



REPUBLIC OF TÜRKİYE

ALTINBAŞ UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

**STUDY THE POSSIBILITY OF DETECTING
CAR ACCIDENTS AND RECOGNITION OF
CAR'S PLATE NUMBERS BY USING (AI)**

Ali Mooaid Salman AL-MADHACHI

Master's Thesis

Supervisor

Ass. Prof. Dr. Hasan ABDULKADER

Istanbul, 2022

**STUDY THE POSSIBILITY OF DETECTING CAR ACCIDENTS AND
RECOGNITION OF CAR'S PLATE NUMBER BY USING (AI)**

Ali Mooaid AL-MADHACHI

Electrical and Computer Engineering

Master's Thesis

ALTINBAŞ UNIVERSITY

2022

The thesis titled STUDY THE POSSIBILITY OF DETECTING CAR ACCIDENTS AND RECOGNITION OF CAR'S PLATE NUMBER BY USING (AI) prepared by ALI MOOAID SALMAN AL-MADHACHI and submitted on 14/12/2022 has been **accepted unanimously** for the degree of Master of Science in Electrical and Computer Engineering

Asst. Prof. Dr HASAN ABDULKADER

The Supervisor

Thesis Defense Committee Members:

Asst. Prof. Dr HASAN
ABDULKADER

Faculty Of Engineering And
Natural Science ,
Altinbas University

Asst. prof. Dr TIMUR INAN

Faculty Of Engineering And
Natural Science,

AltinbasUniversity

Asst. Prof. Dr ABDULKADER
ALWER

Faculty Of Engineering And
Natural Science,

AltinbasUniversity

I hereby declare that this thesis/dissertation meets all format and submission requirements of a Master's thesis

Submission date of the thesis to Institute of Graduate Studies: ____/____/____

I hereby declare that all information/data presented in this graduation project has been obtained in full accordance with academic rules and ethical conduct. I also declare all unoriginal materials and conclusions have been cited in the text and all references mentioned in the Reference List have been cited in the text, and vice versa as required by the abovementioned rules and conduct.

Ali Mooaid Salman ALMADHACHI

Signature

DEDICATION

First and foremost, I want to express my gratitude to Allah Almighty for providing me with the Mental acuity, health, and stamina necessary to finish this course of study.

I would like to thank my sincere thanks to everyone who supported me and helped me in my academic career, especially my supervisor Ass. Prof. Dr. Hasan Abdulkader for his efforts and cooperation in supervising this thesis.

To my parents, I cannot repay Their grace and efforts for me, they support me from the beginning of my life to what I have reached now, words will not be enough to express how they tried for me in all my life stages.

I dedicate this thesis to my wife, the person that deserves all meaning of thanks and gratitude for supporting and encouraging me during my academic career.

ABSTRACT

Study The Possibility of Detecting Car Accidents and Recognition of Car's Plate Numbers by Using (AI)

ALMADHACHI, Ali

MSc. electrical and computer engineering, Altınbaş University,

Supervisor: Asst. Prof. Dr. Hassan Abdulkader

Date: December/2022

Pages: 79

Car accidents cause the death of more than a million humans every year, and most of these deaths are due to the special delay in assisting the wounded in proper time. Especially the accidents that occur outside the cities or on highways. Car accidents is considered a serious issue that cannot be avoided, even with strict traffic laws in some countries. With the development of artificial intelligence algorithms and the evolution of deep learning techniques, some researchers suggest and create models or methods that can detect car accidents and then inform the relevant units for the purpose of supporting quick assistance to wounded and injured people. Most of the previous methods adopt and based on image processing and video analysis. In this thesis, first we study the possibility of detecting car accidents by reviewing prior research related to detecting car accidents. Then based on deep learning with computer vision techniques which is considered a part of the wide artificial intelligence field, we propose a method to detect accidents using YOLOv5. In addition, we have trained Faster R-CNN, and the purpose is to compare and select the most suitable model for the mentioned task. Besides to accident detection, we designed two different models to recognize the car plate numbers, one of them includes the use of the traditional Easy OCR, and the other one includes the use of YOLOv7 with Tesseract engine to detect the plates' number first and then recognize and extract the characters of the plate number. Extensive experiments showed that our model can detect car accidents with an accuracy of 76.2%, which is less than the accuracy of Faster R-CNN with two different types of backbones that achieve accuracy of 81.5%, and 78% successively. For the second task, Easy OCR achieves accuracy 96%, which is better than the

YOLOv7-based model which achieves 94.3% for recognizing plate numbers. The main objective of this thesis is to design models for safety and security purposes.

Keywords: Artificial intelligence, Artificial Neural Networks, Computer vision, Convolutional Neural Networks, Deep Learning, Object Detection.



TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	vi
LIST OF FIGURES	xi
ABBREVIATIONS	xiv
1. INTRODUCTION	1
1.1 CONVOLUTIONAL NETWORK	4
1.2 AN OVERVIEW OF OBJECT DETECTION	8
1.3 COMPUTER VISION	10
1.4 RESEARCH QUESTION	11
2. Related Work	12
2.1 PREVIOUS METHODS TO DETECT CAR ACCIDENTS	12
2.1.1 Methods that fully depended on the software and AI techniques	12
2.1.2 Methods that depend on both hardware and AI techniques	14
2.2 PREVIOUS METHODS TO RECOGNIZE CAR LICENSE PLATE NUMBER	16
3. MATERIAL AND METHODS	19
3.1 SINGLE STEP OBJECT DETECTION ALGORITHM	19
3.2 TWO STEPS OBJECT DETECTION ALGORITHM	21
3.3 CHARACTER RECOGNITION BY EASY OCR	23
3.4 CHARACTER RECOGNITION BY TESSEARCT	25
3.5 PROPOSED METHODS FOR CARS ACCIDENTS DETECTION	26
3.5.1 Accidents Detection by YOLO v5	34
3.5.2 Accidents Detection by Faster R-CNN	40
3.6 PROPOSED METHODS FOR CARS PLATE NUMBER RECOGNITION	44
3.6.1 Car Plate Number Recognition by Using Easy OCR	44

3.6.2 Car Plate Number Recognition by Using YOLOv7 with Pytesearch	47
4. EXPERIMENTS AND DISCUSSIONS	50
4.1 CAR ACCIDENT DETECTION RESULTS AND DISCUSSIONS	50
4.2 CAR PLATE NUMBER RECOGNITION RESULTS AND DISCUSSION	56
5. CONCLUSION	61
REFERENCES	63



LIST OF TABLES

	<u>Page</u>
Table 3.1: Comparison Between Different Two-Step Detectors Models.....	35
Table 3.2: Components of MobileNet v3 [44].....	43
Table 4.1: Performance Evaluation of the Training Step for YOLOv5.....	52
Table 4.2: Performance Evaluation of the Training Step for Faster R-CNN.....	56
Table 4.3: Comparison between YOLOv5 and Faster R-CNN on CAD-DATASET.....	58

LIST OF FIGURES

	<u>Page</u>
Figure 1.1: Relationship Between AI and ML and DL	2
Figure 1.2: Structure of DNN [5].....	3
Figure 1.3: CNN capability to detect the object of different classes.....	4
Figure 1.4: Images Channels.....	5
Figure 1.5: Effect of using kernel.....	6
Figure 1.6: Effect of ReLU.....	7
Figure 1.7: Max and sum pooling effect.....	7
Figure 1.8: classification stage of CNN.....	8
Figure 1.9: Two approaches of object detection [6].....	10
Figure 3.1: Single Step Detection Architecture.....	19
Figure 3.2: Grid Cells And Resulting Vectors [26].....	19
Figure 3.3: VGG Backbone Architecture.....	20
Figure 3.4: Backbone After Pre-Training Process[26].....	20
Figure 3.5: Relation Between The Grid Cells And The Features Maps[26].....	21
Figure 3.6: Difference Between Single-Step and Two-Step Detectors[26]	21
Figure 3.7: The Work Basis Of RPN[7].....	22
Figure 3.8: Structure Diagram Of Faster R-CNN[7]	23
Figure 3.9: Easy OCR Framework.....	24
Figure 3.10: EasyOCR Framework.....	26

Figure 3.11 Feature network design.....	30
Figure 3.12 training time comparison between YOLOv4 and YOLOv5.....	30
Figure 3.13 Both Models Have Similar mAP.....	31
Figure 3.14 Performance Comparison On The MS COCO Benchmark.....	31
Figure 3.15 R-CNN System Overview [28].....	32
Figure 3.16 The Whole Process of R-CNN [28].....	33
Figure 3.17 SPPNet vs Fast R-CNN [29].....	33
Figure 3.18 Faster R-CNN Approach [7].....	34
Figure 3.19 Instances From CAD-DATASET.....	36
Figure 3.20 Train Batch (0).....	37
Figure 3.21 Train Batch (1).....	38
Figure 3.22 New Csp DarkNet-Backbone.....	39
Figure 3.23 Proposed Intersection Between Bounding Boxes.....	40
Figure 3.24 Architecture of MobileNet v3 [44].....	42
Figure 3.25 last stage in MobileNet v2 and efficient last stage in MobileNetv3 [44].....	42
Figure 3.26 Input Image.....	46
Figure 3.27 Gary Image (step 1).....	46
Figure 3.28 The Effect of Bilateral Filter and Canny Edge Detection (step 2).....	46
Figure 3.29 The Resulted Image After (step3, 4).....	47
Figure 3.30 The Resulted Image After (step 5).....	48
Figure 3.31 final resulting image.....	48

