based on similarities [32,66].
- **Learning through reinforcement:** The focus is on the system learning from experience. When the system learns, it is penalized for its mistakes,

encouraging it to learn to maximize rewards. That is comparable to teaching concepts or behaviors to children, where they are rewarded rather than punished for getting things right. The idea of reinforcement learning is very similar [32,66].

In this instance, we will employ the method of supervised learning. We aim to capture images labelled with objects we wish to identify to permit the system to learn their properties. Following the completion of training, we will initiate the recognition process, which entails validating that the system has accurately assimilated the objects it trained on.

3.2. DEEP LEARNING

3.2.1. Concept

Deep learning is a subfield of machine learning that involves an automatic learning paradigm aiming to model the way humans learn specific data and tasks [33]. Information processing in deep learning is executed through artificial neural networks, which are composed of numerous layers of simple processing units or neurons that transform information step by step. Technical jargon will be explained when first introduced. These layers include network input layers, hidden layers, and output layers. Later, we will explore the functionality of each layer, but for now, let us focus on the input layer, which serves as the neural network's information input. The hidden layer processes this information by breaking it down, while the output layer makes decisions based on the processing carried out by the previous layer [34,68].

3.2.2. Application

Deep learning is becoming ubiquitous in various industries. For instance, the medical field employs this technology to achieve more precise and definitive diagnoses [35], while finance utilizes models for predicting market behavior. Furthermore, deep learning is prevalent in self-driving cars, error detection, and computing [36].

The number of applications harnessing this technology is constantly growing, even within our immediate surroundings. For example, automated translation programmed like Google Translate can acquire knowledge from written texts for future translations [37]. Another option is speech recognition, frequently used in mobile phones, computers, and vehicles. Siri on Apple products is one example [38].

Additionally, some applications can recognize words or phrases with a mobile device's camera and automatically translate them [38]. Facial recognition can produce biometric security measures and allow users to apply filters to images captured by mobile phone cameras and add decorations such as masks or drawings to the human face [39].

These examples reflect only a few of the numerous applications currently being implemented and the expanding possibilities for utilizing this technology.

3.3. OBJECT DETECTION

The most prevalent machine learning problems are classified as "classification problems." The ability to classify specific data appropriately, based on system configuration parameters, depends on several variables [40].

For instance, a prominent example is the classification of images. Assuming we use supervised learning, different images of objects can be fed into the machine learning system. At least designate a name for each object of interest to facilitate training of the system [41]. Following this, we may demonstrate the system's ability to identify objects like those it has been trained on.

However, object recognition poses an advanced machine learning challenge beyond mere classification. This issue is more complex than simple classification as it involves determining the presence of an object in the image and its location within the image [41]. In image classification, it suffices for the object of interest to be present in the image merely.

However, in the present case, we seek to ascertain the object's presence and location within the image. In our case, object detection involves utilizing a pre-existing system and leveraging its knowledge to detect other object types or variations that the system can identify. To achieve enhanced detection performance, some adjustments will be necessary, which we will address later. To enhance our intuition, we may examine the problem of object recognition as individuals perceive it.

From a young age, pictures are visible to us and through repeated exposure to the objects depicted, we acquire knowledge about the items in our surroundings. Our brains can differentiate between various objects by recalling the learning instilled in us by our parents, either through object labelling or by observing the individuals within our environment. To successfully recognize an object, the eyes serve as the primary source of information, followed by the brain, which consists of neurons that process this information. Consistent repetition of this process allows for absorption and learning of the object's characteristics, ultimately facilitating future recognition of similar objects [42].

3.4. DESCRIPTION OF OBJECT DETECTION

In this section, we outline the components of an object detector and the training process necessary for it to function as intended. We meticulously explicate technical abbreviations upon their debut to guarantee clarity and avoid confusion.

The neural network serves as the foundation of deep learning-based object detectors. It conducts the requisite operations to extract image attributes so that object-related attributes are obtained, ultimately facilitating object recognition in each image.

3.4.1. Neural network

First, let us discuss the foundation of the object recognition system. It comprises a neural network that functions as its core.

Defining TLU (Threshold Logic Unit) or Threshold Logic Unit is imperative to comprehend neural networks.

A TLU is an artificial neuron that consists of inputs and an output. Assigning weights to the input, the TLU computes the weighted sum of the inputs to produce the output, as demonstrated in Figure 3.1. [43].

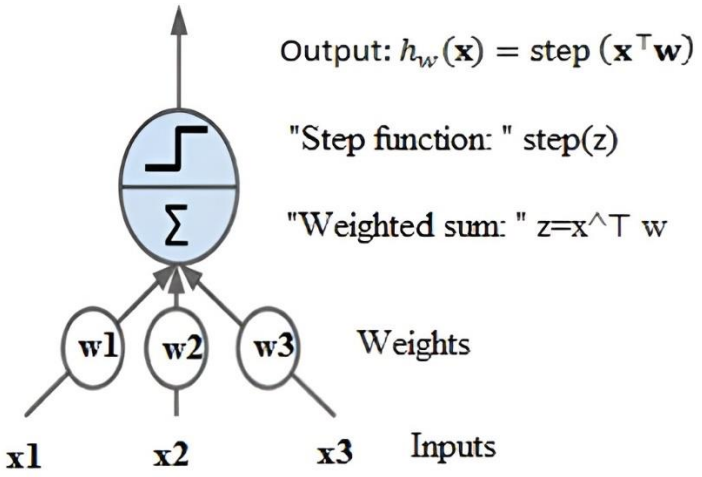

Figure 3.1. Structure of a TLU or threshold logical unit [43].

A perceptron comprises layers of TLUs, with each one being linked to every input. Furthermore, as demonstrated in Figure 3.2., a sequence of TLUs belong to the output layer and are also fully connected to its input [43].

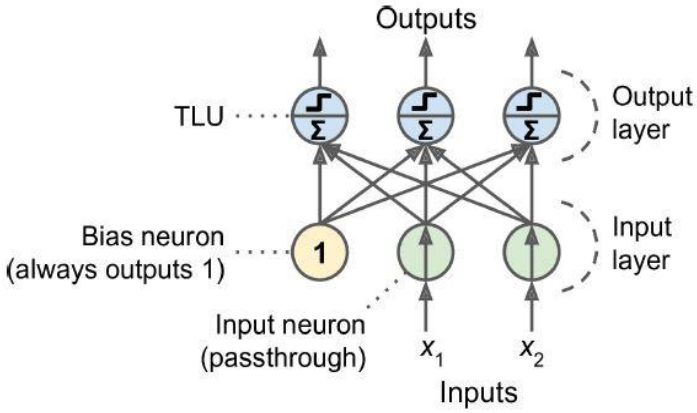

Figure 3.2. Composition of a perceptron [43].

Each Threshold Logic Unit (TLU) is triggered according to whether the weighted sum of inputs is higher or lower than a certain threshold. When the weighted sum surpasses the threshold, an output of value 1 is generated. In contrast, if it is lower than the threshold, the output is either zero or -1, depending on the type of activation function used [43]. Activation functions yield output values based on input values, which can range between 0 and 1 or -1 and 1, among other possibilities.

Finally, the Multilayer Perceptron (MLP) consists of an input layer, one or more hidden layers, and an output layer. The layers closest to the input are known as lower layers, while those closer to the output are called upper layers. This is further illustrated in Figure 3.3 [43].

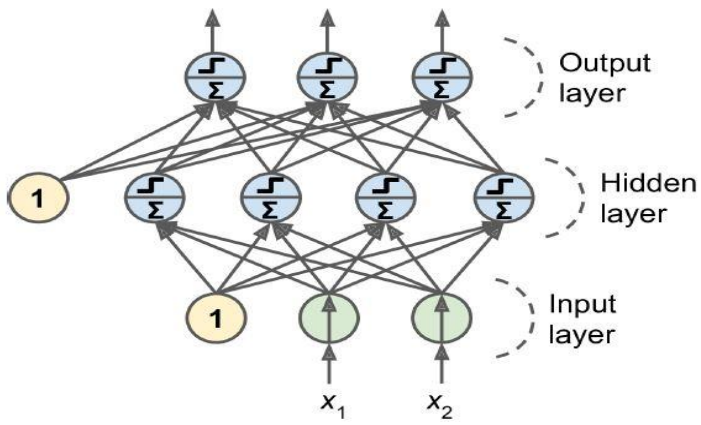

Figure 3.3. Structure of an MLP [43].

In Neural Networks for Object Detection, we will use Convolutional Neural Networks because the problem is with image processing. In this case, the network takes each image's pixels as input. In this case, in the first layer of the convolutional network, it is not connected to all the input image pixels, as we commented before, but to a part of it, the ones we see in Figure 3.4. [43].

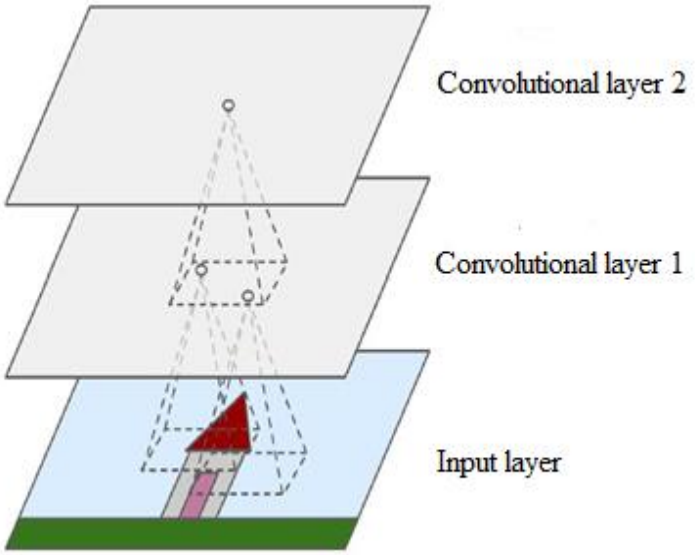

Figure 3.4. Scheme of how a convolutional neural network operates [43].

The figure 3.4. displays how the initial convolutional layer connects exclusively with a particular set of pixels enclosed by the bounding boxes in the input image. These bounding boxes correspond to the receptive field consisting of the pertinent pixels. As such, each neuron in the subsequent convolutional layer solely forms connections with neurons located inside the restricted rectangular area of the preceding layer.

This procedure captures minor features in the first hidden level and combines them into more prominent, higher-level features in the following hidden level, and so forth. The frequent occurrence of a hierarchical structure in authentic images has been found to enhance the performance of Convolutional Neural Networks (CNNs) in image recognition [43].

3.4.2. Filters

The neurons' weights can be depicted as small images of their respective receptive field size. An examination of filters using the example in Figure 3.5 reveals how two types of filters are applied. A neuron employing left filters disregards all its receptive area except for the central vertical line, since everything except the center line is multiplied by one. The same is observed when a central horizontal line is present in other cases [44].

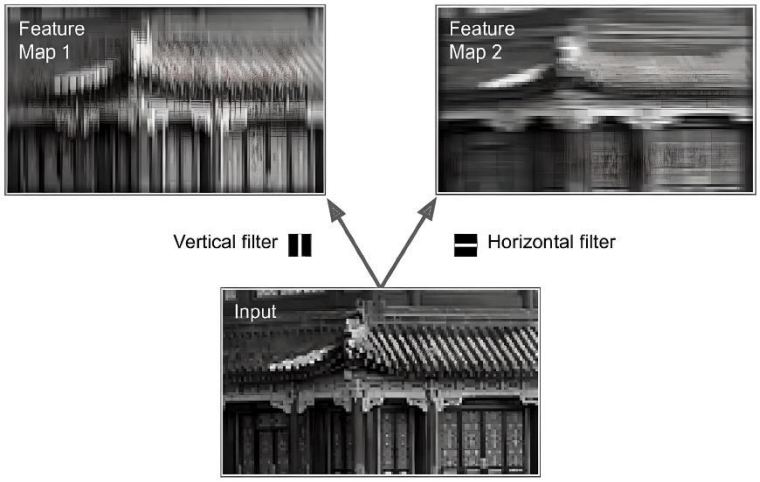

Figure 3.5. Example of applying filters to get two feature map [44].

In the first instance, we can observe the outcomes when all neurons of a particular level implement the same filter. This layer produces the image in the top left corner, evidenced in Figure 3.5. We can discern that the vertical lines are more refined whilst the remaining sections appear blurry. The reverse effect is observed for horizontal line filters. If all neurons in a layer employ the filter on the right, it is evident that the output of that layer is the upper right image, where the horizontal lines are more accentuated, while the remaining lines are more blurred. The outcome of the filter application is a feature map, which is the output of its use in the previous stage of the image [44].

The abovementioned process occurs at every level of all layers in the network. Typically, each convolutional layer utilizes several filters to produce feature maps for each one. With these maps, information regarding the objects present within the image is extracted and used to determine their attributes, thus identifying their composition [44]. Notably, "pooling" layers are commonly incorporated in image-processing networks of this nature.

3.4.3. Training process

We will describe the components of the training process, which is essential for comprehending how the detector functions and achieves object recognition through analyzing image content. Clarifying technical terms such as "detector" and

abbreviations such as "input" adds to the comprehension of the process. Training involves the system ingesting input data to gradually identify the constituent object within each input via iterative analysis of previously annotated images. It enables the system to identify the object it should recognize without human supervision automatically.

3.4.3.1. Input

The first step in annotating is data preparation. It should be noted that specific object detection networks, such as YOLO, are tailored to specific image sizes. This requires resizing the input image to meet the system's size requirements, which reduces computational costs.

Next, the following stage involves labeling the image to indicate the object's specific location in the picture. This step allows the system to learn and recognize the object accurately. Software can make the annotation of images easier; however, it remains time-consuming. Typically, a bounding box is necessary to locate an object within an image, with up to 500 boxes needed for a collection. Once labeled, each image is assigned an annotation file that includes the object's respective class, which is then utilized for training the system [45].

The dataset is initially divided into training, validation and testing, either by extensively labelling images or downloading a pre-prepared dataset. The training dataset provides the system with data during training, allowing the system to learn the properties of the objects to be recognized. The validation dataset allows us to monitor the training. It allows us to evaluate the model's fit while tuning the hyperparameters. Finally, the test dataset helps us to determine the system's performance, as it allows us to understand the effects of training and object detection. It also allows us to obtain performance metrics to evaluate the system [46] effectively.

3.4.3.2. Preprocessing

Before submitting the input to the system, the image must be modified. The first step involved is the normalization of the size and pixel values of the image. This procedure ensures optimal training of the system. This process adjusts the network to the picture size and computational performance. Subsequently, we eliminate the outliers and assign each pixel a value between zero and one.

Our study of the YOLO architecture found that the anchor boxes are loaded alongside the images and transformations. These anchor boxes contain a standard set of widths and heights used to match the sizes of objects within the dataset accurately [47].