Figure 3.32 Output Image From YOLOv7.....	50
Figure 3.33 Region Of Interest Image.....	51
Figure 3.34 Extracted Characters from Image.....	51
Figure 4.1 Results of Training and Validation Step.....	53
Figure 4.2 metric achieved after training and validation step with 300 epochs.....	54
Figure 4.3 Accident Detected by Intersection Between Bounding Box.....	55
Figure 4.4 Accident Detected by YOLOv5 Itself.....	55
Figure 4.5 Metric of Faster R-CNN with ResNet50-FPN.....	56
Figure 4.6 Overall mAP and mAP for Large Object for Faster R-CNN (mobilev3 large)....	57
Figure 4.7 Recall at different thresholds.....	57
Figure 4.8 Accidents Detect By Faster R-CNN Itself.....	57
Figure 4.9 Accidents Detect By Intersection Between Bounding Boxes.....	58
Figure 4.10 Plate Number Recognition by EasyOCR.....	59
Figure 4.11 F1 Score for YOLOv7.....	60
Figure 4.12 Metrics for YOLOv7.....	60
Figure 4.13 Plate Number Detected By YOLOv7.....	61
Figure 4.14 samples of isolated plate numbers by the second step.....	62
Figure 4.15 samples of isolated plate numbers by the second step.....	63

ABBREVIATIONS

RAM	: Random Access Memory
CPU	: Central Processing Unit
AI	: Artificial Intelligence
YOLO	: You Look Only Once
Faster R-CNN	: Faster Region-Based Convolutional Neural Networks
Easy OCR	: Easy Optical Character Recognition
ML	: Machine Learning
DNN	: Deep Neural Network
ANN	: Artificial Neural Network
RNN	: Recurrent Neural Network
CNN	: Convolutional Neural Network
ReLU	: Rectified Linear Unit
LeNet	: Learnable Network
SVM	: Support Vector Machine
DPM	: Deformable Part-based Model
Mask R-CNN	: Mask Region-Based Convolutional Neural Networks
Fast R-CNN	: Fast Region-Based Convolutional Neural Networks
SSD	: Single Shot Detection
G-CNN	: Grid Convolutional Neural Network

DSSD	: Deconvolutional Single Shot Detector
DSOD	: Deeply Supervised Object Detector
GRU	: Gated Recurrent Unit
ROC- AUC	: A Receiver Operating Characteristic - Area; Under The Curve
SORT	: Simple Online Real-Time Tracking
CVIS	: Corporative Vehicle Infrastructure System
AP	: Average Precision
MSFF	: Multi-Scale Feature Fusion
LSTM	: Long Short-Term Memory
SHRP2	: Strategic Highway Research Program
NDS	: Naturalistic Driving Study
MLP	: Multilayer Perceptron
AE	: Auto Encoders
HC	: Healthy Control
PRC	: Pattern Recognition Chain
ERs	: Extremal Regions
HDRBMs	: Hybrid Discriminative Restricted Boltzmann Machines
LPs	: License Plates
CRR	: Character Recognition Rate
LPRS	: License Plate Recognition System
CCA	: Connected Component Analysis

GUI	: Graphic User Interface
ALPR	: Automatic License Plate Recognition
VGG	: Visual Geometry Group
RPN	: Region Proposal Network
ROI	: Region of Interest
NLP	: Natural Language Processing
SAT	: Self-Adversarial-Training
CmBN	: Cross-mini Batch Normalization
WRC	: Weighted-Residual-Connections
CSP	: Cross Stage Partial connections
SPP	: Spatial Pyramid Pooling
PANet	: Path Aggregation Network
NAS-FPN	: Neural Architecture Search-Feature Pyramid Network
SPPF	: Spatial Pyramid Pooling - Fast
FLOPs	: floating-point operations per second.
ResNet	: Residual Network
MnasNet	: Platform-Aware Neural Architecture Search for Mobile
NetAdapt	: Platform-Aware Neural Network Adaptation for Mobile Applications
E-ELAN	: Extended Efficient Layer Aggregation Network

1. INTRODUCTION

According to a statistical study published by World Health Organization in 2018[12], car accidents kill 1.35 million persons each year and 3511 persons each day, a large proportion of deaths occur because of delay in first aiding the victims or injured. Low- and middle-income countries account forming 90% of the world's road traffic deaths, for many reasons such as the lack of rapid and emergency response procedures for car accidents and the lack of road infrastructure suitable for public safety mechanisms. So, it is necessary to Reliance on and benefits from Artificial Intelligence (AI) specifically Deep Learning algorithms .so to reduce the number of car accident victims and at the same to provide surveillance and monitoring systems this research study the possibility of using deep learning algorithms for two objectives :

- a. Car accident detection
- b. Car plate numbers recognition

For the first goal we will study the use of two different algorithms which are YOLO and Faster R-CNN and study the result and compare between them in many terms of speed of processing, accuracy, and consumption of computer resources (RAM, CPU, etc.) The second goal also will implement two different algorithms which are Easy OCR (Optical Character Recognition) and the modern YOLO 7, and computer between them according to speed, accuracy and resource consumption. Recognition of plate numbers is useful for many purposes such as knowing the stolen cars and specifying their location, knowing the cars that exceed the speed limit. It's worth noting we will use the Computer Vision library from python to develop our techniques. This study aims also to detect an accident without using any sensors to measure the velocity of the vehicles and any other radars or sensors because the use of these instruments may be expensive and require maintenance also and many of the sensors have inaccurate results. Besides that, the instruments themselves should be reliable in term of accuracy and speed which make them expensive and lead to an increase in the cost of the surveillance and monitoring system.

To complete and perform our idea and study, it's necessary to understand *deep learning, computer vision, Object Detection, and Convolutional Neural networks (CNN)* to achieve knowledge about any algorithm that will serve our topic and which algorithm or technique will lead to more accurate

results. Also, speed and time of processing are important parameters in choosing the proper algorithm .in general deep learning consider an important section of artificial intelligence [2]. Its modern section is also in comparison with machine learning .actually emergence of deep learning in 2006 was the result for many surveys, research, and development on machine learning. Figure 1.1. Illustrate the relation between AI and Machine Learning and deep learning. AI is a wide and huge field that emerged for the first time in the middle of the last century, development of AI had contributed to the emergence of machine learning which is in turn developed into deep learning [3].

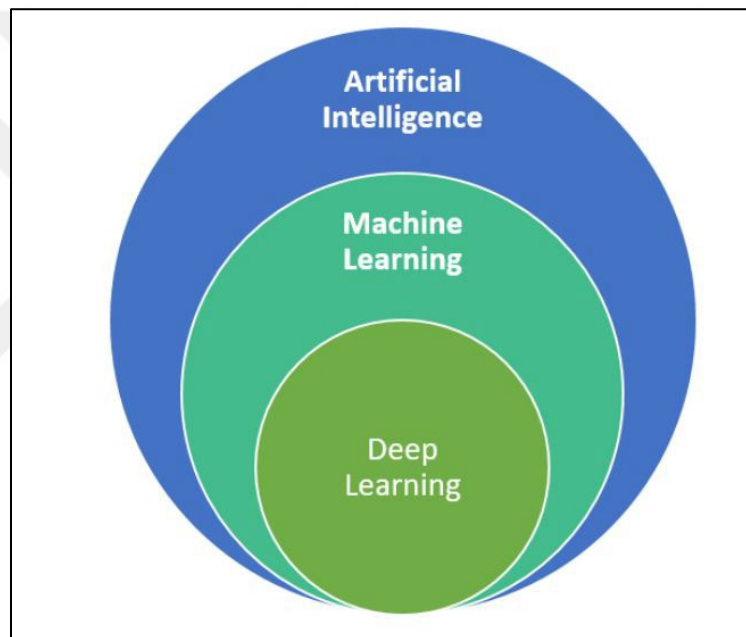


Figure 1.1: Relationship Between AI and ML and DL

Deep learning as a subcategory of ML uses multiple layers of non-linear processing to extract features and transform properties from the input data [3].ML as well as deep learning can be classified into two types, supervised machine learning and unsupervised machine learning.

Supervised deep learning also known as (supervised machine learning) means using and adopting labeled data in the training stage [1]. This means we should label the data (image or video) for example according to many annotations or features. After this labeled data will enter the training stage. Supervised deep learning mostly used in classification tasks. While in Unsupervised deep learning the training stage can be done without using labeled data [1]. Unsupervised deep learning

is mostly used in regression tasks. As a result of the development of machine learning and deep learning, the concept of Deep Neural Networks (DNN) emerged which are constructed and designed to mimic the behaviour of the human brain, specifically the behaviour of neurons [4]. The core of deep learning is the neural network which is also called a deep neural network. Generally, DNN contains three layers: the input layer that represents the link between the world and the network, the hidden layer after that makes processes on the input data, and finally the output layer as shown in Figure 2.1. Which illustrates these layers. many types of DNN differ from each other in the mechanism of feeding and the task that perform it such as Artificial Neural Network (ANN) which consists of multiple perceptrons at each layer and is also known as a feed-forward neural network because the direction of processing of the input data goes only in forward direction [5], whereas Recurrent Neural Network (RNN) has a recurrent connection on the hidden state. This looping constraint ensures that sequential information is captured in the input data. And finally, the convolutional neural network will be implemented in our thesis.

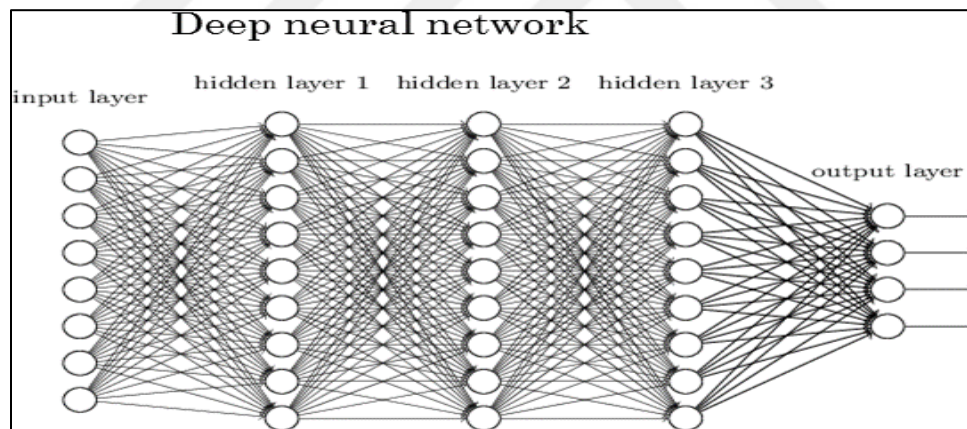


Figure 1.2: Structure of DNN [5]

In this thesis we will use a supervised deep learning model in both algorithms for car accident detection, which is the YOLOv5 single-step object detection algorithm that performs detection and classification at the same time in one step, the second algorithm is Faster R-CNN which proposes region of interest first and in the second step classify the objects. Briefly, we will use algorithms based on different techniques of object detection: Region proposal object detection, regression, or classification object detection .after that we will study the result to discover which algorithm and technique are better to perform the objective. in cars plate numbers recognition especially in the Easy OCR algorithm we will not use supervised learning just in YOLO 7, in general, all methods

and techniques we will implement it can be done by using computer vision as part of deep learning. Object detection considers the basic technique in our thesis since it is the first stage in our mechanism for accident detection and the major process in the car plate number algorithm.

1.1 CONVOLUTIONAL NETWORK

It is a type of neural network that has been proven effective in various fields such as image recognition and classification. Convolutional neural networks consider a promising technique in face recognition, object detection, and traffic signals. CNN has the ability to detect objects like cars, cats, boats, pedestrians, or humans, whether there is one object or several objects of different classes in the same image as shown in Figure 3.1. In addition to image and video processing in classification task CNN also can be used in natural language processing such as parsing and classification of the sentences.

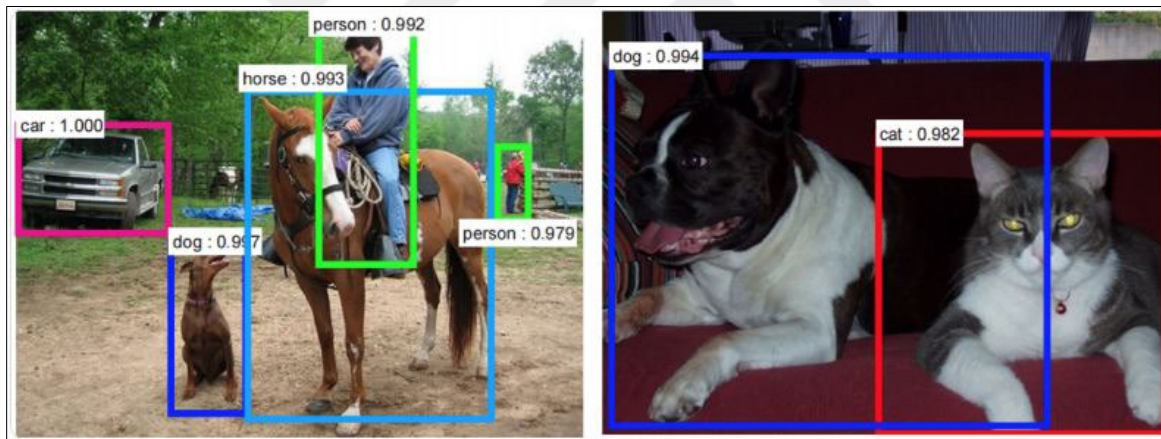


Figure 1.3: CNN capability to detect the object of different classes

CNN consists of four basic successive operations which analyze and extract features from the input data, these basic operations are:

- a. Convolution or filtering
- b. Nonlinearity also called (ReLU)
- c. Pooling or subsampling
- d. Classification

First of all, it's worth noting the image is a matrix consisting of pixels with different values. Some images or matrices contain more than single channels such as RGB images that contain a Red channel, Blue channel, and Green channel, so for example a grayscale image is 2 dimensional with one channel consisting of pixels with values ranging from 0 to 255. So, for the colored image or RGB image is a matrix with three channels Red, Blue, and Green every channel or layer consist of pixel values ranging from 0 to 255 as shown in Figure 4.1 .

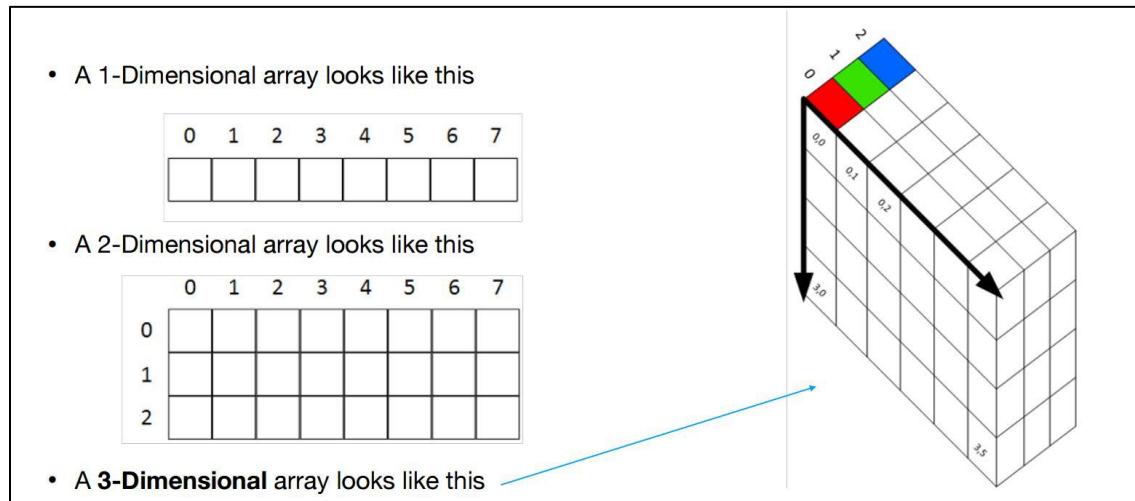


Figure 1.4: Images Channels

For the first process which is convolution or filtering, this step or process aims to extract features from the input image by saving the spatial relations between pixels in the original images by using a small square matrix called kernel on the original matrix. Feature map or convoluted feature can be obtained by sliding the kernel matrix on the original matrix, by one pixel (the size of the scroll step or what is known as the Stride) for each pixel in the image. In each Stride step, the two matrices will be multiplied together, and the product will be summed to extract a single value in the feature map. It should be noted that the Kernel box only sees a portion of the image entered in each pass-through step [2]. Kernel matrix or box sometimes called filter or feature detector and the use of this kernel will lead to finding and getting the feature map. The extracted feature may be edges or curves. There are many types of filters or kernels such as Gaussian blur, box blur, edge detection and sharpen filter. Using each one of the filters will give and lead to different feature maps from other filters of the same input images [2]. Figure 5.1 shows the effect of using filters or kernels.

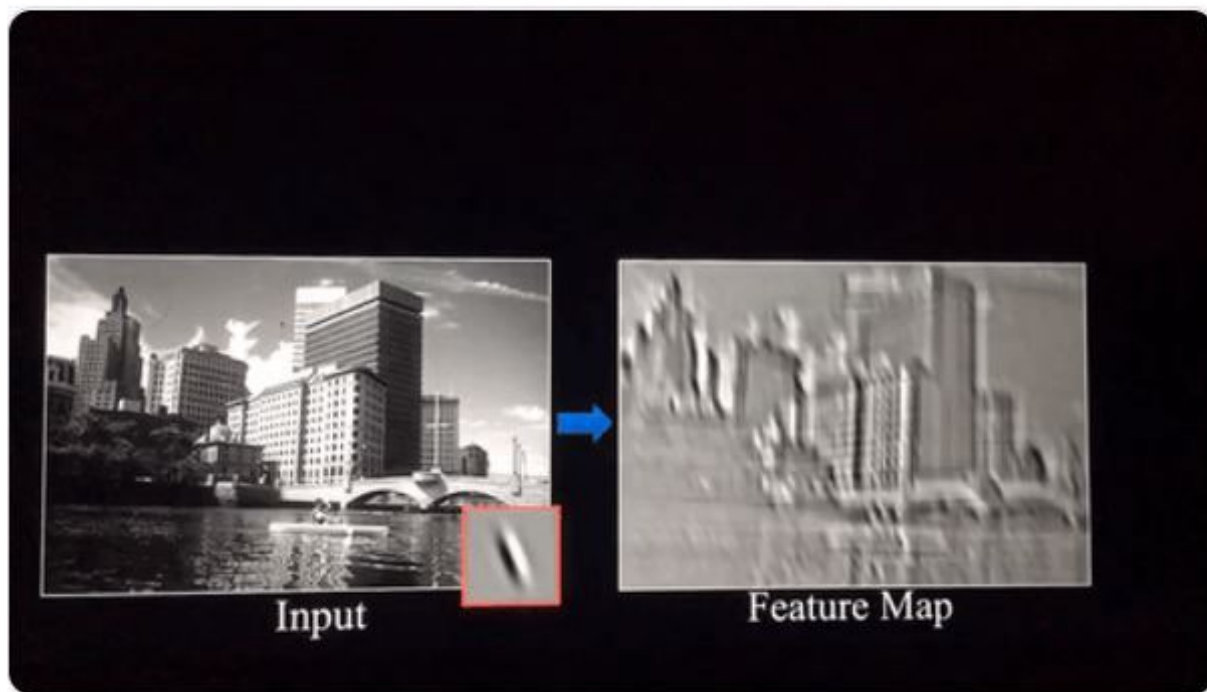


Figure 1.5: Effect of using kernel

The second stage which is called (ReLU) stands for Rectified Linear Unit is used after the convolution stage. ReLU stage can be done by replacing every negative value of pixels with zero to make a nonlinearity process on the features map. The result of this stage is called Rectified feature map. Generally, ReLU is not the only nonlinear function but there are more such the sigmoid and the tanh, but the ReLU function has the best performance. Figure 6.1 shows the effect of nonlinearity (ReLU)