3.4.3.3 Load the model-involved parameters and generate training

Parameters are needed to progress and implement training properly. First, we must identify training epochs. That represents the number of times we iterate through the entire dataset during training. For example, if we select 50 epochs, the training process will involve iterating through the training dataset 50 times. Additionally, we must consider the BatchSize, which denotes the quantity of images in each batch.

It is necessary to transmit this information to the network before modifying weights. If the BatchSize is set to 1/100 of the complete dataset, the weights will be altered 10 times during a single epoch. Another critical factor is the number of classes, or the variety of objects employed for training.

It would help if you also were mindful of the learning rate, which determines the size of the steps taken to achieve the optimal solution and its interpretation in conjunction with the cost function. The cost function measures the discrepancy between the predicted value produced by the model and the actual value. Therefore, when training, we aim to minimize the cost function by searching for values. Here, the learning rate becomes necessary because a high value implies that the minimum cost cannot be achieved, and it may even diverge, leading to an unattainable optimal solution. However, if the value of the learning rate is too low, it results in an extended time to

reach the optimal solution. However, employing smaller steps increases the probability of obtaining a solution that minimizes the cost function [48].

Next, it is essential to determine the neural network for detection and training. That can be achieved using various programming methods, such as declaring each layer individually. However, in some cases, the default network model can be loaded. Resnet, GoogleNet, and VGG are some examples of such networks, which can be trained from scratch by feeding input data directly into them. However, it is possible to use the knowledge that an existing network has gained from identifying a set of objects to teach it how to recognize new objects. This process is known as transfer learning [49].

However, it is possible to use the knowledge that an existing network has gained from identifying a set of objects to teach it how to recognize new objects. Suppose an AI system has become proficient at identifying items within a particular category. Yet, certain factors must be considered if we were to create a new AI system and use it to recognize objects in a different category. Currently, two options exist for the training phase: randomly assigning initial values to the weights and biases of the first layer in the new neural network or initializing these values with the weights and biases from the lower layers of the first network.

By adopting this approach, the network can avoid needing knowledge of all intricacies associated with the low level from the outset, thus diminishing the potential for extended training periods. The network only needs to acquire knowledge pertaining to the superstructure. Using this technique benefits the network by way of its expedited training process and reduced data requirements for effective training [46].

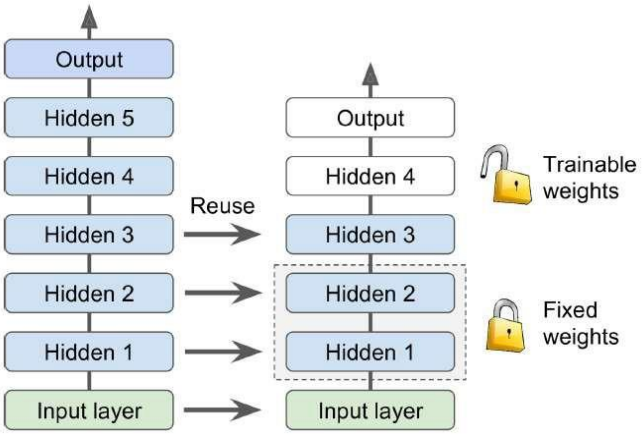

Figure 3.6. Visual example of the explanation of the learning transfer process [50].

Figure 3.6 shows how a novel network can be employed to serve a distinct purpose. In order to achieve this objective, a previously trained layer of another network that was developed for a disparate task is utilized. To achieve this, layers for the second task must be generated in the images and images of the trained network. Subsequently, the weights and biases of the trained network should be integrated into the new network. When the layers are trained with pre-trained weights, they become frozen, and the weights do not alter. The upper layers of the network are trained to make minor adjustments to enable recognition of other objects.

Therefore, when performing transfer learning training, we must load the pre-trained weights into the network we wish to train from scratch.

The subsequent step in configuring the training is to compile the model and define the loss function and optimizer to be used during the process. The optimizer enables adjustment of model parameters to reduce the cost function loss in the training dataset. In contrast, loss functions evaluate algorithm performance with the use of data.

After compiling the model, we can establish multiple additional parameters to enhance training management. One such parameter is the management of checkpoints or training control points, which store the precise values of all parameters utilized by the model. After a training epoch, a checkpoint is generated with the parameter values used for that epoch, providing tremendous advantages for later detection purposes [51].

Another essential parameter is Early Stopping. This parameter enables automatic cessation of training if the loss fails to decrease after a specified period or period. We can set it to monitor validation losses attained since the aim is to reduce them to attain the cost function's minimum. Hence, if these losses fail to decrease for consecutive epochs, it is inferred that the cost function is considered minimal, albeit not always the case, resulting in halting the training process.

Subsequently, after defining the prior parameters and performing the associated description task, we train to enable the neural network to learn the desired object properties [52].

3.4.3.4. Trained Object Detection

Finally, the procedure for identifying trained objects within the network involves loading the neural network with the adjusted weights obtained during training. Finally, the procedure for identifying trained objects within the network involves loading the neural network with the adjusted weights obtained during training. Subsequently, data, in the form of images containing the object in question (or lack thereof), is fed to the network to be recognized. Using the processes detailed previously, the system can accurately pinpoint objects present within the image. Therefore, it can delineate the location of the objects by drawing bounding boxes around them. Additionally, in the architecture to be introduced later, we can retrieve exact coordinates for the detected objects as well as determine their type, specifically the class of the object [53].

3.4.4. Performance Indicators

It is paramount that the detection system can identify the desired object with precision and accuracy. Thus, the system's capability must be evaluated using various metrics to analyze its performance.

Cross-validation is a fascinating method for assessing performance. The process involves dividing the training set into smaller training and validation sets. The model is then trained using each newly obtained training set, followed by application to the

validation set to assess resulting errors. Moreover, it provides an accuracy assessment to understand the functioning of indicators. This technique renders an estimation of model performance by averaging attained validation errors. For better comprehension, specific critical definitions need to be formulated [54].

• The Intersection over Union (IOU) is a metric used to determine the percentage of overlap between two bounding boxes. There are two types of bounding boxes: ground boxes, annotations used to train and validate object recognition systems on images, and frames obtained from recognition performed by the system, as shown in Figure 3.7. When the recognition and annotation bounding boxes match, the IOU equals 1. The lower the value, the less reliable the detection [55].

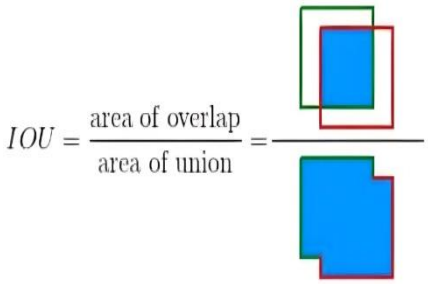

Figure 3.7. Graphical explanation of the intersection over the union [56].

Specifically, to know whether a recognition is good or not, we have the following concepts:-

- **True Positive** (TP): Occurs when an object is detected and IOU>=threshold. That is a correct detection.
- **False Positive** (FP): When the computer system detects an object, but when calculating IOU, it finds that IOU<Threshold. Therefore, this is a false detection.
- **False Negative** (FN): This happens when a ground frame is present in the image but not detected.

- **True Negative** (TN): Not used since these are all possible bounding boxes that do not contain objects and where no objects are detected. In object detection, an infinite number of possible bounding boxes should not be seen in an image.
- **Threshold:** The threshold set for object detection. Usually 50%, 75% or 95% [56].

3.4.4.1. ROC Curve

Receiver Operator Characteristic (ROC) curve: ROC is used to calculate the cost sensitivity in classification processes. This curve is generated by plotting the False Positive Rate (FPR) against the True Positive Rate (TPR) during anomaly detection. In this way, the performance of the algorithm used as a classifier is compared between error costs and class distributions. The area under the curve indicates the accuracy of the model estimation resulting from classification [67].

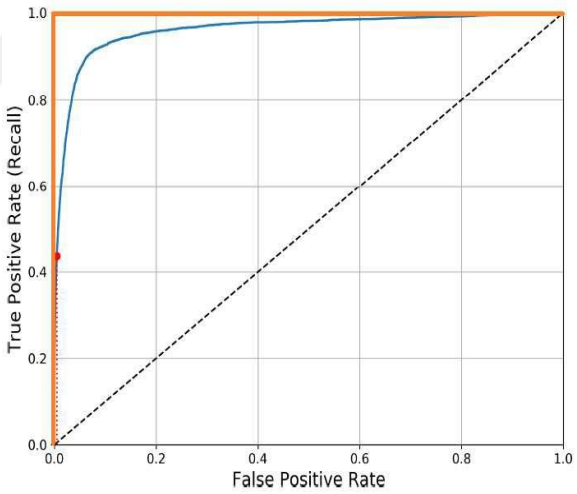

Figure 3.8. ROC Curve Example.

As shown in figure 3.8. The orange line corresponds to the ideal system, while the blue line corresponds to the real system. A good system will be as close to the prediction line as possible.

The orange line marks the ideal. If we look closely, we can see how to find a compromise. The higher the true positive rate (TPR), the false positive rate (FPR) appears in the system [57].

Calculating the area under the curve (AUC) is the most effective method to compare different systems. An ideal system would have an AUC of 1, while a real system would be below this value. However, the closer it is to 1, the better the system because it is close to the ideal state [57].

3.4.4.2. Precision and recall

Accuracy provides us with a measure of the quality of the proposed model. It indicates the proportion of all boxes returned by the computer system (i.e., all objects it found) that contained the object to be recognized. The ideal value is one. In Equation 3.1 we saw how to calculate it [58].

$$Precision = \frac{TP}{TP+FP} = \frac{TP}{all\ detections} \tag{3.1}$$

Recall or sensitivity is the proportion of objects that a detector finds that are present. We can say that a system is very sensitive if it can find all relevant cases, that is, if it finds all ground truths, i.e., all annotations present in the image [58]. In Equation 3.2 we see how is calculated.

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths} \tag{3.2}$$

3.4.4.3. Precision-Recall Curve

The Precision-recall concepts are closely related. In general, if we configure the system to achieve higher precision, it will result in lower recall, i.e., lower recall. H. The system becomes less sensitive, so fewer objects are detected. On the other hand, if we configure the system to have a higher sensitivity, we will find that the system is less precise because the system makes more lenient decisions when it comes to detection. Everything is managed.

From the threshold illustrated In Figure 3.9. we see graphically what is happening: as we increase the threshold, we see how the sensitivity decreases but the accuracy

increases. On the other hand, if we lower it, usually the opposite happens: higher sensitivity and worse accuracy [59].

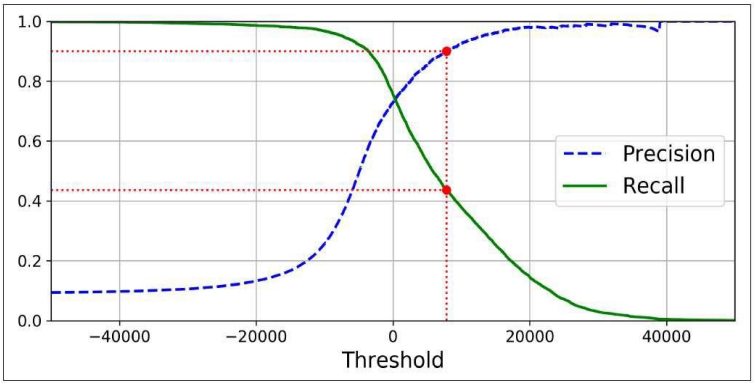

Figure 3.9. Illustration of the operation of the confidence threshold in the curves of Precision-Recall

If the precision remains high until the recall increases, we consider the detector to be good. We can see this in Figure 3.10. where we find a case for the ideal detector and another case for the real detector. The green line is an ideal line of how a perfect system should behave, while the blue line is an example of a real system [59].

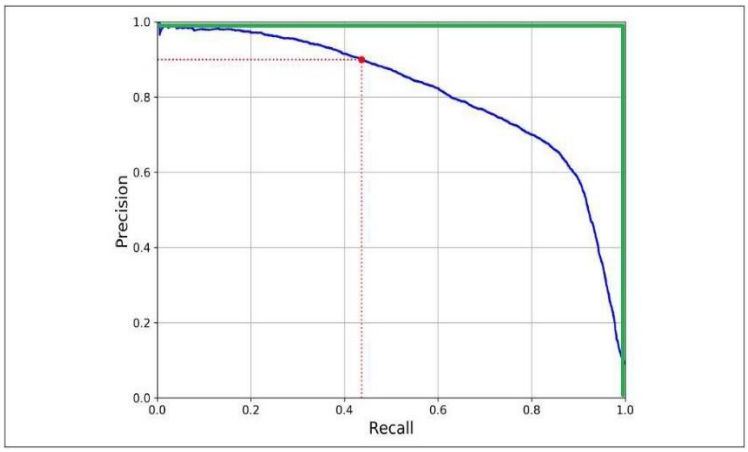

Figure 3.10. Example of an object detection system curve (blue) and the ideal curve to look for (green)

3.4.4.4. Mean Average Precision (mAP)

mAP (mean average precision) is an average value often used as a detection accuracy indicator in target detection. The mAP indicator is calculated by using an average

target to detect different AP (average precision) values corresponding to multiple targets in the task. The value of AP is the area used to accurately draw a PR curve based on the precision and recall in the experimental results obtained through predictive analysis. The calculation formulas of precision, recall and mAP indicators are as follows.

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

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

$$mean\ Average\ precision\ (mAP) = \frac{\sum_{i=1}^{k} A_{Pi}}{k}$$

In the formula, TP represents the number of both actual and detected positive; FP represents the number of detected positive, but actually negative; FN represents the number of detected negative, but actually positive. The accuracy is the ratio of the number of correct tests to the number of positive tests, that is, how many of the positive test results are correct. The recall rate represents the ratio of correct detections to the actual number of positive ones, that is, how many detections are correct during the detection process [60].

3.5. OBJECT DETECTION ARCHITECTURES

In this section, we discuss various object detection architectures that currently exist. Knowing how it works is crucial because it tells us different ways to make detectors.

3.5.1. R-CNN

The latest advancement in object recognition is R-CNN, which incorporates the Region Proposal Network (RPN) principle. The model identifies an area of interest in the image, usually the object's location. Once detected, a pre-trained network analyses the images and classifies the designated regions of interest (ROIs) [25].

The procedure entails generating latent bounding boxes in the picture in the first step, then using a classifier to examine these boxes in the second stage. After classification, post-processing is employed to streamline bounding boxes, enhance accuracy, reassess objects against one another in the image, and eliminate duplicated bounding boxes.

However, this model's major drawback is its sluggish speed. The initial algorithm operates on the regions, followed by classifier refinement and object detection, causing each stage to be time-consuming.