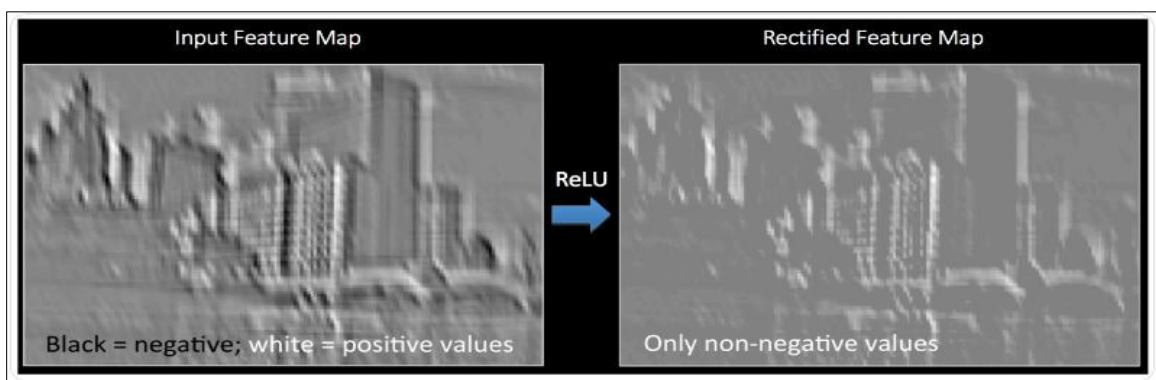


Figure 1.6: Effect of ReLU

At the third stage which is called pooling or sometimes called down sampling reducing and shrinking the dimensions of rectified feature map is done, with keeping all the important information of the feature map. Pooling has many types such as max pooling, average pooling, and sum. Generally, the best one is max pooling since it shows the best performance [2]. Pooling process makes the feature map smaller in size to be managed easily, at the same time reducing the size of parameters and conclusion in the networks and then reducing and controlling the overfitting problem. Also, pooling makes the network vulnerable to small changes and distortion in the input image and leads to any distortion in the input image will not cause or effect on the pooling results since the pooling takes only max values or the average value of the feature map [2]. Figure 7.1 shows the effect of using max pooling or sum pooling.

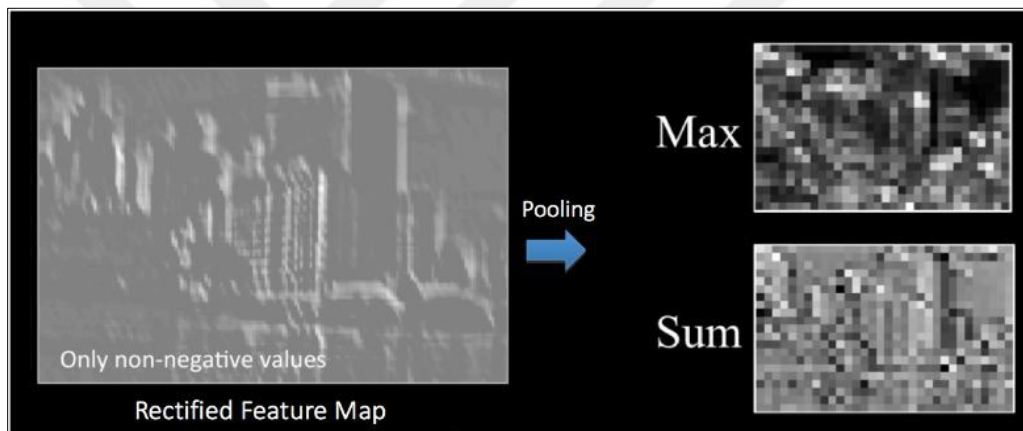


Figure 1.7: Max and sum pooling effect

The outputs of the pooling layer are the input for the fully connected layers which the classification process is done through it .fully connected term refers to that every Neurons in the previous layer is attached to every Neurons in the next layer. Fully connected layer may use many types of activation function, but the best function is called Softmax. The outputs from the convolution and pooling layers are high-level features of the input image. So, the goal of the fully connected layer is to use these features to classify the input image into several classes based on the training of the data. For example, the task of classifying the input image that we showed previously in Figure 3.1 has four possibilities (dog, cat, car, bird) as shown in Figure 8.1. Most of the features extracted from the convolution and pooling layers may be good for the classification task, but compounds with these features may be better. The sum of the output probabilities of the fully connected layer is equal to one, and this is what the softmax layer checks for in the output layer of the fully

connected layer. The softmax layer takes a random vector of real values and then integrate them into a vector with values between zero and one so that their sum equals one [4].



Figure 1.8: classification stage of CNN.

In CNN there is an additional step called backpropagation which use to calculate the Derivative error. Since the CNN will assume random values to the weights and perform the fourth process that we mentioned previously and calculate the probability of each object. Because the weights are assumed randomly in the first step of training so the classification probabilities of outputs will be random too. After that, backpropagation will be used to update weights that used gradient descent and update the values of filters and parameters to reduce the output error ratio, the backpropagation will be repeated until the networks achieve the closest probabilities for each class. It's worth noting the architecture that we explained above belongs to the LeNet type of CNN which is the oldest and simple one but there are many types of convolutional neural networks such as AlexNet (2012), ZFNet (2013), GooLeNet (2014), VGGNet (2014) , ResNet (2015), and DensNet (2016).

1.2 AN OVERVIEW OF OBJECT DETECTION

It is one of the computer vision technologies that allow us to determine the class and location of objects in a certain image or video. This technology can also count the number of objects in a particular scene and determine their exact locations and label them with bounding boxes. In the object detection process, each of the detected elements is identified within a frame, and the location of that element in the given scene is additionally determined. From this we conclude that object identification is a more complex process than classifying images and gives us more information about the image, so we can benefit from it more. The importance of object detection is the ability of the applied algorithm to determine the type and location of a specific object in an image or video, the speed required to do so, as well as the number of objects that this algorithm can detect.

The importance of this we see when applying the detection of objects in practice. For example, in self-driving cars. Real-time object detection is one of the most important keys to the success of intelligent driving systems. These systems need to know and determine the locations of the surrounding objects and recognize traffic lights, all to be able to walk safely and effectively. Object detection can also be applied to surveillance videos and crowd counting, which has benefits for organization and security. Generally, any object detection model consists of three steps which are: region selection, feature extraction, and classification. In the first step scanning of the footage is done with a multi-scale sliding window since the objects may be found in any position in the image with varying sizes and scales.

In the second step visual features are extracted to detect many objects, feature extraction provides semantic and robust representation. The final step which is classification is needed to recognize whether the detected object belongs to any class or not [6]. Many technologies are used in the classification step such as Supported Vector Machine (SVM), AdaBoost, and Deformable Part-based Model (DPM). The frameworks for generic object detection approaches can be subdivided into two types. One can use the typical object detection pipeline, which involves first producing region proposals and then classifying each proposal into multiple item categories. The other approaches object detection as a regression or classification problem, employing a unified framework to immediately obtain final findings (categories and locations). The most common examples of region proposal-based approaches are R-CNN (Region-based Convolutional Neural Networks), Mask R-CNN, FPN, Fast R-CNN, and Faster R-CNN. Whereas regression-classification-based approach examples are YOLO [8], MultiBox, AttentionNet, SSD, G-CNN, YOLOv2, DSSD, and DSOD. Figure 9.1. Illustrates the two approaches to object detection. Object detection is essential to initialize tracking process. It is continually applied in every frame. A common approach for moving object detection is to use temporal information extracted from a sequence of images, for instance by computing inter-frame difference, learning a static background scene model and comparing it with the current scene, or finding high motion areas.

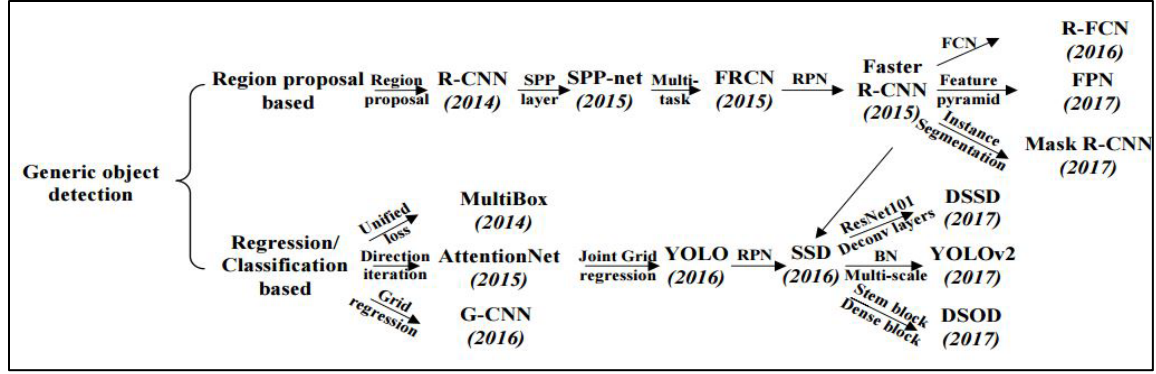


Figure 1.9: Two approaches of object detection

1.3 COMPUTER VISION

The automatic extraction of information from images is known as computer vision. 3D models, camera location, object detection, and recognition, as well as categorizing and searching visual content, are all examples of information. We use a broad definition of computer vision in this work, including image warping, de-noising, and augmented reality. Sometimes computer vision seeks to imitate human vision, other times it focuses on data and statistics, and other times geometry is the key to resolving issues [10]. In other words, computer vision is a multidisciplinary field whose goal is to help computers to comprehend what they see in images and videos. Computer vision is present in most programming languages such as python which contains a huge library of computer vision called (OpenCV), MATLAB and scratch also contain computer vision library. Computers have a great capability to do several complex processes such as facial recognition, optical character recognition, self-driving cars, and sporting analysis in addition to the most important task of object detection. Sometimes computer vision tries to mimic human vision, sometimes it uses a data and statistical approach, and sometimes geometry is the key to solving problems.

In this thesis, we will use a computer vision library of python for accident detection and recognition of car plate numbers. Briefly, the computer vision library in python is responsible for image processing and video analyses.

1.4 RESEARCH QUESTION

This study presents many questions and try to find proper answers to that questions to make robust and powerful models.

I-Is it possible to detect a cars accident precisely?

II- Which algorithm gives accurate results in cars accident detection process?

III- Is it possible to recognize car plate numbers precisely?

IV-Which algorithm gives accurate results in the recognition of plate numbers process?

2. RELATED WORK

In recent years, there has been an increase in the rates of car accidents as a result of the increase in the production of vehicles and the speed of these vehicles, knowing that car accidents constitute about 2.2% of deaths around the world and are classified as the 9th leading cause of death [12]. Therefore, this topic or subject attracted the attention of researchers and scientists in various fields. With the development of artificial intelligence algorithms and the emergence of deep neural networks, many kinds of research based on artificial intelligence and deep learning appeared aimed at detecting accidents to develop rapid monitoring systems capable of distinguishing accidents, and making communication and reporting emergency units to provide Immediate and quick help for victims of car accidents, and because most of the accident victims die due to the lack of Immediate assistance and first aid, especially accidents that occur on highways. Relying on artificial intelligence techniques and deep learning creates faster and more efficient surveillance and control systems than human-managed systems. In recent years, much research has appeared to detect accidents and distinguish car number plates. We will mention some of them and briefly summarize the techniques used in them and the accuracy of these mechanisms with the obstacles and defects in each of this research.

2.1 PREVIOUS METHODS TO DETECT CAR ACCIDENTS

In general, attempts or methods to detect car accidents can be subdivided into two categories: methods that fully depended on the software and AI techniques, and methods that depend on both hardware and AI techniques.

2.1.1 Methods that fully depended on the software and AI techniques

This category of methods uses only artificial intelligence and deep learning algorithms and techniques to extract features from the sequence of frames and forecast whether there was an accident in the frames or not without implementing any hardware or sensor to predict accidents, most of these kinds of methods are based on originally image processing and video analyzing. Also, some of these methods depend on CNN than RNN since the problem of car accidents consider a classification problem which means that the model or methods at the final stage must be able to classify the probability that ranges between two classes (accident or no accident).

Choi [11] et al. propose a model to detect accidents of the cars by adopting on two techniques of deep learning which are CNN and Gated Recurrent Unit (GRU), these techniques are used as classifiers to classify the audio and video data from the cam dashboard of the car. This design proposed by Choi [11] et al. has an important feature that makes it superior to the rest of the proposed systems for detecting car accidents, which is dealing with and processing not only the video during the accident, but also the audio to make the detection system more accurate by dealing with audio and video together. The authors also created an ensemble model as the last step in the detection system to aggregate the features extracted by video and audio. Accidents are detected by the proposed system either by extracting the features from the video frame sequence or by using GRU to extract temporal features from the audio signal, or by the spectrogram image extracted from the audio signal. After that as mentioned earlier, the features extracted by the three methods are collected to infer the occurrence or non-occurrence of car crashes. The authors proposed system achieves an accuracy rate or performance rate (ROC-AUC=89.86) when there are accidents in the video frame or audio signal and achieve an accuracy rate (ROC-AUC=98.60) when there are no accidents in the frames of videos or audio signal. So, the authors study the performance of the model in two situations to evaluate the accuracy ratio of the model which are crash occurrence and non-occurrence cases.

Serrano [13] et al. suggest a model based on visual and temporal features extraction, the model use Inceptionv4 in the first stage to extract visual features after that RNN is used to extract spatial and temporal features from the resulting frame in the first stage, and in the third stage the binary classifications are done by using dense ANN to find out is there an accident in the frame or not. The proposed model shows during tests with a high accuracy rate of 98%. The model has errors in determining accident segments with low illumination (such as nighttime videos) or low resolution and occlusion (low quality video cameras and locations).

Pillai [14] et al. propose a system that involves object detection and then tracking the object and the classification process, for the first one the authors created an algorithm and called it mini-YOLO for detecting the cars which is a knowledge-distilled deep learning model framework with similar accuracy to its YOLO v3 cousin but the smaller model size and computational costs. In the second process the authors used the SORT algorithm to track the detected cars. When a vehicle is involved in an accident, an indicator variable linked to the car's tracks called "damage status" is

set on. The third step, which is a classification step, in which each segmented vehicle picture from the detection stage is sorted into a damaged or undamaged class, classification step also controls this indicator variable. If a vehicle is detected and the system evaluates it as damaged even though the track it is associated to was previously classed as undamaged, the indicator variable is set on. The authors used more than one ML algorithm for the classification process to evaluate the best performance, practical experiments show that SVM as activation functions with radial basis kernel has the highest AUC score of 0.98 with a latency of 12.73 ms.

Using machine vision and Cooperative Vehicle Infrastructure Systems (CVIS), Tian [15] et al. suggested an accident detection approach. To increase the precision of accident detection based on intelligent roadside devices in CVIS, a unique picture dataset called CAD-CVIS is first developed. In particular, CAD-CVIS includes a variety of accident types, meteorological factors, and traffic circumstances at accident locations. After that, the authors create a DNN model based on an accident detection dataset which is called CAD-CVIS. And employ a loss function with dynamic weights and multi-scale feature fusion (MSFF) to enhance the accuracy of tiny object detection. The model proposed by Tian [15] et al. achieves an Average precision (AP) OF 90.02% with a runtime of 0.0461 seconds.

2.1.2 Methods that depend on both hardware and AI techniques

This category utilizes sensors or another physical instrument to obtain data from the outside world and transfer it to the software layer which is mostly artificial intelligence or deep learning algorithms to analyze the data and determine if there is an accident or not, so mostly these methods depend on obtaining information from sensors instead of extract features from the videos or images. Some of the researchers utilize gas pedal sensors, acceleration sensors (when there is an abrupt change in the acceleration of the vehicle that is considered a signal of accident occurrence), distance sensors, and sometimes heartbeat sensors.

Ghosh[16]et al. propose a system involving the Raspberry Pi 3 model acting as a computer with Pi cam attached to it. The authors modify the Inception v3 algorithm In order to incorporate both temporal and spatial information, the LSTM layers were added to the Convolution Network. The authors also train the model with a dataset containing two groups of images one of the images include accidents and the other group does not include accidents. After that, all the software will

be installed into Raspberry Pi 3. So, the Pi cam will capture a given frame and transfer it to the Raspberry Pi 3 and the Inception v3 algorithm will predicate if there is an accident in the frames or not depending on what it learns through the training. The proposed model achieved an accuracy rate of 92.38%.

A deep learning-based Internet of Vehicles (Iivo) system called DeepCrash was suggested by Chang [17] et al. It includes a cloud-based deep learning server, a cloud-based management platform, and In-Vehicle Infotainment (IVI) telematics platform with a vehicle self-collision detection sensor and a front camera. Accident detection data is uploaded to the cloud-based database server for self-collision car collision detection whenever a head-on or single-car collision is detected. The proposed system to detect car accidents consists of three layers: first is the sensing layer then the network and finally the application layer. The sensing layer contains a sensor to monitor the changes in the acceleration of the vehicle to detect accidents. The network contains many kinds of communication technology such as Wi-Fi, 3G, and Bluetooth, the information about accidents is automatically uploaded to the cloud via the communication technology. Chang [17] et al. also made the final layer which is the application layer to manage the information that came from the other layers and also provide deep learning-based data training. it's worth noting that in the application layer the authors modify Inception v3 to fit the problem of car accident detection by using a binary image classifier to forecast if is there an accident in the frame or not. The accuracy of the proposed model is approximately 96%.

Pour [18] et al. adopt on machine learning to design a system that is capable of detecting car accidents by benefits from the inside car sensors. On the Strategic Highway Research Program (SHRP2) Naturalistic Driving Study (NDS) collision data set, five alternative feature extraction methodologies including methods based on feature engineering and feature learning with deep learning are calculated. By using a common framework known as the Pattern Recognition Chain (PRC), which consists of four phases, machine learning may be applied to inside-car data. First, it is important to take into account the availability and type of the selected sensors during data gathering. To prepare the data for further analysis, the second step of the process involves operations like sensor calibration, unit conversion, normalization, and segmentation. The next step in feature extraction is to identify the information from each data segment that is most pertinent to the classes. Classifiers are taught to distinguish between various classes in the feature space in the

final stage. The authors utilize an SVM classifier with a different feature extraction model that is trained on the SHRP2 dataset such as CNN, MLP, HC, LSTM, and AE. The practical experiments show that SVM-classified CNN features beat all other examined methods, including HC, with an accuracy of 85.72%, a weighted F1 score of 84.9%, and an average F1 score of 79.10%.

2.2 PREVIOUS METHODS TO RECOGNIZE CAR LICENSE PLATE NUMBER

Segmentary [19] et al. create a mechanism to extract the character of license plate numbers that involves four basic stages start from uploading the images and pre-processes the image to reduce the noise by converting the image to a binary image and then in the third stage localizations of the plate number is done by splitting the rest of image and just keep the position and form of the plate number which is usually rectangular by using (imputes) library from python. Finally, after the localization of the plate number is done the extraction of character is done by using (Pytesseract) library from python. The accuracy of the proposed mechanism to extract the character of the plate number is about 91.67%.