We attempted to enhance the model to address this issue and developed two new algorithms. The initial algorithm is refined by Fast R-CNN, which utilizes convolutional neural networks to extract functions that lead to slightly improved training and recognition times. However, there is no significant difference. The initial algorithm is refined by Fast R-CNN, which utilizes convolutional neural networks to extract functions that lead to slightly improved training and recognition times. Another algorithm, Faster R-CNN, bolsters processing speeds by integrating the previously mentioned region proposal tool into a convolutional neural network. The system also implements anchor boxes with pre-assigned height and width measurements for estimating the size of the target object during detection (further discussed later) [25].

3.5.2. Single Shot Detector (SSD)

The architecture of this detector follows a pyramidal structure in its convolutional neural network, enabling the detection of objects both large and small. Through multi-layer detection, various feature maps are generated, each with a unique scale.

Consequently, Convolutional Neural Networks decrease the spatial dimension and resolution of the map over time. Lower-resolution maps detect more prominent objects, while higher-resolution maps detect smaller objects.

The structure of the SSD network is illustrated in Figure 3.11 below, employing the VGG16 network. The procedure comprises extracting characteristics and detecting through convolutional layers [20].

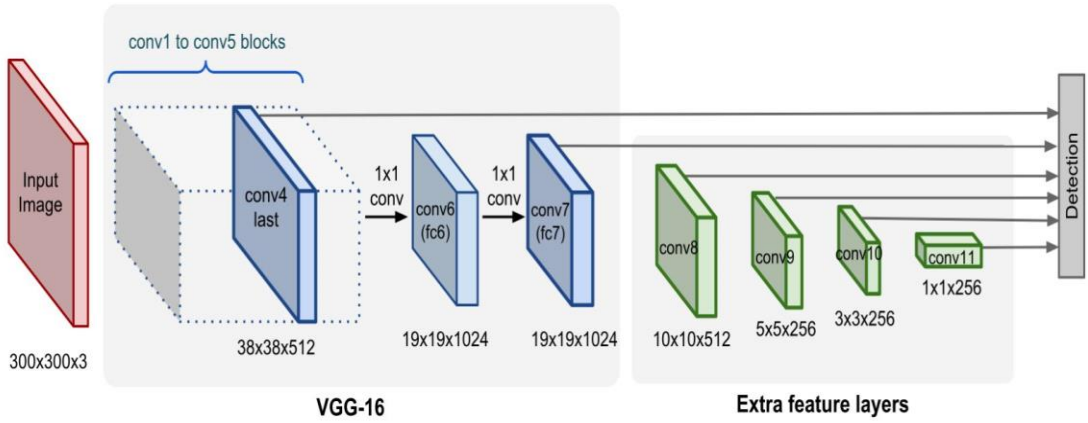

Figure 3.11. SSD network architecture.

Predictions consist of bounding boxes and N+1 class values, where N represents the number of identifiable classes, including classes that do not match any objects. It is worth noting that SSD does not depend on region proposals, unlike R-CNN. Instead, it obtains scores and locations by utilizing small convolutional filters after the extraction process of the feature map. Building upon the VGG16 architecture, this model incorporates six additional levels, with five specifically devoted to object detection. The precision of these multi-feature maps is significantly enhanced, which leads to a substantial improvement in speed. As a result, this model can be applied for real-time object detection [20].

3.5.3. Retina Network

This model is designed to identify objects of varying sizes, encompassing small and large objects. The development of this model was founded upon two significant advancements in the field of single-stage detectors, namely the feature pyramid network (FPN) and the focal loss.

The utilization of focal loss has been seen to enhance the class imbalance issue in single-stage object detection models. The focal loss is a loss function that diminishes the impact of accurately predicted values within a neural network. Thus, the focus is directed on instances in which the predicted class is inaccurate.

Pyramid Network (FPN) features are derived from the problem of detecting objects at different scales. A first approximation is to take images at different scales and obtain a feature map called an image pyramid from each scale.

It can be seen in Figure 3.12. on the left (a). However, it is computationally expensive and time-consuming [30].

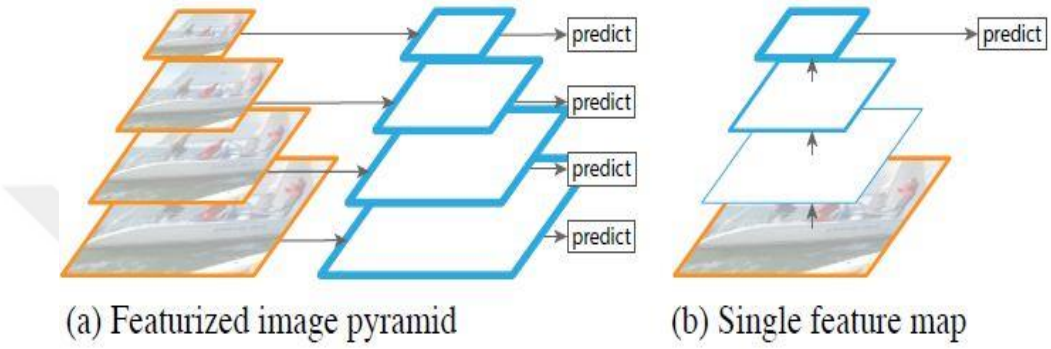

Figure 3.12. Difference between image pyramid (left) and feature map pyramid (right) [30].

Therefore, the approach is usually changed to do what is shown on the right side (b) in Figure 3.12: build a feature pyramid and use it for object recognition. Feature maps near the input plane (image plane) usually consist of low-level, ineffective structures in detecting objects. FPN was born, a feature extractor designed with a pyramid shape, so it has good accuracy and speed.

It has two workflows. Looking at it from the bottom up, this is the usual convolutional network we already know, applying filters to extract features. A top-down workflow creates layers with higher spatial resolution and helps to facilitate training and better predict locations [30].

In this way, we can better understand the RetinaNet architecture, which consists of four elements, as shown in Figure 3.13.

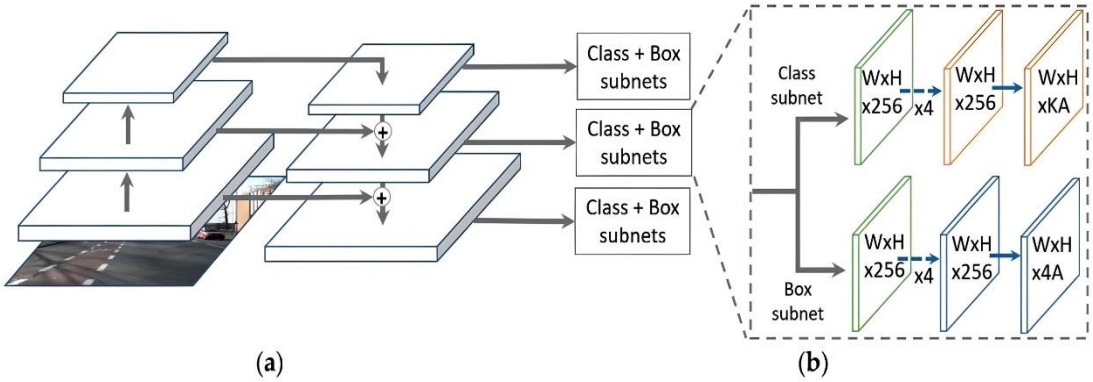

Figure 3.13. RetinaNet architecture [30].

In order from left to right, the first thing we find is the ascending path, that is, the backbone network that calculates feature maps of different scales. On the right, we see a descending path with lateral connections that sample feature maps that are spatially thicker than the top layers of the pyramid. In contrast, lateral connections merge layers from top to bottom and vice versa, with the same spatial size. Then, further to the right, we find the classification network, which predicts the probability that an object is or is not at a specific spatial location. Finally, we find the regression network, which returns the offset of the bounding box in the prediction relative to the ground truth [30].

3.6. YOLO

The object detection architecture, developed by Joseph Redmond as the primary author, is known for its exceptional speed and accuracy. Indeed, the architecture under consideration exhibits an impressive rate, rendering it conducive for real-time video detection. The YOLO (You only look once) framework encompasses a convolutional neural network that performs simultaneous predictions of the probabilities associated with numerous bounding boxes and the classification of objects contained within such bounding boxes [29].

3.6.1. Benefits

- Yolo is a high-speed system because detection is reduced to a linear regression problem, so no complex pipeline is required in testing; the neural network was run on images to make predictions [29].
- To make predictions, the system uses the entire image rather than just individual regions of the image, which limits errors in identifying object classes in images [29].
- Learning a generic representation of objects is less likely to fail when new input data is introduced than the above techniques [29].

3.6.2. Operation

Bounding boxes are delineated using the entire image and a convolutional neural network, as previously detailed. That enables the neural network to process and detect all elements within the image. As shown in Figure 3.14, the image is segregated into a grid with a size of S x S, where the central cell of this grid identifies objects that appear within its confines [61].

Conversely, every grid cell projects B bounding areas based on its confidence value. The confidence values reflect the model's reliability and the prediction's precision when an object is in the cell. To define trust formally, we use the equation Pr (Object)* IOU (as illustrated in Equation 3). When no object is in the cell, the confidence level should be 0. In the presence of an object in the cell, the confidence value should be Intersection over Union (IOU) [61].

Each bounding box has five predictions: x, y, w, h, and confidence. The x and y coordinates indicate the center of the field for the grid cell boundaries. The center of the green marker at coordinates (220,190), as shown in Figure 3.14, represents this location. When calculating it with reference to the unit size of 149×149, we must normalize it relative to the unit and maintain its value between 0 and 1, as illustrated in the calculation. The identical principle applies to the y coordinate.

The dimensions are proportionate to the entire image, meaning that the normalization process is based on the image size and not on the cells. To achieve this, the width (indicated by the red box in Figure 3.14) and height of the detected object are normalized based on the image size (448x448 in this case).

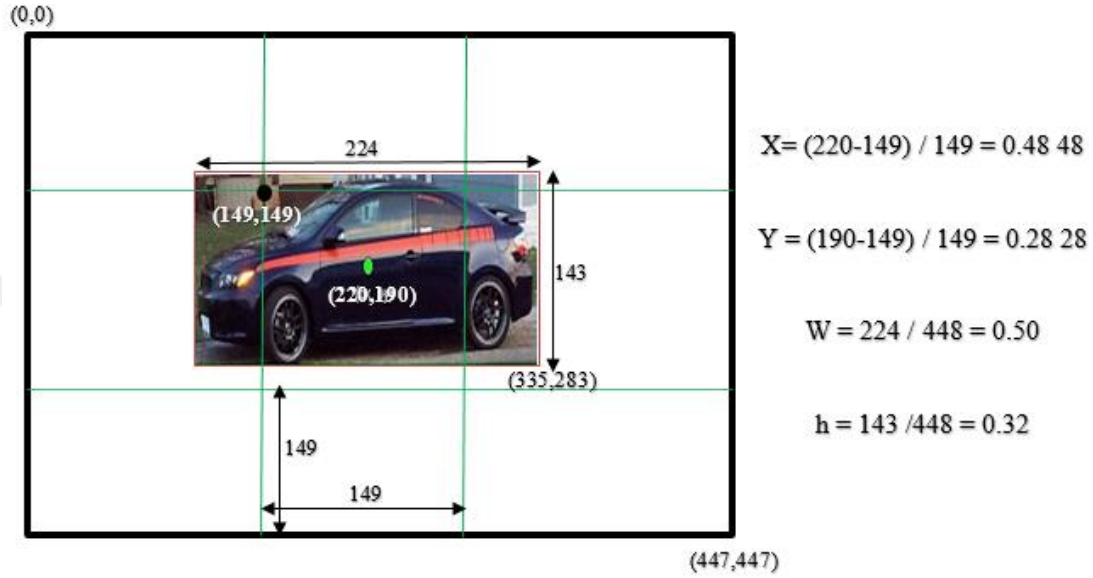

Figure 3.14. Example of how the coordinates of an image frame are calculated. The size is 448x448 pixels and S=3

Finally, to get a detection confidence value, we need several parameters. First, we need the IOU intercept of the bounding box and the annotation (Groundtruth) on the image, as shown in Figure 3.15. As we can see, we calculate the IOU (intercept of union) as the quotient of intersection points between unions. As a result, we get the intersection point.

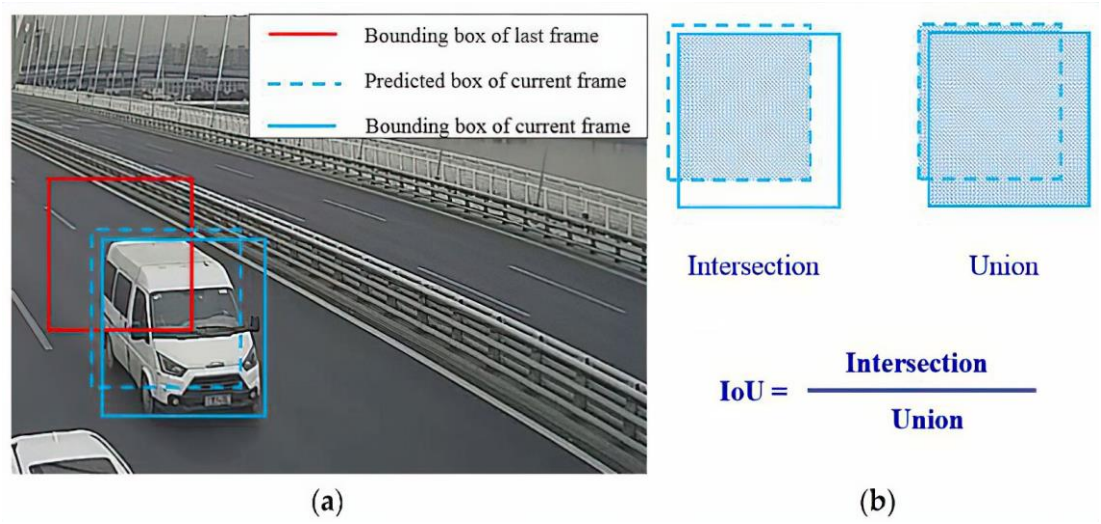

Figure 3.15. Graphic description of the intersection over the union [62].

A range of probabilities are utilized to determine a level of confidence. Each grid cell predicts the conditional class probability PR(class | object). The odds are reliant on the grid cell that holds the object. Furthermore, regardless of the number of bounding boxes, denoted as B, only one object is expected to be projected. When recognition is executed, the probabilities of each class conditional are multiplied by the probabilities linked to individual frames. As a result, specific class confidence ratings for each bounding box are obtained, enabling insights into the probability of an object's presence. The areas under scrutiny are the box's contents and how closely the expected box matches the object being investigated.

$$Pr\left(Class_i \mid Object\right) * Pr\left(Object\right) * IOU_{pred}^{truth} = Pr\left(Class_i\right) * IOU_{pred}^{truth} \quad \ldots. (3.3)$$

3.6.3. YOLO Network Architecture

The convolutional neural network has been employed to implement the model, commonly known as FCN or fully convolutional network, due to its convolutional layers. The first layer of the network extracts image features, and the fully connected layer is responsible for predicting output probabilities and coordinates. The inspiration for this network comes from the Google Net model, which is employed for image classification.

The network has a total of 24 convolutional layers and 2 fully connected layers. To decrease the number of layers, developers employ 1x1 convolutions, which can decrease the depth of characteristic maps. A 3x3 convolutional layer is executed, with 1x1 and 3x3 layers alternating, as depicted in Figure 3.16. In addition, the final convolutional layer produces a tensor of (7, 7, and 1024) dimensions. Two fully connected layers then reduce the tensor, resulting in a 7x7x30-sized tensor, as presented in Figure 3.16 [29].

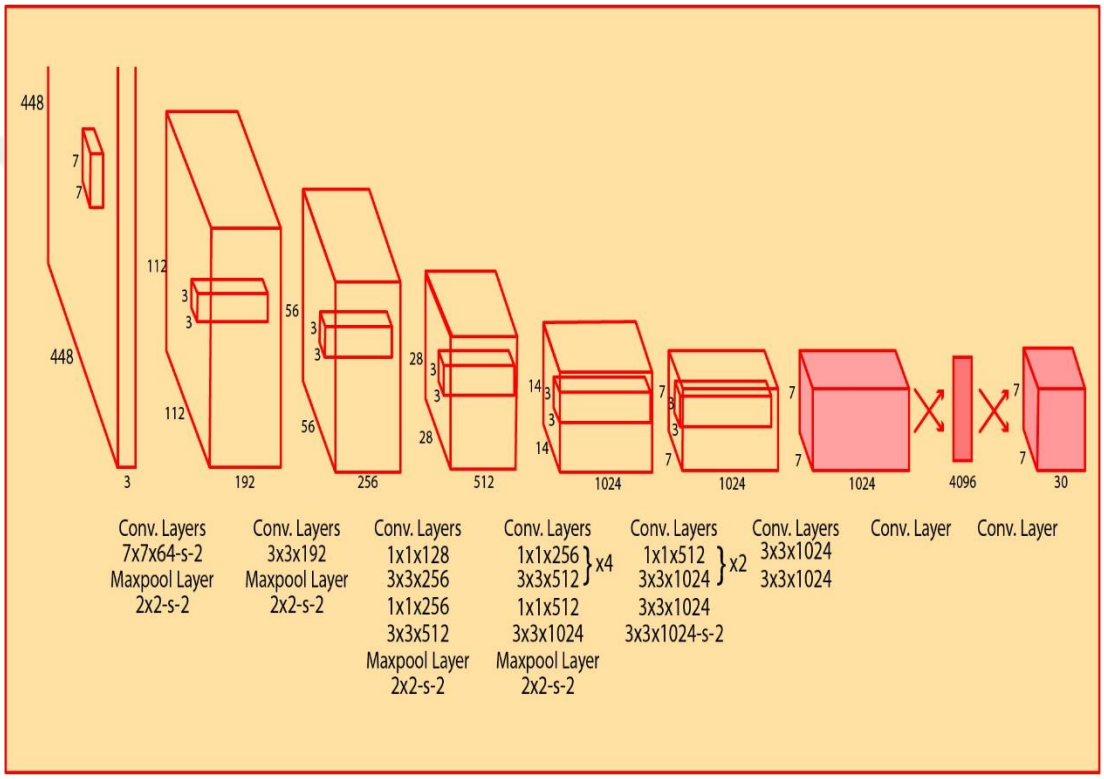

Figure 3.16. Schematic of YOLO neural network architecture [29].

The activation function used is LeakyReLu, except in linear layers where linear activation layers are used. In this case, the LeakyReLu function transforms the input values by multiplying negative values by coefficients to correct them, while positive coefficients do not correct them but remain unchanged [29].

As with the loss function, each unit in YOLO predicts many bounding boxes. Only one of all possible bounding boxes is considered to compute the loss for true positives, i.e., the one that gives the highest result with the annotation box (ground truth) when computing the intersection via IOU union [29].

To compute the loss, YOLO uses the sum of squared error (SSE) between the detected bounding box and the annotated ground box. Thus, the loss function consists of three elements [29].

The ranking loss is the error squared of the class-dependent probabilities of each class. It depends on whether the cell contains an object or not. If the cell is empty, the loss is zero. The expression for ranking loss in equation 3.4 is pi(c), where c is the class-dependent probability of c in cell i [29].

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in \, classes} (p_i(c) - \hat{p}_i(c))^2 \tag{3.4}$$

- Loss of position, the error between the measured position and the detected and annotated bounding box size. Compute when the bounding box of detection j is responsible for detecting the object in cell i. Otherwise zero. The expression for this loss is shown in Equation 3.5 [29].

$$\begin{aligned} &\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ &+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \end{aligned} \tag{3.5.}$$

Notably, here, one would not want to weight the error in a large bounding box equally with the error in a small bounding box, because the error for a given number of pixels in a large bounding box is different from the error in a small bounding box, and the error difference depending on the size is significant [29].

To avoid this, it calculates the square root of the bounding box width and height instead of applying them directly (no square root). For more emphasis on accuracy, the bounding boxes are also multiplied by the loss λcoord, which defaults to 5 [29].

• The confidence loss when an object is detected in a bounding box is given in Equation 3.6, where Ci is the confidence value for bounding box j in cell i.

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \tag{3.6}$$

And if there is no detected object, it is the one we have in Equation 3.7.

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2 \tag{3.7}$$

λnoobj refers to a factor that is placed because most bounding boxes do not contain any objects.

Finally, the final loss is the sum of all losses above, as shown in Equation 3.8. This feature to be minimized for training consists of the confidence drop indicated in the red box; the classification loss, shown in blue; and the site loss, shown in the green box.

$$\begin{aligned} &\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2\right] \\ &+\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right] \\ &+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\ &+\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\ &+\sum_{i=0}^{S^2} 1_i^{\text{obj}} \sum_{c \in \text{ classes}} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \tag{3.8}$$

3.6.4. Advantages of YOLO

Among the advantages of this model, firstly, we must deal with a high-speed system and find the main alternative to detect objects over time. Additionally, the detection function is performed by a single red neural device, which is fundamentally designed for maximum accuracy.

In contrast, YOLO's regional propaganda model takes the global view and finds almost no false positives [29].

3.7. DEVELOPMENT OF YOLO SYSTEM

We have roughly described the improved versions of the model that have emerged over the years since its inception in 2015. Along the way, we have shown the most exciting aspects of the system, or the most significant improvements made by each model revision. We have paid particular attention to the second optimized version (YOLOv2) because of our study baseline on this model.

3.7.1. YOlOv2

This release aims to enhance significantly accuracy and hasten detection. As for subtle enhancements, the inclusion of batch normalization in convolutional layers was executed [14].

Batch normalization is a process that includes supplementary operations into a model before or after the activation function of each hidden layer. As a result, each input undergoes normalization and is centered around the zero point. Ultimately, the results are modified by using two new sets of parameters for each step. This process enables the model to obtain the best dimensions and average values for each input level. Batch normalization is highly effective for deep neural networks as it can overcome the internal covariate shift problem. Normalizing the input process reduces the dependence of each layer on the previous ones, enabling faster and more consistent training. Additionally, batch normalization acts as a regularization method, reducing the need for other methods, such as dropout.

It is also worth noting that the classifier was enhanced; the initial iteration was trained on 224x224 images, while the updated version recognizes 448x448 images. In the updated version, the classifier is additionally trained on images of the last size to enhance the mean accuracy (mAP).

The use of anchor boxes is another noteworthy improvement, enabling the system to identify multiple objects per cell, instead of just one previously. This was problematic

before since it was believed that only one object could be detected in a cell with multiple objects.

Using anchor boxes, YOLO can detect numerous bounding boxes. An anchor box simply refers to a specific width and height that predicts bounding boxes for detections. This methodology allows for detection in relation to anchor boxes instead of predicting bounding boxes relative to the entire image. The shapes of the anchor boxes are selected automatically by YOLO, simplifying the network's object recognition process.

The K-Means algorithm is utilized in YOLO for unsupervised learning to determine the size [14].

The system is supplied with all available data, and we indicate the number of groups or "clusters" we wish to sort it into. Assuming we desire to divide our data into two groups or "clusters," we can first investigate the quantity of data in scenario a) displayed in Figure 3.17 for a more straightforward explanation. This algorithm assists in arranging data according to its properties. Assuming we desire to divide our data into two groups or "clusters," we can first investigate the quantity of data in scenario a) displayed in Figure 3.17 for a more straightforward explanation. Assuming we desire to divide our data into two groups or "clusters," we can first investigate the quantity of data in scenario a) displayed in Figure 3.17 for a more straightforward explanation.

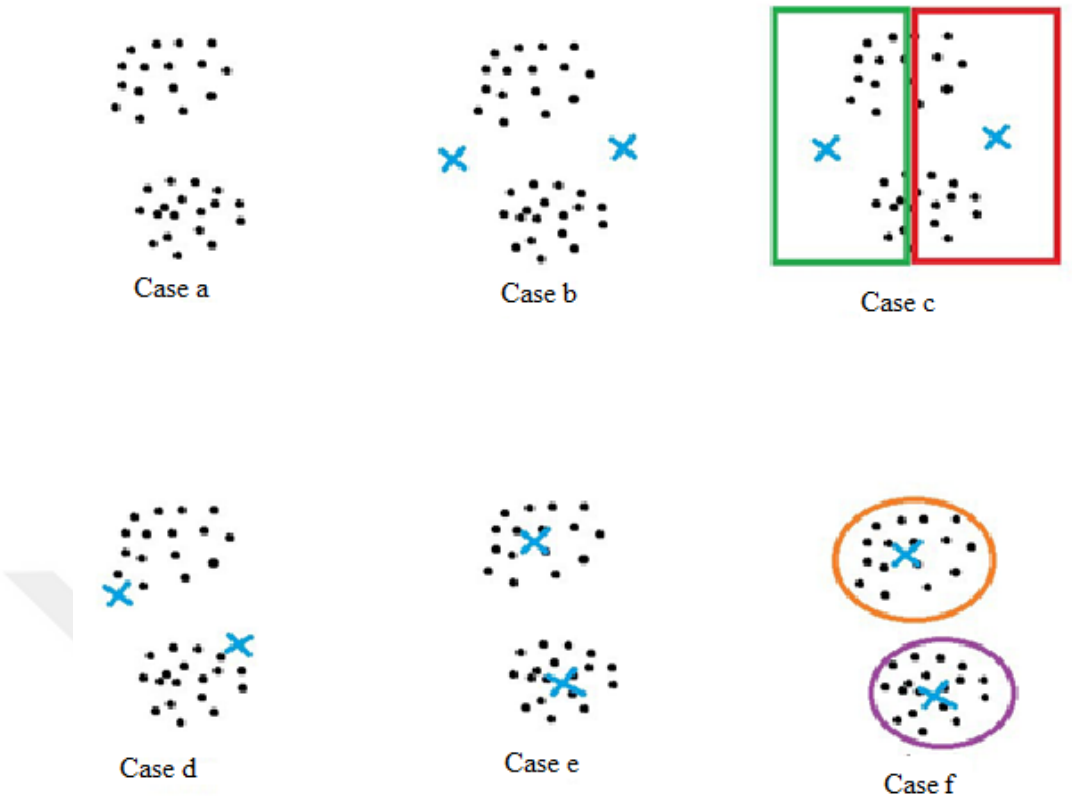

Figure 3.17. Illustration of the operation of the K-Means algorithm.

To partition the data into two clusters, the algorithm initializes two centroids randomly, as shown in case b) of Figure 3.17, and highlights them in blue. Subsequently, the algorithm calculates the mean of the points nearest to each centroid. In case c), we assume that the points closest to centroids 1 and 2 are highlighted by green and red boxes, respectively. After averaging the points closest to each centroid, the algorithm moves the centroid to that location, as demonstrated in case d. This is repeated for multiple iterations until reaching case e, whereby the algorithm identifies the point nearest to the first centroid as belonging to group number 1 (highlighted in orange) and the point closest to the second centroid as belonging to group number 2 (marked in purple), as illustrated in case f. The current methodology employs anchor boxes. However, the given scenario necessitates the execution of the described algorithm with multiple group values (k) to generate the average Intersection over Union (IOU) with the closest centroid.

The current iteration of the YOLO algorithm integrates the grid cells into predicting position coordinates. The GroundTruths values fall within the interval of 0 to 1. For

each cell, the neural network generates a maximum of five bounding boxes and estimates five sets of coordinates for each of them, namely, tx, ty, tw, th, and to. Equation 3.9 presents the outcome when the cell's position is displaced by cx and cy about the image's top-left corner, and the dimensions of the anchor box are denoted by pw and ph, respectively. Technical term abbreviations are explained when first used. For improved clarity, refer to Figure 3.18.

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_x} \\ \Pr(\text{object}) * \text{IOU}(b, \text{object}) &= \sigma(t_o) \end{aligned} \tag{3.9}$$

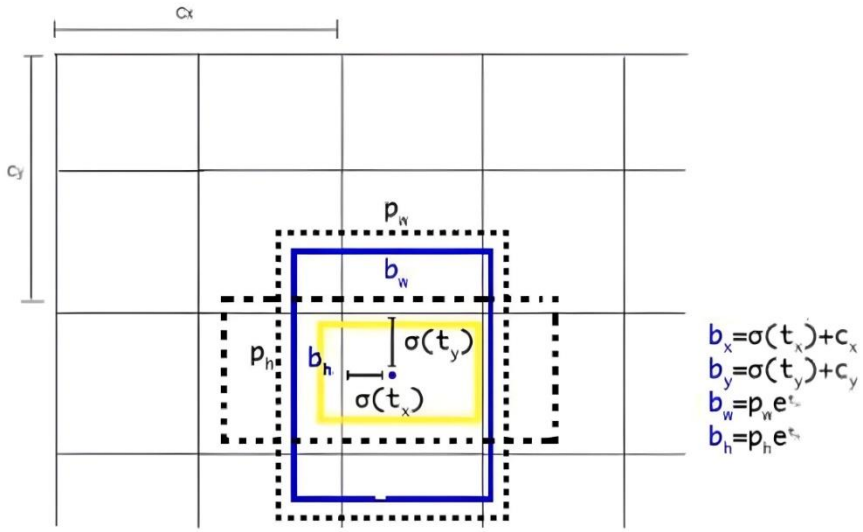

Figure 3.18. Intuitive explanation of the position of a bounding box [14].

YOLOv2 is more precise and intricate regarding network architecture because it introduces a novel classification model known as Darknet 19. The network comprises 19 convolution layers and 5 max-pooling layers, resulting in a notable increase in accuracy that outperforms YOLOv1 [14].