Rajput [21] et al. propose a model involving three stages which are: image capture that is responsible for image normalization to new dimensions and converting the input image into grayscale. In the second stage, in order to achieve plate localization, the authors used the blue spectrum of the color picture. They then used a wavelet decomposition to calculate the approximation and specifics factor matrices. At the final stage, number and character recognition is done by converting the grayscale localized plate number into a binary image and then extracting the character and numbers. The proposed model shows an accuracy rate of 97%.

A vehicle license plate identification approach based on character-specific Extremal Regions (ERs) and Hybrid Discriminative Restricted Boltzmann Machines (HDRBMs) is suggested by Gou [22] et al. First, upper rotation, vertical edge detection, morphological procedures, and numerous validations are used to conduct coarse license plate identification. Then, from license plate candidates, character-specific ERs are retrieved as character regions. The segmentation of characters and fine to precise LPD are accomplished concurrently after an appropriate choice of ERs. Finally, the characters are recognized using an offline trained HDRBM pattern classifier. The suggested approach is resistant to variations in lighting and weather over a day or 24 hours. The effectiveness of the suggested strategy in complicated traffic settings is shown by experimental

findings on extensive data sets. The authors adopted two criteria to conclude the overall performance which is OVR1 and OVR2 that can be calculated by the following equations:

$$OVR1 = \frac{\text{Number of Correctly Detected and Recognized LPs}}{\text{Number of All Ground Truth LPs}} \quad (1.1)$$

$$OVR2 = LDR \times CRR \quad (1.2)$$

Practical experiments show that the values of OVR1 and OVR2 are 91.9%, and 94.1% respectively.

Hussain [23] et al. suggest a recognition system uses the License Plate Recognition System (LPRS), which makes use of a digital camera to gather vehicle plate numbers. The proposed system basically includes three stages: character segmentation, character recognition, and vehicle license plate localization. The License Plate (LP) detection are done by utilizing a canny edge detection algorithm and Connected Component Analysis (CCA) has been used for character segmentation. The car license plate characters are finally identified and detected using a Multi-Layer Perceptron Artificial Neural Network (MLPANN) model, and the findings are shown as text on GUI. Under various circumstances, the suggested system effectively identifies LP and recognizes Arabic letters in many styles with rates of 96% and 97.872%, successively.

Henry [24] et al. suggest a method consists of three basic stages: transnational LP layout detection, unified character recognition, and LP detection. You only look once (YOLO) networks are the basis of the system in large part. In particular, tiny YOLOv3 was utilized for the first stage whereas YOLOv3-SPP, an issue of YOLOv3 that includes the spatial pyramid pooling (SPP) block [43], was employed for the second stage. For character recognition, YOLOv3-SPP receives the localized LP. The bounding boxes of the forecasted characters are returned by the character recognition network, but the sequence of the LP number is not disclosed. An erroneous sequence is regarded as a false LP number. Henry [24] et al. thus provide a layout identification technique that can extract the proper sequence of LP numbers from international LPs in order to obtain the correct sequence. Henry [24] et al. gathered a dataset of Korean license plates (KarPlate) and made it accessible to the general public. Also, on the LP datasets from five different countries such as South Korea, Taiwan, Greece, the United States, and Croatia, the suggested approach was tested. A small dataset of LPs from seventeen different nations was also gathered to test the efficacy of

the global LP layout identification system. The suggested ALPR system takes an average of 42 ms per picture to extract the LP number. The proposed method by Henry [24] et al. achieves the following accuracies rates: 99.34% for the AOLP dataset, 98.36% for the MediaLab dataset, 95.65% for the Caltech dataset, 98% for the University of Zagreb, 98.59% for the KarPlate dataset.

Zou [25] et al. suggest a method that consists of character feature extraction, license plate feature extraction, and localization of license plate characters. Firstly, the model can completely capture the character from license plates and activate the regional attributes of characters. Next, using Bi-LSTM and the context location data of license plates, find each character on a license plate. After Bi-LSTM placement, One Dimensional -Attention is used to improve relevant character features, decrease unnecessary character features, and achieve effective acquisition of character characteristics of license plates. Zou [25] et al. make experiments on different datasets under different conditions such as bad illumination to prove that the proposed model can operate in any dataset without any change or additional algorithm, in general, the model achieve the following accuracies rates: 99.3% for CCPD-Base dataset, 98.5% for CCPD-DB dataset, 98.6% for CCPD-FN, 96.4%for CCPD-Tilt dataset, 99.3% for CCPD-Weather, and 86.6% for CCPD-Challenge dataset.

Three neural network-based image categorization models are used by Dipu et al. [20] for Bangla character recognition. Inception V3, VGG-16, and Vision Transformer are the models. The BanglaLekha-Isolated dataset, which comprises 98,950 pictures of Bengali characters, was used to train these models. The models provided the following accuracy rate on the test set (Inception v3 98.65%), (VGG-16 97.82%), and (Vision Transformer 96.88%). Compared to the current systems, each of these models is much more accurate. The intricacy and irregularity of Bengali letters, as well as some of the characters' geometric resemblance, all aforementioned reasons form limitations and disadvantages of the proposed method.

3. MATERIAL AND METHODS

3.1 SINGLE STEP OBJECT DETECTION ALGORITHM

It is one branch of object detection; it is called a single step since the prediction of the bounding box and classification occur at the same time. Its consider a highly effective algorithm since it is much faster than other types of object detection. Examples of this kind of object detection are: YOLO, SSD, DetectNet, etc. Most algorithms of this kind consist of three main part which are: backbone, neck, and head (predication step). Examples of backbone types are VGG, Inception v3, DarkNet, GhostNet, etc.

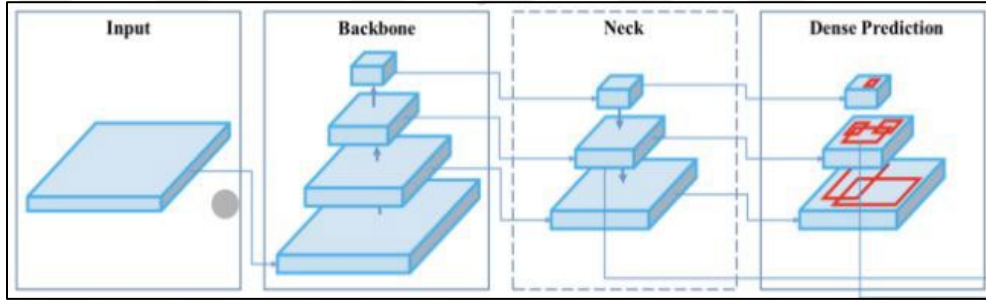


Figure 3.1: Single-Step Detection Architecture

Firstly, the algorithm will create a bounding box for each object in the image and after that, it will divide the image into $S \times S$ cells this will lead to producing a vector for each cell in the grid, and the vectors will contain information about the related cell in the grid such as the probability of absence of the class in the cell, the coordinates of the center of the bounding box (width and height), the coordinates of the bounding box(x and y coordinates) and finally type of the class that found in the cell.

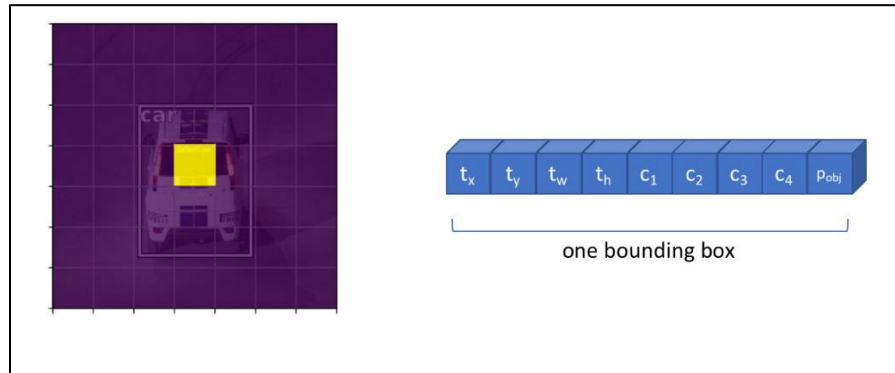


Figure 3.2: Grid Cells and Resulting Vectors [26]

to more easily learn how to extract features from an image, the "backbone" network, which is often pre-trained as an image classifier is used. This is because labeling data for image classification just needs a single label rather than defining bounding box annotations for each image, making it simpler (and less expensive) to do so. As a result, we can train on a resizable labeled dataset (like ImageNet) to discover effective feature representations.