PART 4

METHODOLOGY

4.1. SCALE-INVARIANT CONVOLUTIONAL NEURAL NETWORK (SICNN)

Despite achieving near-human performance in various computer vision tasks, convolutional neural networks (CNN) have limited ability to tolerate scale variations. The traditional approach involves initially building a larger model and training it with data augmentation, including extensive scale jittering. CNN handles shift-variance much more effectively than scale-invariance [63]. Not being able to handle scale invariance directly contradicts the design philosophy of CNN, as higher layers may capture features of specific simple patterns simply because they are visible.

Filters are larger at the input, not because they are more complex. In other words, there is no alignment between the position of a filter and the complexity it captures. CNNs do not handle other invariances internally, such as rotations and flips, as they only capture certain features. Many natural objects are mostly symmetric, one solution to address scale variations of the same feature is to increase the network size by introducing more filters and apply scale-jittering to the input images, often by an order of magnitude. Such an approach is widely used today [63]. Proposals that directly deal with this problem include a method presented in [63], wherein crops of various sizes and positions drive the CNN at three distinct scales, followed using VLAD pooling to generate a feature summary of the patches.

In 2014, a group of researchers proposed a scale-invariant convolutional neural network (SiCNN) that extracts multi-scale features and classifies them into a network structure. SiCNN employs a multi-column architecture where each column is dedicated to a specific scale. Unlike previous multi-column techniques, the columns

share the same set of filter parameters through a scale transformation. This design effectively manages scale variation without inflating the model size. Experimental findings demonstrate that SiCNN identifies features at different scales and produces a classification outcome that provides significant robustness against object scale variations. Network (SiCNN) utilizes a multi-column framework, where each column concentrates on a specific scale. In contrast to the previous multi-column approach, these columns share the same collection of filter parameters using scale transformations among them. The model's dimensions remain constant despite changes in scale.

4.1.1. Scale-Invariance CNN Architecture

SiCNN employs multiple columns of convolutional stacks with fluctuating filter sizes to detect objects with unknown scales within input images. Figure 4.1 illustrates the architecture in which the input image is fed into all columns from bottom to top. Each column includes several convolutional layers with max pooling. The major distinction from traditional multi-column CNNs is that, despite utilizing different filter sizes, these columns still possess a standard set of parameters they share among their filters. Column 1 in Figure 4.1 maintains canonical filters within each layer. The remaining columns, called scale columns, convert these filters into their filters. Combined, a canonical filter and its transformed filters detect a pattern at various scales across multiple columns simultaneously. As a result, a unique pattern at different scales activates one or more columns [63].

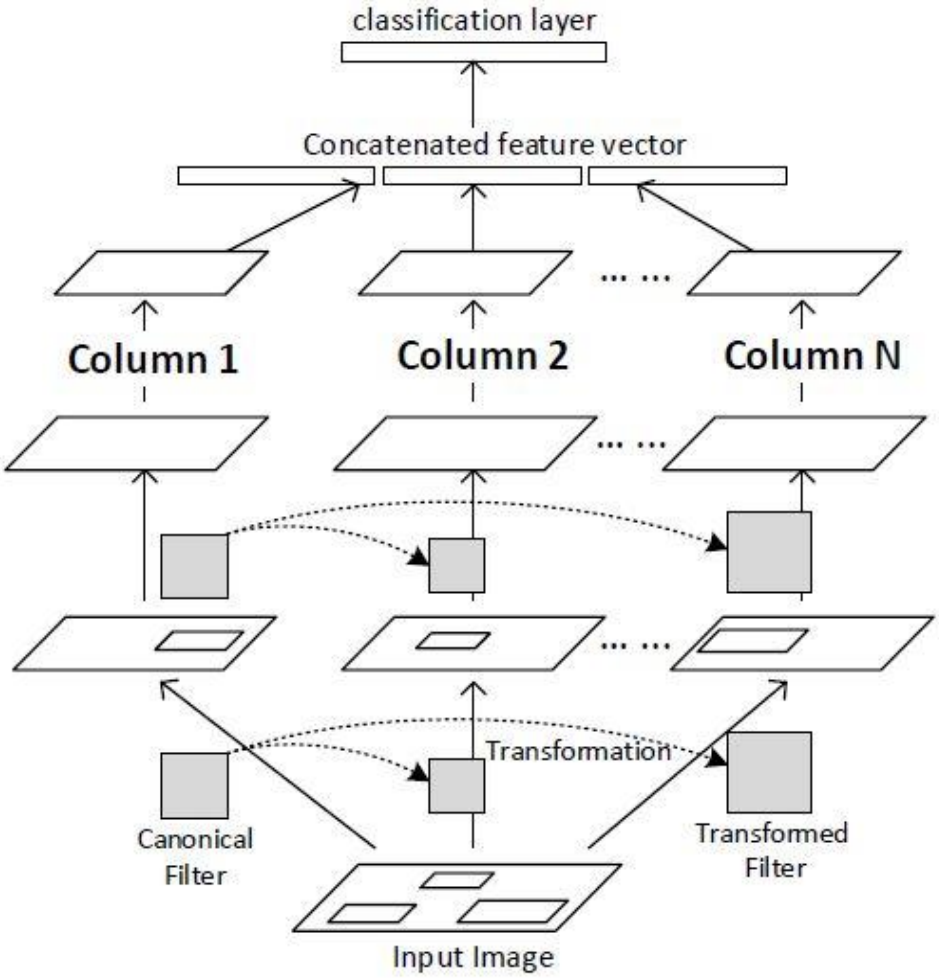

Figure 4.1. Architecture of SiCNN [63].

If the specification column I is obtained through input Ym the corresponding other columns S(I) are obtained through input S(Ym), which S refers to the scale operation.

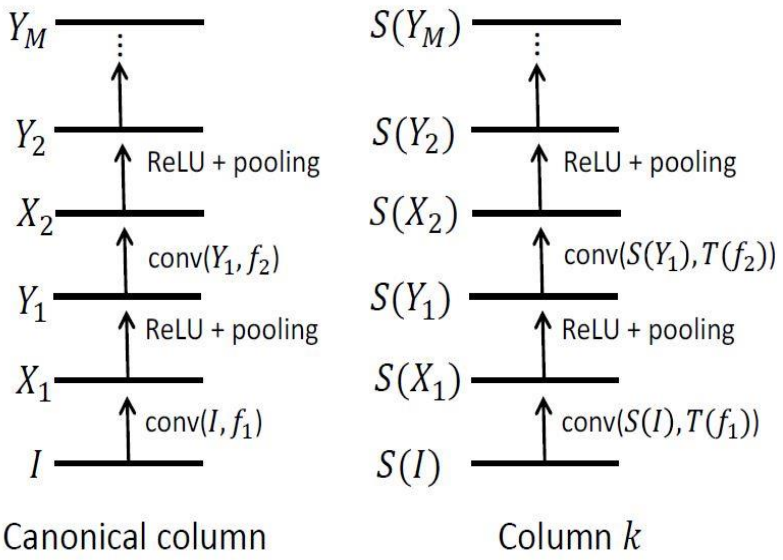

Figure 4.2. Columns with multiple layers to capture patterns in different scales [63].

The SIH-CNN model enhances the outcomes of conventional Convolutional Neural Networks (CNNs) and is a valuable addition to existing optimization strategies. The author's findings from the study are that the model acquires knowledge of features of varying sizes across distinct columns. The generalizable idea can be applied in all aspects where CNN is employed, including supervised and unsupervised learning, recognition, detection, and localization tasks. The first findings of the authors' investigation suggest a favorable balance between performance and the expense associated with training.

There are several unresolved issues. An alternative approach to summarize all the columns, rather than concatenation, is a distinct method. Additionally, the connectivity structure among the columns may be modified by implementing pair-wise connections between columns instead of a singular connection to the canonical column [63].

4.2. VEHICLE DETECTION MECHANISM

The vehicle detection methodology utilizes the YOLO v2 object detector's capabilities. The YOLO v2 detector operates through a unified neural network framework that concurrently handles the responsibilities of vehicle categorization and localization. The localization concept involves approximating bounding boxes that determine a

vehicle's precise dimensions and location within a specific frame. In this task's context, categorization involves identifying the specific type of vehicle detected, which can vary between cars, buses, or vans.

The initial stage of the process involves partitioning the supplied video frames into grid cells of dimensions 13x13. This phase aims to enhance the detection process by dividing the entire frame into more manageable portions. Every individual grid cell is tasked with detecting automobiles, contingent upon the need that the central point of the vehicle lies within the confines of that specific cell.

Subsequently, each of the 169 cells comprising the 13x13 grid assesses two essential attributes: confidence ratings and bounding box parameters. The confidence score is a comprehensive statistic that indicates the likelihood of a vehicle being present within a specified bounding box. The score is determined by the product of the probability of a vehicle's presence within a cell and the Intersection over Union (IoU) value between the predicted bounding box and the ground truth. In the present context, "ground truth" refers to the precise and verifiable depiction of the vehicle's position and size within the image. A confidence score of 0 signifies the absence of a vehicle within the designated cell.

In contrast, when the model identifies a vehicle within a bounding box, it assigns a confidence score based on the Intersection over Union (IoU) between the predicted bounding box and the ground truth. This procedure guarantees that the model's predictions closely align with actual observations.

Each bounding box within the grid predicts five essential components to accomplish the vehicle detection task. The components encompass the x and y coordinates of the vehicle's center to the grid cell, the previously mentioned confidence score, and the predicted height and width of the vehicle. By making predictions on these five components, the model fully depicts the possible position and magnitude of the vehicle within each cell.

The graphical representation of the vehicle detection method is depicted in Figure 4.3, which showcases the sequential steps followed by the model to achieve accurate vehicle detection. The methodology offers a promising solution for real-time vehicle recognition in video frames by leveraging the strengths of the YOLO v2 detector and a distinctive grid-based approach. This combination ensures both efficiency and accuracy in the detection process.

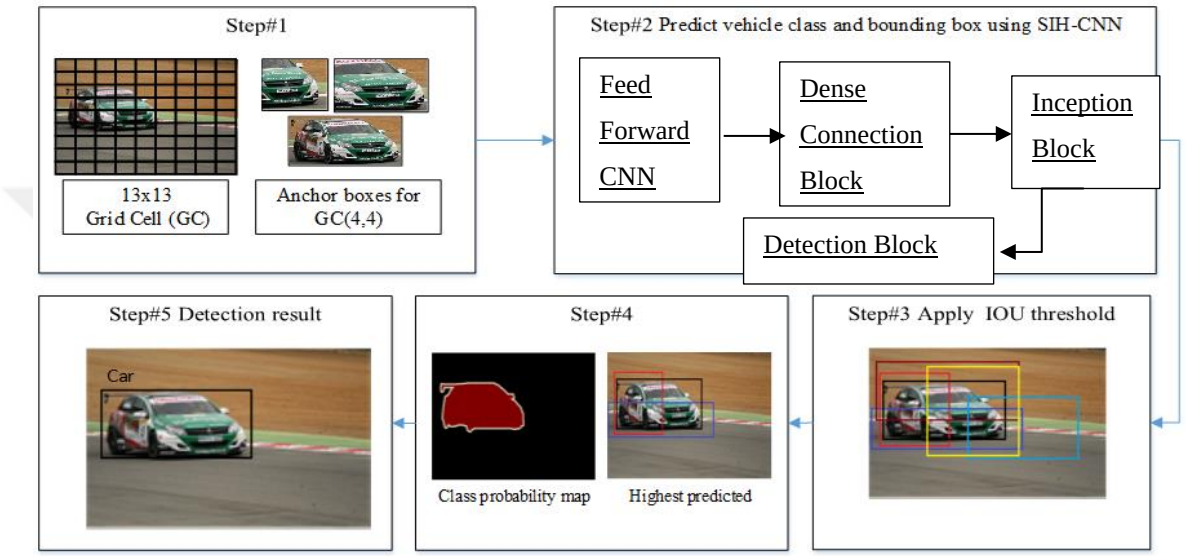

Figure 4.3. SIH Vehicle Detection Mechanism [69]

4.3. SCALE INVARIANT HYBRID CONVOLUTIONAL NEURAL NETWORK VEHICLE DETECTION STEPS

Step 1: Pre-processing the Input Video Frame

- Capture a frame from the surveillance video feed.
- Resize the frame to fit the input dimensions required by the YOLO model, maintaining the aspect ratio.

Step 2: Grid Division

- Divide the processed frame into a 13x13 grid, following prior evidence supporting the effectiveness of this grid size for balancing granularity and computational efficiency [64].

Step 3: Feature Extraction

- Pass the input frame through the SIH-CNN architecture, designed to be scale-invariant.
- Extract features and object descriptors that are useful for object localization and classification.

Step 4: Bounding Box and Confidence Score Prediction

- Predict bounding boxes and associated confidence scores for each cell in the 13x13 grid.
- The confidence score (Pr(Vehicle) * IoU_{Predicted}^{Ground_Truth}) serves as an indicator of the detection's reliability.

Step 5: Object Centroid Identification

- Determine whether the centroid of a detected vehicle falls within a given grid cell.
- The grid cell containing the object's centroid becomes responsible for the detection of that vehicle.

Step 6: Non-Maximum Suppression

- Eliminate multiple bounding boxes around the same vehicle by applying non-maximum suppression based on their confidence scores and overlap (IoU).

Step 7: Final Detection

- Retain bounding boxes with confidence scores above a certain threshold (e.g., 0.5) as final detections.

- Each retained bounding box will predict the vehicle's center (x and y coordinates), height, and width, along with the confidence score.

Step 8: Post-processing and Output

- Convert the bounding box coordinates back to the dimensions of the original frame.
- Overlay the bounding boxes and confidence scores on the original video frame for visual verification.

Step 9: Continuous Monitoring

- Move to the next video frame and repeat Steps 1 to 8 for continuous vehicle monitoring.

Step 10: Real-Time Performance Measurement

- Evaluate the model's performance in real-time, ensuring that it maintains the desired frame rate (e.g., 48.4 FPS as per our SIH-CNN architecture).

The vehicle detection process is visually demonstrated in Fig 4.4., which provides an overview of the SIH vehicle detection mechanism.

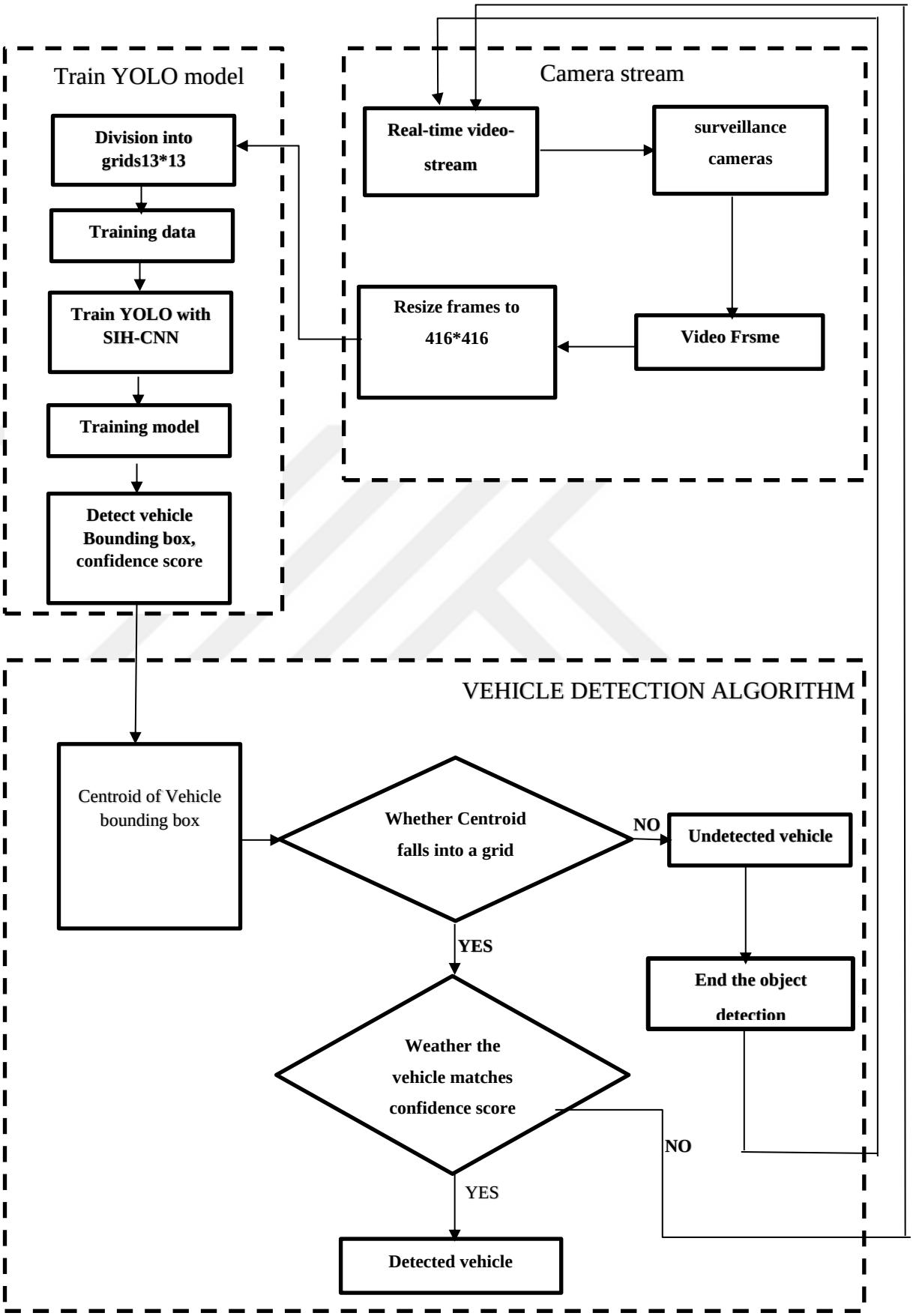

Figure 4.4. Propsed Framework algorithm flowchart.

4.4. SCALE INVARIANT HYBRID CONVOLUTIONAL NEURAL NETWORK (SIH-CNN) ARCHITECTURE

This work aims to rectify the deficiency in the existing Darknet-19 backbone design. To achieve this objective, a new SIH-CNN backbone architecture is proposed. As depicted in Figure 4.5, the SIH-CNN architecture initiates the process by reducing the size of the convolutional structure to extract features by utilizing numerous convolutional and max-pooling units.

After receiving an input image with dimensions of 416x416, the architecture initially applies Convolution_1. This operation utilizes 32 filters of size 3x3, with a stride value of 1. Afterwards, a max-pooling layer is implemented, employing a 2x2 filter and a stride (S) value of 2, which yields feature maps with dimensions of 208x208x32. The subsequent units in the architecture adhere to the same max-pooling value.

The second unit subsequently executes Convolution_2, employing 64 kernels with dimensions of 3x3. This results in feature maps measuring 104x104x64 after the second max-pooling process. The third unit utilizes three convolutional layers, namely Convolution_3 (3x3), Convolution_4 (1x1), and Convolution_5 (3x3), with 128, 64, and 128 kernels accordingly. After the third max-pooling process, the outcome is a total of 52x52x128 feature maps.

Subsequently, the fourth unit of the model integrates three convolution layers, each utilizing 256, 128, and 256 kernels accordingly. The fourth max-pooling operation yields feature maps with dimensions of 26x26x256. Following the completion of this unit, the architectural design yields feature maps that exhibit enhanced robustness.

To enhance the feature extraction process and address the issue of gradient vanishing during backpropagation, a Dense Connection Block (DCB) is proposed at this juncture. In this architectural arrangement, the feature maps originating from the L-1 Layer are merged with the feature maps from the current Layer L and the subsequent Layer L+1.

Nevertheless, incorporating an excessive number of DC-convolution layers inside the convolutional neural network architecture can impede detection speed and introduce complexity to the model. Therefore, the SIH-CNN model integrates the DCB within the penultimate convolutional block to extract the most semantically meaningful features. The dense connection module of the proposed SIH-CNN architecture consists of four DC units, each comprising a 1x1 and a 3x3 Conv-layer. Each DC unit has a batch normalization (BN) layer before its 3x3 convolutional layer.

By the Deep Convolutional Block (DCB), the Spatial Invariant Hierarchical Convolutional Neural Network (SIH-CNN) employs an inception block. To extract multi-scale features, the feature maps from the previous layer undergo processing using three different dimension filters (1x1, 3x3, and 5x5) and one initial max-pooling filter.

The multi-scale feature maps are then integrated to create a robust framework ideal for identifying small vehicles.

By utilizing a convolutional neural network (CNN) architecture that performs convolutions at a singular layer, the network displays greater width without a proportional increase in depth. This attribute enhances the feature extraction process across different scales. The last convolutional layer receives the feature maps generated by the inception block.

To mitigate overfitting and reduce the number of parameters, a transition has been introduced from the flatten layer to the global pooling layer. This advanced SIH-CNN architecture aims to address the limitations of its precursor, Darknet-19, by improving the accuracy and effectiveness of object detection [69]

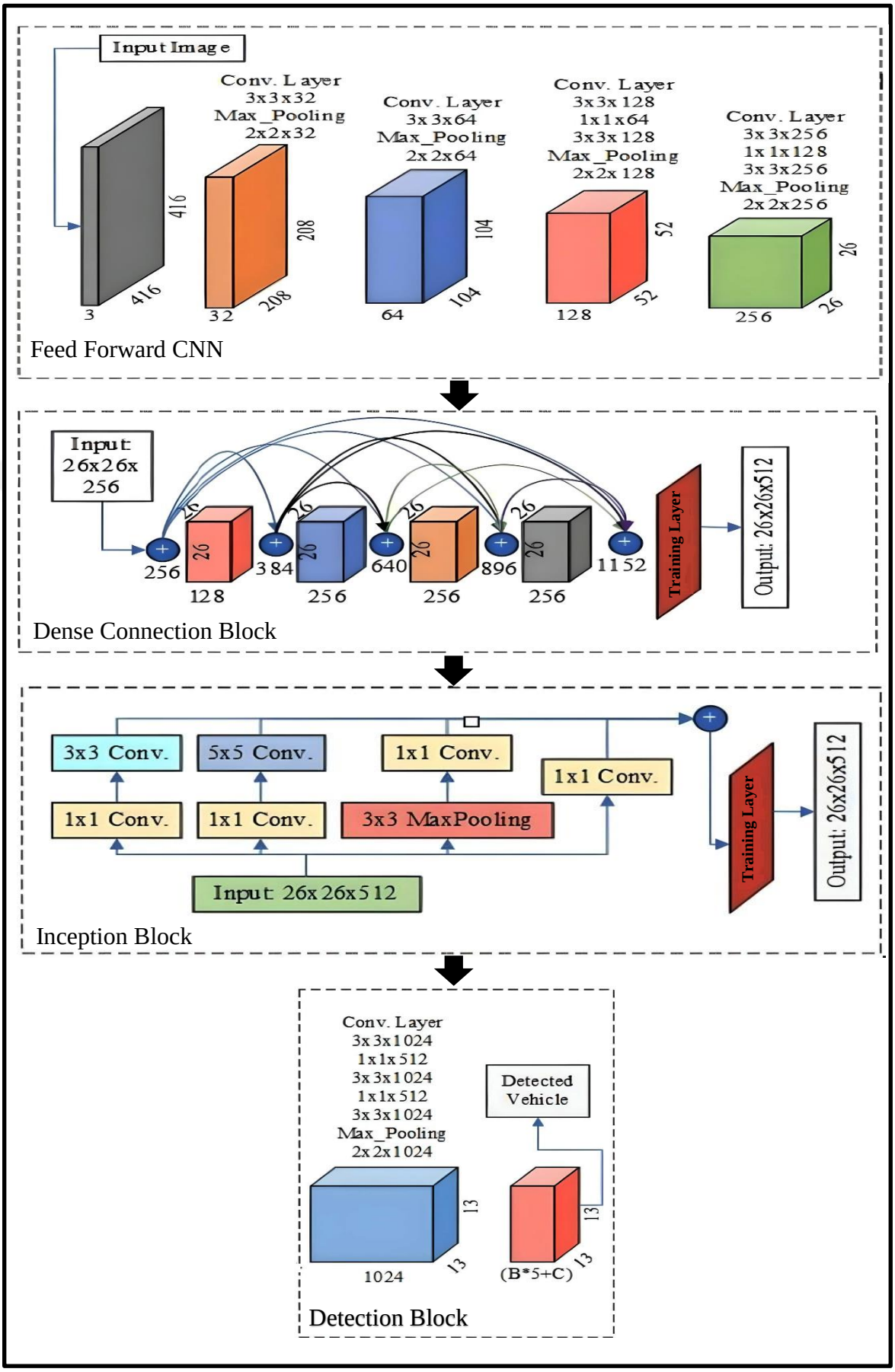

Figure 4.5. Proposed SIH-CNN Backbone Architecture [69]

PART 5

EXPERIMENT STUDY

5.1. SIMULATION FRAMEWORKS

All experiments were run on a computer with on Intel(R) Core(TM) i7-7700K CPU, 4.50 GHz Max Turbo Frequency, and NVIDIA Titan X GPU with 12.00 GB memory.

5.1.1. Model Training

The SIH-CNN model utilizes Stochastic Gradient Descent (SGD) as its optimization algorithm. Stochastic Gradient Descent (SGD) refines the weights of a model by utilizing the gradient of the loss function to those weights. The stochastic gradient descent (SGD) algorithm exhibits sensitivity to the initial learning rate configurations. Hence, a learning rate of 0.0001 has been chosen. The preference is for selecting a smaller number since it facilitates consistent convergence and reduces the likelihood of going beyond the optimal solution point.

The SIH-CNN model incorporates the Leaky Rectified Linear Unit (Leaky ReLU) activation function. Leaky ReLU is like the traditional Rectified Linear Unit (ReLU) in terms of its functionality. However, it distinguishes itself by maintaining a small non-zero gradient for negative input values. This characteristic prevents any neuron from becoming inactive throughout the learning process. In addition to mitigating the issue of the vanishing gradient problem, incorporating this technique can enhance the model's capacity to acquire knowledge. The batch size chosen for training the SIH-CNN model is 32. The quantity of training examples handled in each iteration is a determining factor that impacts the speed and stability of the training process. A greater batch size is associated with accelerated convergence rates, albeit at the

expense of increased memory requirements. Conversely, a smaller batch size promotes more stable convergence, albeit at a slower pace.

5.1.2. Benchmark Acquisition

To evaluate the efficacy of a computer vision-driven system for detecting vehicles. It is usual practice to employ a publicly available benchmark dataset. To conduct this investigation, the researchers used the UA-DETRAC benchmark [65] as it offers a diverse collection of real-world traffic video sequences that can effectively test and evaluate the capabilities of the proposed system.

The UA-DETRAC benchmark dataset comprises a comprehensive collection of 1.21 million bounding boxes derived from a diverse set of 8,250 unique cars. The scope of this study encompasses four distinct categories of vehicles, specifically automobiles, buses, vans, and a miscellaneous group that includes various sorts of vehicles [65].

To provide comprehensive and rigorous testing, the training phase of the system employed 80% of the benchmark, while the remaining 20% was allocated for testing purposes.

This benchmark presents vehicles categorized into three distinct sizes: little (less than 50 pixels), medium (50-150 pixels), and gigantic (more than 150 pixels). The performance of the proposed system may be assessed across a diverse variety of vehicle types and sizes due to the availability of this vast scale range.

Providing ground truth data by the UA-DETRAC benchmark is of utmost importance. Based on the provided data, it is possible to compute performance metrics that effectively measure the system's efficiency to its real-world performance. Figures 5.1 and 5.2 illustrate a subset of samples extracted from the UA-DETRAC benchmark dataset. These figures showcase a diverse variety of vehicle types and sizes and various environmental conditions represented within the dataset.

The evaluation of the system's capability to reliably detect cars in real-world circumstances is facilitated by testing the proposed system against a robust benchmark dataset.

These situations represent a comprehensive evaluation of the system's effectiveness, and the ability to detect in such circumstances accurately serves as evidence of the potential of the suggested Convolutional Neural Network (CNN) structure in real-world implementations [65].

Figure 5.1. Sample annotated frames in the UA-DETRAC [69].

Figure 5.2. UA-DETRAC detection sample In different environments [65].

5.1.3. Evaluation Metrics

The mean Average Precision (mAP) is an often-utilized metric in the evaluation of object detection systems, particularly those developed explicitly to detect vehicles. The detection algorithm's precision is evaluated by calculating the average precision over all categories of items and scales. In this work, the evaluation of the proposed vehicle detector was conducted by employing the mean average precision (mAP) metric. The performance evaluation was performed using the UA-DETRAC benchmark, which provided precise annotations for vehicle detection. Figure 5.3 depicts a representative image sourced from the UA-DETRAC dataset, showcasing instances of False Positive (FP), False Negative (FN), and True Positive (TP) occurrences utilized in the evaluation of performance.

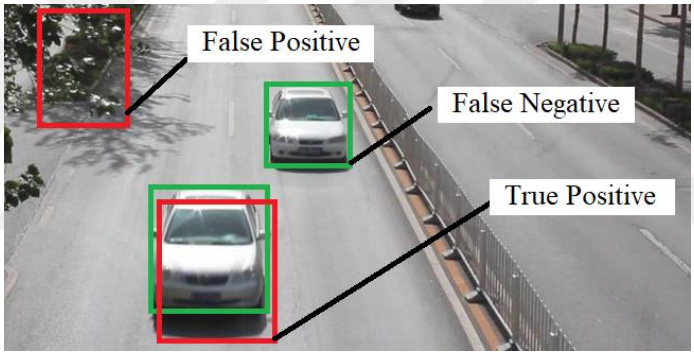

Figure 5.3. UA-DETRAC detection sample.