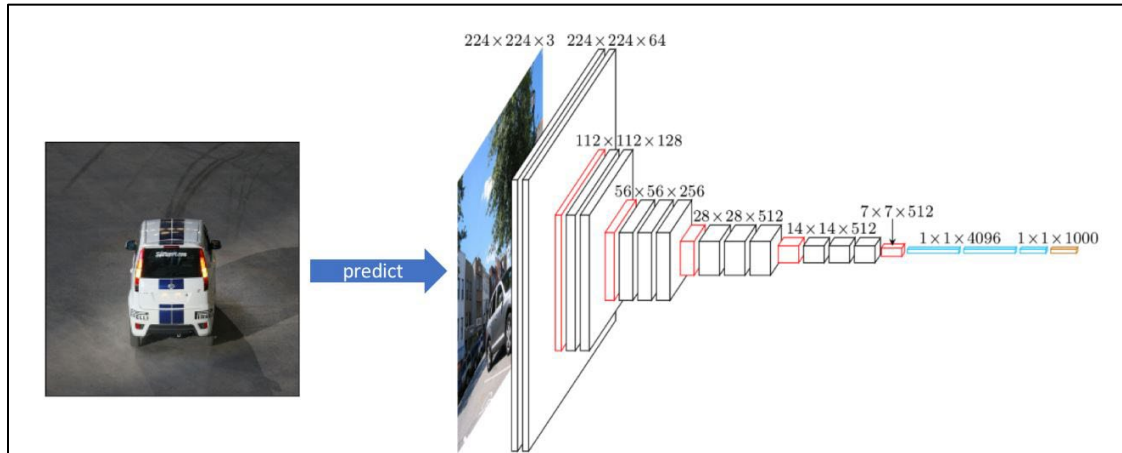


Figure 3.3: VGG Backbone Architecture

The final few layers of the network could be eliminated after pre-training the backbone architecture as an image classifier so that it produces a collection of stacked feature maps that characterize the original image with a low spatial resolution but a high feature (channel) resolution. If we have a $7 \times 7 \times 512$ representation of our observation in the example below. The 512 feature maps each explain various aspects of the original image.

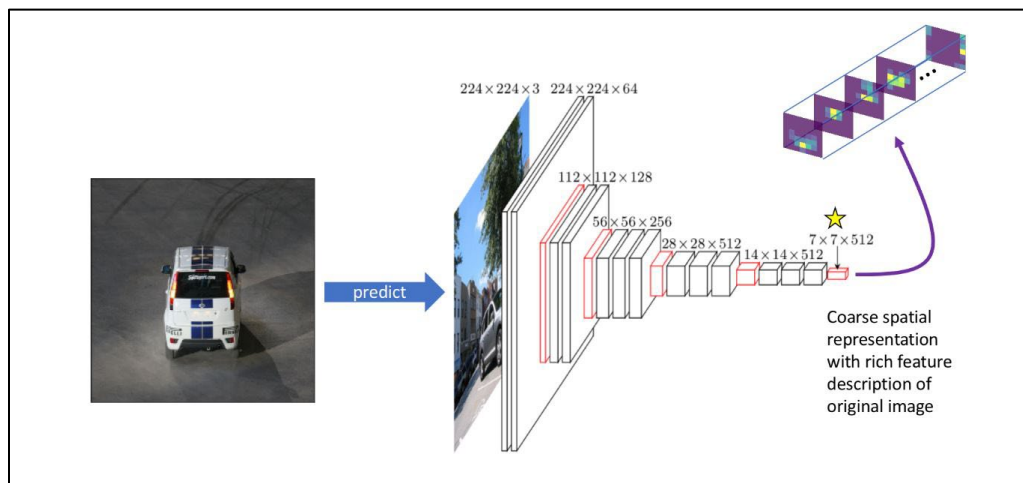


Figure 3.4: Backbone After Pre-Training Process [26]

a second convolutional layer and learn the kernel parameters will be added that integrate the context of all 512 feature maps to produce an activation that corresponds to the grid cell that contains our object in order to detect it.

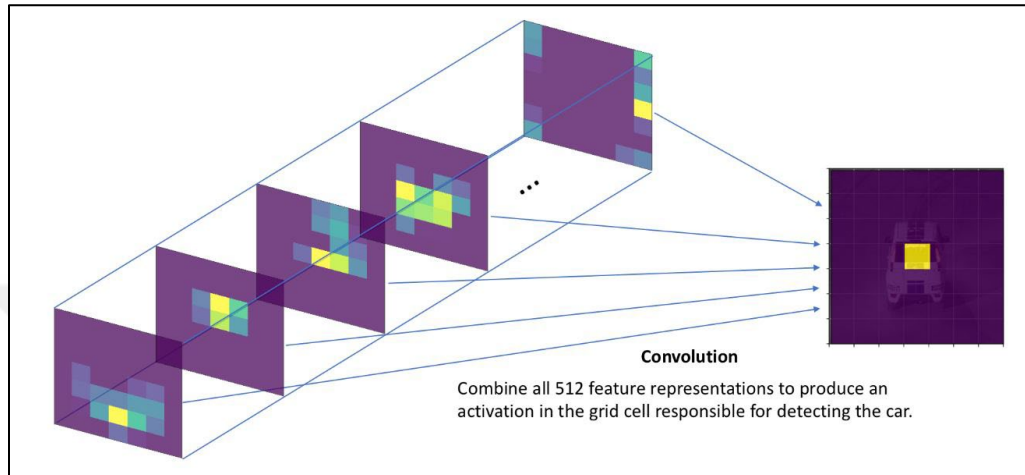


Figure 3.5: Relation Between The Grid Cells And The Features Maps [26]

3.2 TWO STEPS OBJECT DETECTION ALGORITHM

It's a kind of object detection algorithm that makes a detection with two steps, first, it proposes Regions of Interest (RoI) and in the second step classification and localization of each object is done. Actually, most of these algorithms are inspired by the work of R-CNN [28]. These kinds of algorithms consider slow somewhat but very strong algorithms and more accurate than single-step object detection. Examples of this kind of object detection are Fast-R-CNN, Faster R-CNN , and Mask R-CNN.

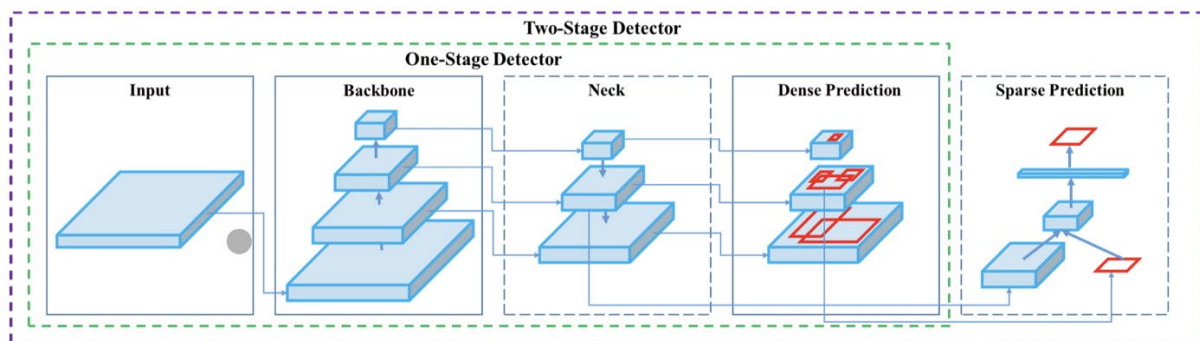


Figure 3.6: Difference in Architecture Between Single-Step And Two-Step Detectors [26]

Most algorithms of this kind of object detection share the same principle of work but with a little bit of difference between them, to understand the principle of work of the two-step object detector we will take for instance Faster R-CNN, in general, Faster R-CNN jointly learns to locate objects spatially and classify them, and it differs from Fast R-CNN by using Region Proposal Network (RPN) instead of using the Selective Search Method that used in Fast R-CNN. Faster R-CNN utilizes RPN to introduce variable benefits for computing object proposals by combining convolution layers with the feature extractor network. Fast R-CNN consists of three stages firstly, the input image has been processed by the convolution layer to extract features and produce a features map, then it's the turn of RPN to create both anchors and regions proposal . The use of RPN makes Faster R-CNN quicker than Fast R-CNN because of the effective use of the CNN. It is worth noting that after receiving the feature map generated by the feature extractor, the RPN model slides a tiny CNN over the feature map to produce a list of object proposals. The network forecast object proposals for k reference boxes (anchors) at each point of the sliding window, with each object proposal consisting of 4 coordinates and a score that calculates the item's probability.

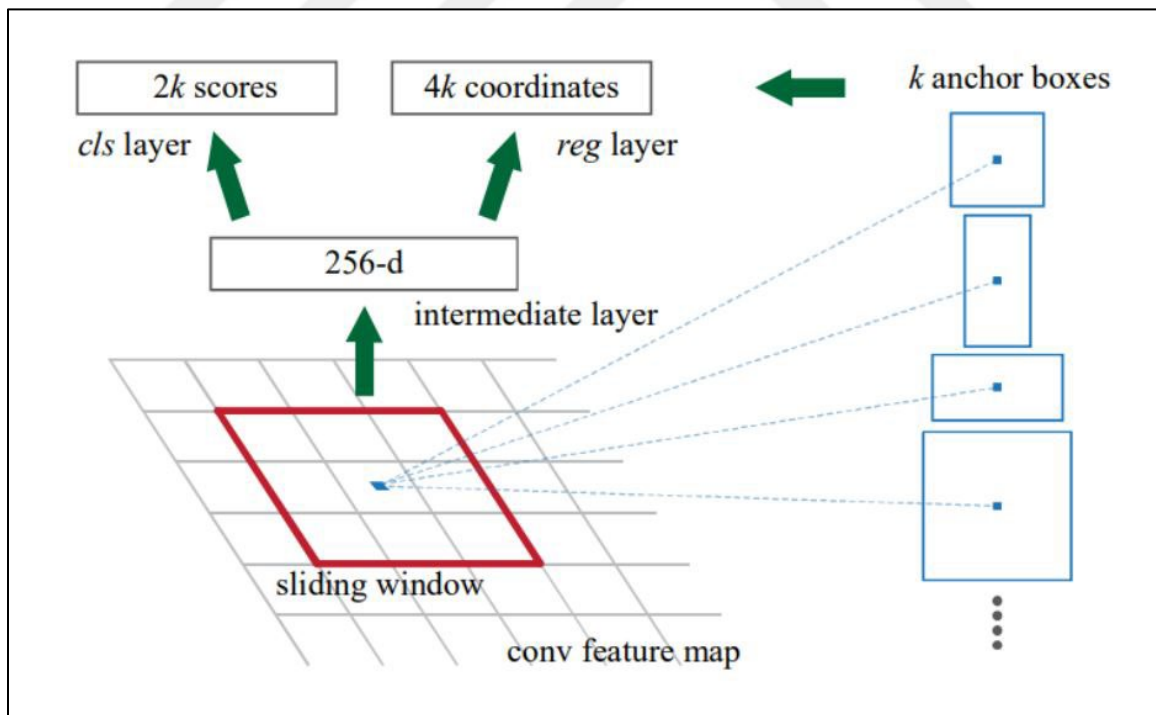


Figure 3.7: The Work Basis of RPN [7]

In order to produce a precise region proposal, the judgment function first evaluates if the anchors are foreground or background, and then borders the anchors accordingly [7]. In the third stage which is the RoI pooling stage, the fully connected layer deals with varying sizes of features map that enter to it to produce fixed size by performing an upsampling process. Finally, the classification and regression layer, the classification layer determines which class an object corresponds to while the regression layer controls the placements of regions of interest (RoIs) to get the final object detection result.