To calculate mAP, we used a set of performance evaluation matrices represented in equations (4.1.) - (4.4.).

$$\text{Precision} = \frac{TP}{TP+FP} \tag{4.1}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4.2}$$

$$\text{Average precision (AP)}: = \int_0^1 P(r)\,dr = \frac{1}{11}\sum_{rec=0,0.1,0.2,\ldots,1} Pinterp(rec) \tag{4.3}$$

$$mean\ Average\ precision\ (mAP) = \frac{\sum_{i=1}^{k} A_{Pi}}{k} \tag{4.4}$$

$$Frame\ Per\ Secon\ d\ (FPS) = Number\ of\ frames Duration\ in\ seconds \quad (4.5.)$$

5.2. RESULT AND DISCUSSION

5.2.1. Comparative Evaluation of The Proposed SIH-CNN

- Introduction to the comparative models: The performance evaluation of the proposed SIH-CNN model was conducted by comparing its results with other well-known detection models, including (Faster R-CNN , SSD , YOLO v2) [27,20,14].
- Figure 5.4. These figures visually demonstrate SIH-CNN's comparison against other models based on the AUC (area under the curve) derived from the Precision-Recall curve for different vehicle classes.

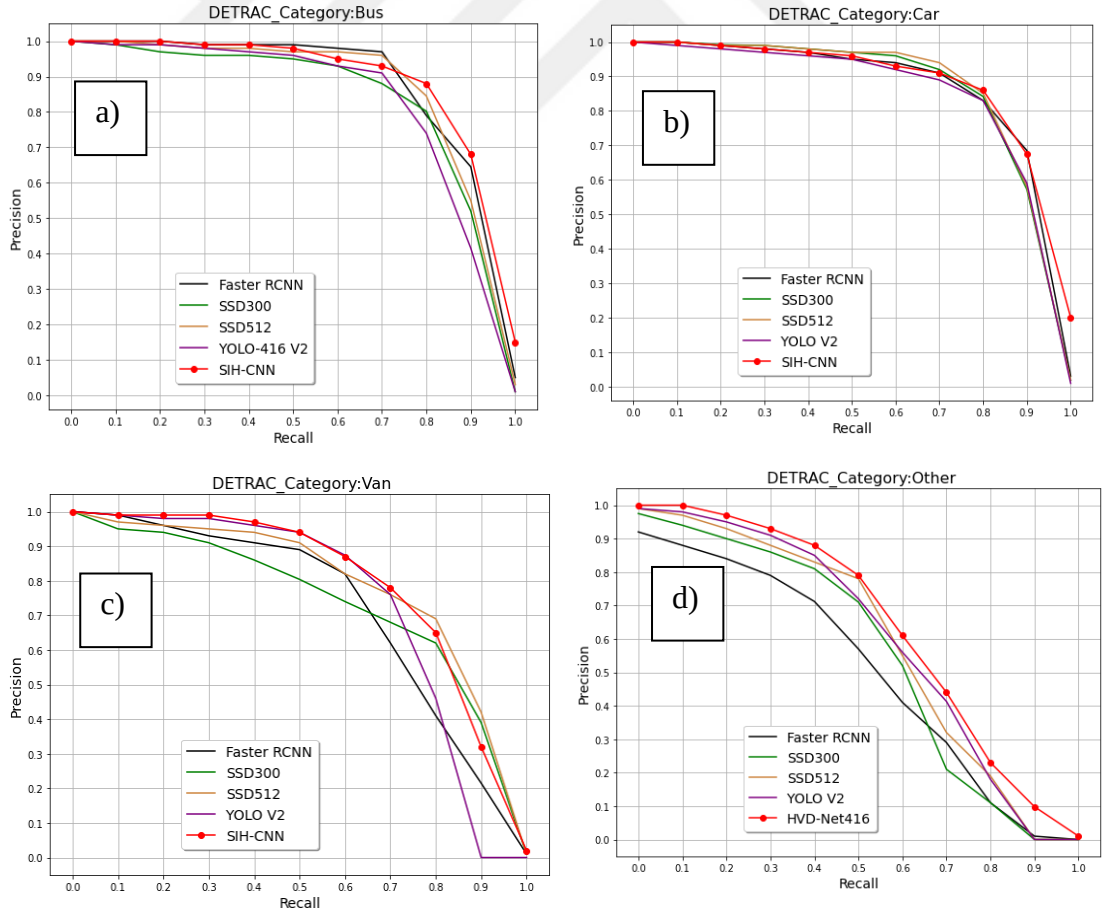

Figure 5.4. a) AUC of Bus Class, b) AUC of Car Class, c) AUC of Van Class, d) AUC of Others Class [69].

5.2.2. Performance Metrics Across Various Models

Table 6.1 comprehensively evaluates various vehicle identification algorithms using the UA-DETRAC dataset. The evaluation focuses on four specific vehicle categories: buses, cars, vans, and others.

The SIH-CNN model, as proposed, demonstrates exceptional performance by achieving a mAP score of 77.76%, the highest among the models evaluated. The model has remarkable performance in terms of achieving the highest Average Precision (AP) scores for cars (86.03%), vans (77.25%), and the miscellaneous category (63.15%). The AP score for buses, 84.61%, shows a slight decrease compared to the Faster RCNN model. However, the Faster-RCNN model has a mean average precision (mAP) of 72.67%. The SSD300 model achieves a slightly higher mAP of 73.08%, while the SSD512 model achieves a higher mAP of 75.99%. In addition, the YOLO-V2 model achieves a mAP of 73.82%.

Table 6.1. Model Performance comparison [69]

Vehicle Detection Framework	**CNN Architecture**	**Input Size**	**Buses AP %**	**Cars AP %**	**Vans AP %**	**Others Category AP %**	**mAP %**	**FPS**
Faster-RCNN [27]	VGG-16	600x600	**85.49**	84.4	70.49	50.29	72.67	12.7
SSD300 [20]	VGG-16	300x300	81.56	84.05	71.85	54.86	73.08	25.2
SSD512 [20]	VGG-16	512x512	84.32	84.46	76.64	58.55	75.99	19.6
YOLO-V2 [14]	DarkNet-19	416x416	80.86	82.63	72.22	59.57	73.82	**51.7**
Proposed Framework	SIH-CNN	416x416	84.61	**86.03**	**77.25**	**63.15**	**77.76**	48.4

5.2.3. Analysis of Detection Accuracy

A spider graph shown in Figure 6.2 visually illustrates the average precision values for different types of vehicles, including cars, buses, vans, and others. The numbers in the graph were collected using various detectors. The data indicates that the SIH-CNN model described in this report has exhibited a higher performance level than all other

detectors concerning average precision across all classes. The most significant finding is that the SIH-CNN curve covers the largest area of the graph. The discovery suggests that the proposed model demonstrates superior precision and reliability in identifying various car categories in contrast to other advanced detectors currently available.

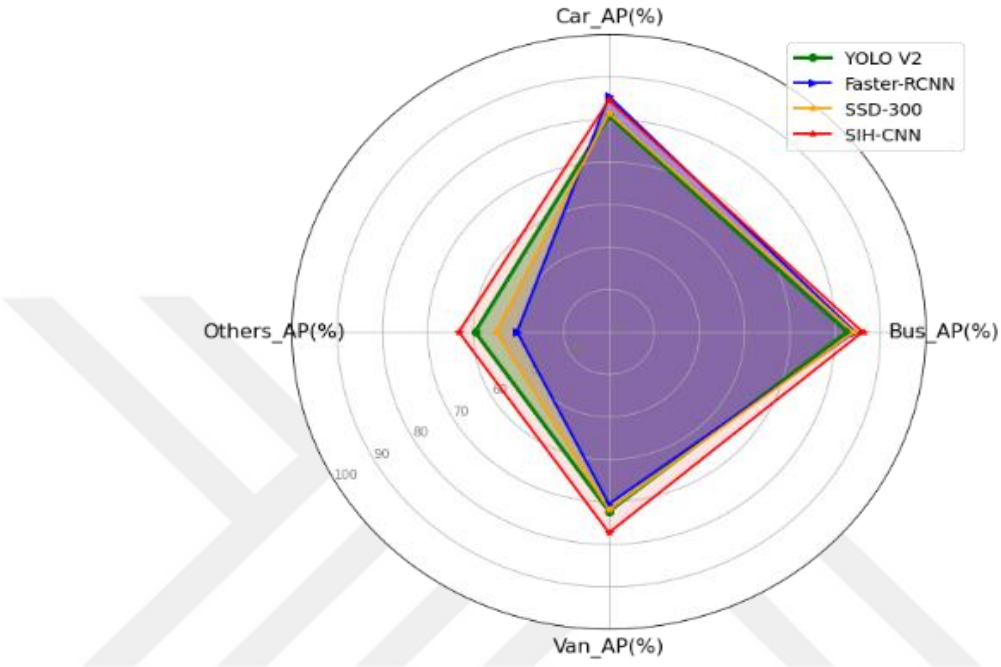

Figure 5.5. Average precision of all vehicle classes [69].

5.2.4. Qualitative Results

Figure 5.6 Displays SIH-CNN's vehicle detection performance on the UA-DETRAC benchmark, showing its ability to detect various vehicle sizes with appropriate class labels using video frames.

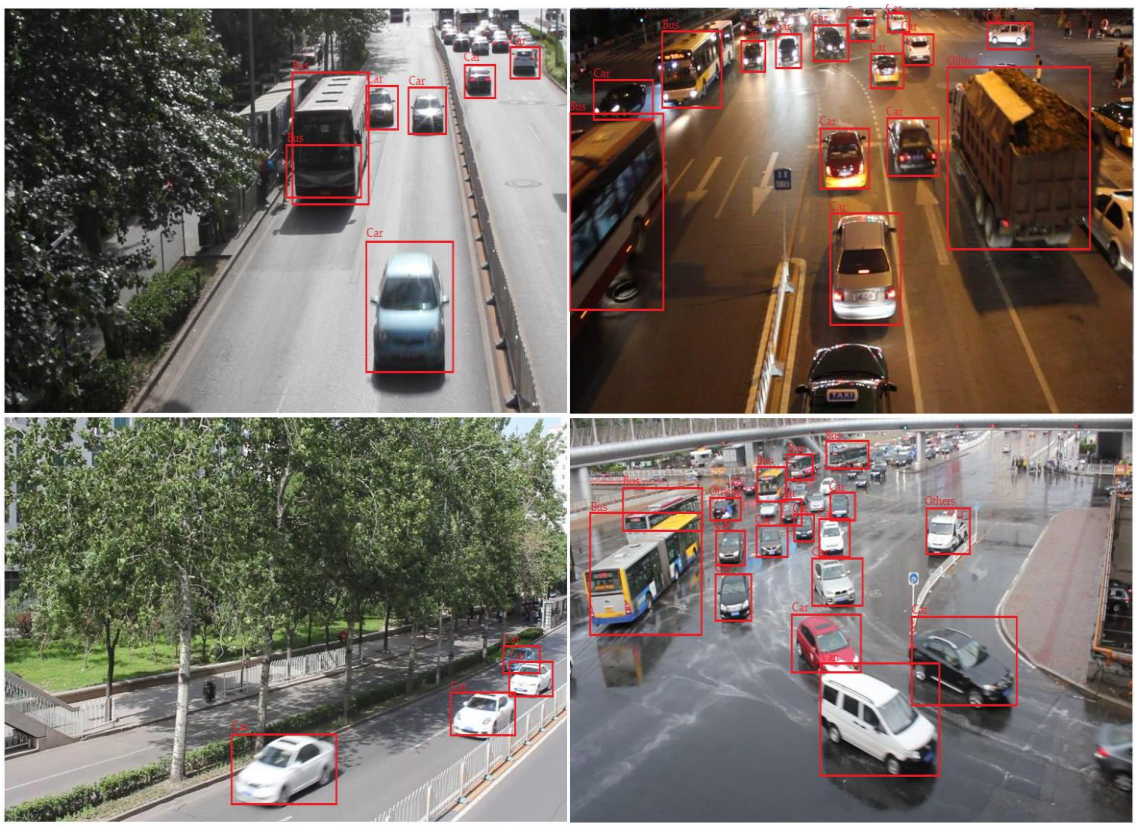

Figure 5.6. Vehicle detection for SIH-CNN Model [69].

5.2.5. Discussion

- Synthesis of results and their implications: The findings underline the SIH-CNN model's potential as a leading tool for accurate, real-time vehicular detection.
- Significance in the context of vehicular detection: The SIH-CNN strikes a balance between accuracy and real-time performance, maintaining competitive detection accuracy while processing frames in real-time.
- The AUC (area under the curve) on the PR (Precision-Recall) curve indicates a model's detection capability. A high AUC is indicative of superior model performance.
- Overview of SIH-CNN's superior performance: The empirical findings unequivocally demonstrate that the SIH-CNN model has exceptional performance across all vehicle categories. The results show that the SIH-CNN model is highly effective and accurate in real-world vehicle detection.

PART 6

CONCLUSION

Vehicle detection in surveillance systems poses significant challenges due to the variety of vehicles' sizes, styles, colors and shapes. This diversity creates a daunting obstacle for detection systems, making identifying them reliably and consistently challenging. Additionally, the geometric transformations of cars over successive video frames, influenced by their position and orientation, introduce further complexities to the detection and recognition process. Environmental variables, like dust, rain and clouds, can significantly add to the situation's complexity, which may jeopardies the accuracy of the results. The existing object detectors tend to neglect the crucial inter-frame information contained within the video feeds, as they are mainly developed for single-image detection purposes. Moreover, the application of grid cells for detection presents certain limitations. Although larger grid sizes prove effective for small object detection, they also result in high computational costs.

Conversely, smaller grid sizes are computationally efficient but may not accurately detect vehicles. Moreover, fixed-length grid cells may struggle to predict vehicles far from cameras. Furthermore, some systems, such as those utilizing DarkNet-19 as their underlying framework, fail to recognize the importance of incorporating multi-level features, which are of utmost importance in efficiently handling variations in vehicle class.

This thesis aims to attain multiple crucial goals in vehicle identification for surveillance systems that operate in real-time. To begin with, it concentrates on elevating the precision and effectiveness of car detection, specifically for distant or small vehicles, via an innovative technique that employs multi-level features and essential inter-frame details. This thesis puts forward a Scale Invariant Hybrid Convolutional Neural Network (SIH-CNN) model to tackle the issue of differing

scales in vehicles on the move. The SIH-CNN architecture adeptly adjusts to changing vehicle sizes across consecutive frames, enhancing vehicle detection accuracy and reliability. Moreover, incorporating a multi-level feature extraction block rectifies issues relating to gradient vanishing and improves the model's capacity to deal with class variation, thus contributing to the overall fortitude of the detection system. Additionally, a multi-scale feature extraction block is included to tackle the detection of diminutive vehicles, allowing for a more precise identification of small vehicles. Ultimately, the SIH-CNN proposed exhibits an impressive 48.4 frames per second, thus demonstrating its excellent efficiency and suitability for observing real-time activities.

In future work, we will utilize the same multilevel and multiscale features to develop a vehicle license plate recognition procedure, representing the next phase of vehicle surveillance in river dredging areas.

REFERENCES

1. de-Lima-Santos, M. F., & Ceron, W., "Artificial intelligence in news media: current perceptions and future outlook". ***Journalism and media***, *3*(1), 13-26 (2021).

2. Ramirez-Amaro, K., Beetz, M., & Cheng, G., "Transferring skills to humanoid robots by extracting semantic representations from observations of human activities". ***Artificial Intelligence***, *247*, 95-118 (2017).

3. Ang, L. M., Seng, K. P., Zungeru, A. M., & Ijemaru, G. K., "Big sensor data systems for smart cities". ***IEEE Internet of Things Journal***, *4*(5), 1259-1271 (2017).

4. Perwej, Y., Haq, K., Parwej, F., Mumdouh, M., & Hassan, M., "The internet of things (IoT) and its application domains". ***International Journal of Computer Applications***, *975*(8887), 182 (2019).

5. Cisco. (2011) Global internet traffic projected to quadruple by 2015. [Online]. Available:https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=324003.

6. Corchado Rodríguez, J. M., "Efficiency and Reliability in Bringing AI into Transport and Smart City Solutions" (2020).

7. Turtiainen, H., Costin, A., Lahtinen, T., Sintonen, L., & Hamalainen, T., "Towards large-scale, automated, accurate detection of CCTV camera objects using computer vision". Applications and implications for privacy, safety, and cybersecurity.(Preprint). ***arXiv preprint arXiv:2006.03870*** (2020).

8. Qian, Y., Yu, L., Liu, W., & Hauptmann, A. G., "Electricity: An efficient multi-camera vehicle tracking system for intelligent city". In ***Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*** (pp. 588-589) (2020).

9. Szeliski, R., & Szeliski, R., "Segmentation". ***Computer Vision: Algorithms and Applications***, 235-271 (2011).

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

11. Guerrero-Ibáñez, J., Zeadally, S., & Contreras-Castillo, J., "Sensor technologies for intelligent transportation systems". ***Sensors***, *18*(4), 1212 (2018).

12. Hu, X., Xu, X., Xiao, Y., Chen, H., He, S., Qin, J., & Heng, P. A., "SINet: A scale-

insensitive convolutional neural network for fast vehicle detection", ***IEEE transactions on intelligent transportation systems***, *20*(3), 1010-1019 (2018).

13. Chen, L., Ye, F., Ruan, Y., Fan, H., & Chen, Q., "An algorithm for highway vehicle detection based on convolutional neural network", ***Eurasip Journal on Image and Video Processing***, 1-7 (2018).

14. Redmon, J., & Farhadi, A., "YOLO9000: better, faster, stronger", **In *Proceedings of the IEEE conference on computer vision and pattern recognition***, (pp. 7263-7271) (2017).

15. Haritha, H., & Thangavel, S. K., "A modified deep learning architecture for vehicle detection in traffic monitoring system", ***International Journal of Computers and Applications***, *43*(9), 968-977 (2021).

16. Alam, A., Jaffery, Z. A., & Sharma, H., "A cost-effective computer vision-based vehicle detection system", ***Concurrent Engineering***, *30*(2), 148-158 (2022).

17. Gomaa, A., Minematsu, T., Abdelwahab, M. M., Abo-Zahhad, M., & Taniguchi, R. I., "Faster CNN-based vehicle detection and counting strategy for fixed camera scenes", ***Multimedia Tools and Applications***, *81*(18), 25443-25471 (2022).

18. Deshmukh, P., Satyanarayana, G. S. R., Majhi, S., Sahoo, U. K., & Das, S. K., "Swin transformer-based vehicle detection in undisciplined traffic environment", ***Expert Systems with Applications***, *213*, 118992 (2023).

19. Arora, N., Kumar, Y., Karkra, R., & Kumar, M., "Automatic vehicle detection system in different environment conditions using fast R-CNN", ***Multimedia Tools and Applications***, *81*(13), 18715-18735 (2022).

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

21. Chen, Z., Guo, H., Yang, J., Jiao, H., Feng, Z., Chen, L., & Gao, T., "Fast vehicle detection algorithm in traffic scene based on improved SSD", ***Measurement***, *201*, 111655 (2022).

22. Zhai, J., Li, B., Lv, S., & Zhou, Q., "FPGA-based vehicle detection and tracking accelerator", ***Sensors***, *23*(4), 2208 (2023).

23. Mittal, U., Chawla, P., & Tiwari, R., "EnsembleNet: A hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models", ***Neural Computing and Applications***, *35*(6), 4755-4774 (2023).

24. Ghosh, R., "An Improved You Only Look Once Based Intelligent System for Moving Vehicle Detection", ***International Journal of Intelligent Transportation***

Systems Research, 1-9 (2023).

25. Uluer, A., Albayrak, Z., Ozalp, A., & Cakmak, M., " BGP Anomali Tespitinde Hibrit Model Yaklaşımı ***"30th Signal Processing and Communications Applications Conference,*** (2022).

26. Girshick, R., "Fast r-cnn**", In *Proceedings of the IEEE international conference on computer vision,*** (pp. 1440-1448) (2015).

27. Han, G., Zhang, X., & Li, C., "Revisiting faster r-cnn: A deeper look at region proposal network", ***In Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China,*** *November 14-18, 2017, Proceedings, Part III 24* (pp. 14-24). Springer International Publishing.

28. Dai, J., Li, Y., He, K., & Sun, J., "R-fcn: Object detection via region-based fully convolutional networks", ***Advances in neural information processing systems***, *29* (2016).

29. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A., "You only look once: Unified, real-time object detection**", In *Proceedings of the IEEE conference on computer vision and pattern recognition*** (pp. 779-788) (2016).

30. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P., "Focal loss for dense object detection", In ***Proceedings of the IEEE international conference on computer vision*** (pp. 2980-2988) (2017).

31. González García, C., Núñez Valdéz, E. R., García Díaz, V., Pelayo García-Bustelo, B. C., & Cueva Lovelle, J. M., "A review of artificial intelligence in the internet of things". ***International Journal Of Interactive Multimedia And Artificial Intelligence***, *5* (2019).

32. Saravanan, R., & Sujatha, P., "A state of art techniques on machine learning algorithms: a perspective of supervised learning approaches in data classification". In ***2018 Second international conference on intelligent computing and control systems (ICICCS)*** (pp. 945-949). IEEE (2018).

33. Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T., & Philbrick, K., "Toolkits and libraries for deep learning". ***Journal of digital imaging***, *30*, 400-405 (2017).

34. Sözen, A., Akçayol, M. A., & Arcaklioğlu, E., "Forecasting net energy consumption using artificial neural network". ***Energy Sources, Part B***, *1*(2), 147-155 (2006).

35. Bzdok, D., & Meyer-Lindenberg, A., "Machine learning for precision psychiatry: opportunities and challenges". ***Biological Psychiatry: Cognitive Neuroscience and Neuroimaging***, *3*(3), 223-230 (2018).

36. Surden, H., & Williams, M. A., "Technological opacity, predictability, and self-driving cars". ***Cardozo L. Rev.***, *38*, 121 (2016).

37. Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J., "Google's neural machine translation system: Bridging the gap between human and machine translation". ***arXiv preprint arXiv:1609.08144*** (2016).

38. Alam, M., Samad, M. D., Vidyaratne, L., Glandon, A., & Iftekharuddin, K. M., "Survey on deep neural networks in speech and vision systems". ***Neurocomputing, 417***, 302-321 (2020).

39. Albayrak, Z., Çakmak, M., " A Review: Mobile Communication Past, Present and Future ***". International Conference on Advanced Technologies, Computer Engineering and Science,*** 141-145 (2018).

40. Pérez-Ortiz, M., Jiménez-Fernández, S., Gutiérrez, P. A., Alexandre, E., Hervás-Martínez, C., & Salcedo-Sanz, S., "A review of classification problems and algorithms in renewable energy applications". ***Energies***, *9*(8), 607 (2016).

41. Chen, X., & Gupta, A., "Webly supervised learning of convolutional networks" In ***Proceedings of the IEEE international conference on computer vision*** (pp. 1431-1439) (2015).

42. Saffran, J. R., Aslin, R. N., & Newport, E. L., "Statistical learning by 8-month-old infants". ***Science***, *274*(5294), 1926-1928 (1996).

43. Gerón, A., "Chapter 10: Introduction to Artificial Neural Networks with Keras. ***Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow***. *OReilly"* (2017)

44. Gerón, A., "Chapter 10: Introduction to Artificial Neural Networks with Keras. ***Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow***. OReilly" (2017)

45. Çakmak, M., Albayrak, Z., "Performance Analysis of Queue Management Algorithms Between Remote-Host and PG-W in LTE Networks". ***Academic Platform Journal of Engineering and Science***, 456-463 (2020).

46. Taye, M. M., "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions". ***Computers***, 12 (5), 91 (2023)

47. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L., "Imagenet large scale visual recognition challenge". ***International journal of computer vision***, 115, 211-252 (2015).

48. Linardatos, P., Papastefanopoulos, V., & Kotsiantis, S., "Explainable ai: A review of machine learning interpretability methods". ***Entropy***, *23*(1), 18 (2020).

49. Turner, J., Cano, J., Radu, V., Crowley, E. J., O'Boyle, M., & Storkey, A., "Characterising across-stack optimisations for deep convolutional neural

networks” In ***2018 IEEE International Symposium on Workload Characterization (IISWC)*** (pp. 101-110). IEEE (2018).

50. Géron, A., “*Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " ***O'Reilly Media, Inc***."(2022).

51. Chang, K., Balachandar, N., Lam, C., Yi, D., Brown, J., Beers, A., ... & Kalpathy-Cramer, J., "Distributed deep learning networks among institutions for medical imaging". ***Journal of the American Medical Informatics Association***, *25*(8), 945-954 (2018).

52. Cakmak, M. & Albayrak, Z., "AFCC-r: Adaptive Feedback Congestion Control Algorithm to Avoid Queue Overflow in LTE Networks ". ***Mobile Networks and Applications***, 7, 11036-022-02011-8 (2022).

53. Girshick, R., Donahue, J., Darrell, T., & Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation". ***In Proceedings of the IEEE conference on computer vision and pattern recognition*** (pp. 580-587) (2014).

54. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A., "The pascal visual object classes (voc) challenge". ***International journal of computer vision***, *88*, 303-338 (2010).

55. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L., "Microsoft coco: Common objects in context". **In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September*** *6-12, 2014, Proceedings, Part V 13* (pp. 740-755). Springer International Publishing.

56. Padilla, R., Netto, S. L., & Da Silva, E. A., “A survey on performance metrics for object-detection algorithms”. In ***2020 international conference on systems, signals and image processing (IWSSIP)*** (pp. 237-242). IEEE (2020).

57. Fleet, D., Pajdla, T., Schiele, B., & Tuytelaars, T. (Eds.)., "***Computer Vision--ECCV 2014: 13th European Conference", Zurich, Switzerland, September*** *6-12, 2014, Proceedings, Part I* (Vol. 8689). Springer (2014).

58. Davis, J., & Goadrich, M., "The relationship between Precision-Recall and ROC curves". ***In Proceedings of the 23rd international conference on Machine learning*** (pp. 233-240) (2006).

59. Saito, T., & Rehmsmeier, M., "Precrec: fast and accurate precision–recall and ROC curve calculations in R". ***Bioinformatics***, *33*(1), 145-147 (2017).

60. Xu, D., & Wu, Y., “Improved YOLO-V3 with DenseNet for multi-scale remote sensing target detection”. ***Sensors***, *20*(15), 4276 (2020).

61. Varadarajan, V., Garg, D., & Kotecha, K., "An efficient deep convolutional neural network approach for object detection and recognition using a multi-scale anchor box in real-time". ***Future Internet***, *13*(12), 307 (2021).

62. Jin, T., Ye, X., Li, Z., & Huo, Z., "Identification and Tracking of Vehicles between Multiple Cameras on Bridges Using a YOLOv4 and OSNet-Based Method". ***Sensors***, *23*(12), 5510 (2023).

63. Xu, Y., Xiao, T., Zhang, J., Yang, K., & Zhang, Z., "Scale-invariant convolutional neural networks". ***arXiv preprint arXiv****:1411.6369* (2014).

64. Ashraf, M. H., Jabeen, F., Alghamdi, H., Zia, M. S., & Almutairi, M. S., "HVD-Net: A Hybrid Vehicle Detection Network for Vision-Based Vehicle Tracking and Speed Estimation". ***Journal of King Saud University-Computer and Information Sciences,*** *35*(8), 101657 (2023).

65. Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M. C., Qi, H., ... & Lyu, S., "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking". ***Computer Vision and Image Understanding***, *193*, 102907 (2020).

66. Altunay, H. C., & Albayrak, Z., "A hybrid CNN+ LSTMbased intrusion detection system for industrial IoT networks". ***Engineering Science and Technology, an International Journal***, *38*, 101322 (2023).

67. Özalp, A. N., & Albayrak, Z., "Detecting Cyber Attacks with High-Frequency Features using Machine Learning" **Algorithms**. ***Acta Polytechnica Hungarica***, *19*(7), 213-233 (2022).

68. Salih, M. M. M., & Çakmak, M., "Neural Network Approach For Classification And Detection Of Chest Infection". ***In 2022 2nd International Conference on Computing and Machine Intelligence (ICMI)*** (pp. 1-5). IEEE (2022).

69. Al Bayati, M. A. Z., & Çakmak, M., "Real-Time Vehicle Detection for Surveillance of River Dredging Areas Using Convolutional Neural Networks", ***International Journal of Image, Graphics and Signal Processing (IJIGSP)***, Vol.15, No.5, pp. 17-28, (2023).

RESUME

Mohammad Abdul-Jabbar ZAID completed their primary and secondary education in Baghdad, Iraq. Followed by the achievement of a bachelorette's degree from AL-Ma ‘moon University College Computer Engineering Department in 2014.

Shortly after graduation, he embarked on a professional journey with Ministry of Water Resources, where he served as a in the Information Technology Department. Eager to further expand their academic horizons, Mohammad Abdul-Jabbar ZAID then decided to pursue a master's degree.

In 2021, a new chapter began in Turkey, where he commenced their postgraduate studies at Karabük University, specifically in the Department of Computer Engineering."

He published his first scientific research in 2023 in Scopus index journal and it was in his field of research interest. under title " Real-Time Vehicle Detection for Surveillance of River Dredging Areas Using Convolutional Neural Networks ".