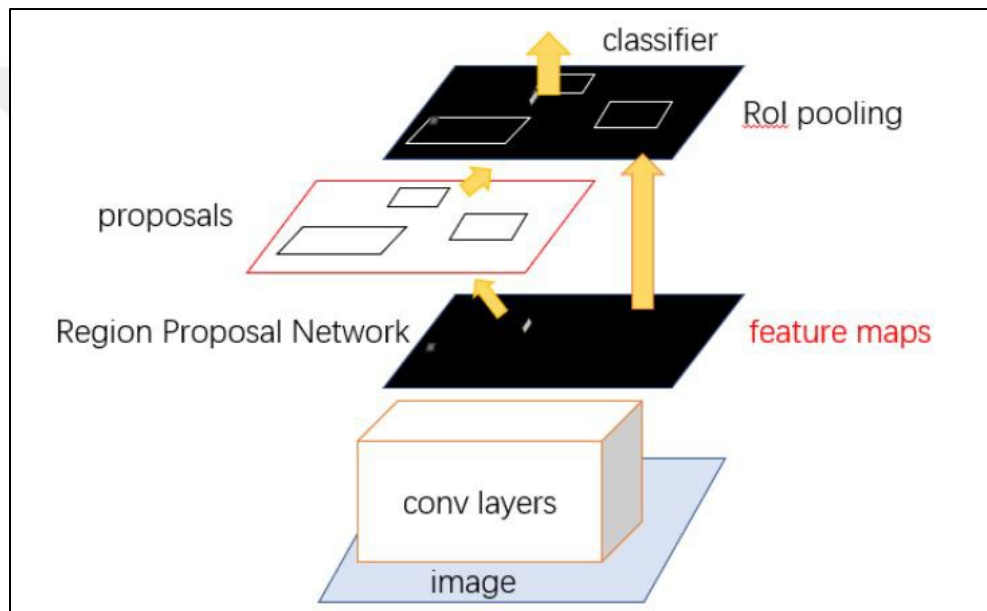


Figure 3.8: Structure Diagram of Faster R-CNN

3.3 CHARACTER RECOGNITION BY EASY OCR

One of the often used applications of deep learning is Optical Character Recognition. Examples include converting handwritten prescriptions, cars plate numbers recognition, converting PDFs or images to text, and verifying signatures. Computer vision is used in the process of OCR to find and analyze the text in photographs. Easy OCR can be implemented using the PyTorch library and Python and was developed by (Jaided AI) a company that deals in optical character recognition services that developed and maintains the Easy OCR package. Easy OCR is a lightweight library to use in comparison to others, as the name indicates. It supports a variety of languages. Additionally, it may be modified to perform better for certain use cases by tweaking different hyper-parameters [32]. Until nowadays Easy OCR can recognize 58 languages, including English,

German, Hindi, Russian, and more. Plans in the future, the Easy OCR maintainers include adding other languages. New OCR algorithms combine Computer Vision and Natural Language Processing (NLP) to detect text from billboards, traffic signs, car plate numbers, and even supermarket product names, enabling them great interpreters and translators. After the text is detected, NLP starts to operate by deciphering the text to clarify the content of the text in the image or document. For text recognition and detection in the real world, algorithms that merge both vision and NLP-based methods have proved very effective. OCR techniques often use algorithms based on the vision that extract textual areas and forecast the bounding box coordinates for those sections. The language processing techniques employ RNNs, LSTMs, and Transformers to decode the feature-based information into textual data after receiving the bounding box data and picture features. The region proposal stage and the language processing stage are the two steps of deep learning-based OCR systems. The initial step in OCR is the identification of text-containing areas in the picture. Convolutional models that recognize text fragments and surround them in bounding boxes are used to accomplish this. Similar to the region proposal network in object detection algorithms like Fast-RCNN, this network's function involves marking and extracting potential areas of interest. Along with information derived from the picture, these areas serve as attention maps and are given to language processing algorithms. The second step is the extraction of data from these areas using NLP-based networks like RNNs and Transformers, which then build meaningful sentences using features provided by the CNN layers. Recent studies have successfully investigated entirely CNN-based algorithms that identify characters directly. These algorithms are particularly helpful to detect text that has little temporal information to convey, such as signboards or car number plates.

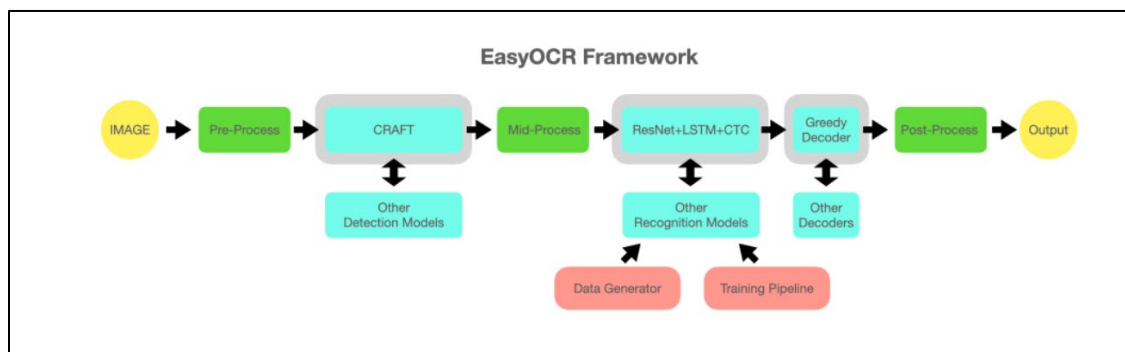


Figure 3.9: Easy OCR Framework

3.4 CHARACTER RECOGNITION BY TESSEARCT

The most well-known and effective OCR library is Tesseract, an optical character recognition engine with open-source code. HP worked on its development from 1984 to 1994, It was more accurately altered and enhanced in 1995 [33]. Late in 2005, HP made Tesseract available as open source. It is quite transportable and less rejection is the main goal rather than precision. There is simply a command base version available for now. Tesseract 3.01 has just been made public and is currently usable. It supports several languages. OCR searches for text and recognizes it on images using artificial intelligence. Tesseract searches for templates in words, letters, and phrases. It employs what is known as an adaptive recognition two-step process. One data step is needed for character recognition, and a second stage is needed to fill in any letters that weren't covered by letters that may fit the word or phrase context. Tesseract actually is a tool for Python. In other words, it will identify and "read" any text that is contained in images. Google's Tesseract-OCR Engine is wrapped in Python-Tesseract. It can read any picture formats supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others, making it handy as a standalone invocation script to Tesseract. Additionally, Python-Tesseract will output the recognized text when used as a script rather than saving it to a file. The Tesseract OCR engine recognizes words using language-specific training data. Like the human brain, OCR algorithms prioritize words and phrases that commonly occur together in a particular language. Therefore, employing training data in the appropriate language will provide the most accurate results. More than 100 languages are supported by Tesseract. The engine may be substantially customized to optimize the detecting algorithms and provide the best outcomes. It's important to note that OCR (generally speaking, pattern recognition) is a tremendously challenging task for computers. Results are rarely perfect, and accuracy drops off quickly as the quality of the input picture rises. Tesseract, however, may often assist in extracting the text from a picture provided the input photos are of acceptable quality. Tesseract works effectively with document pictures that follow the next rules: clean separation between background and foreground text, horizontally aligned, suitably sized, high-quality picture free of noise and blur. The most recent Tesseract 4.0 version offers deep learning-based OCR, which is far more accurate. LSTM network, a kind of recurrent neural network, serves as the foundation for the OCR engine RNN.

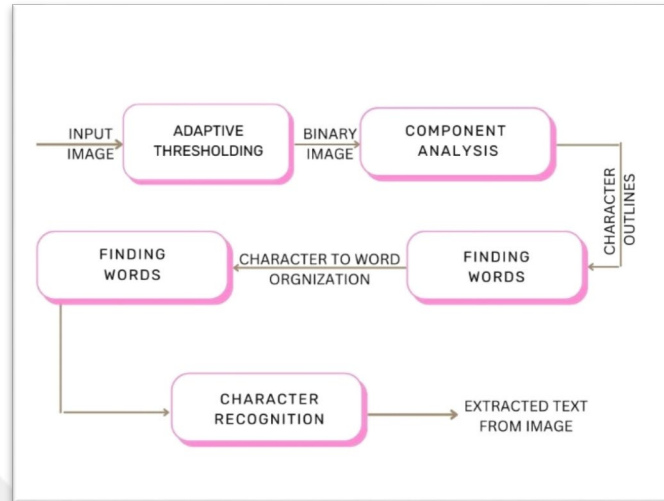


Figure 3.10: Easy OCR Framework

The first stage of Tesseract OCR's step-by-step process is Adaptive Thresholding, which turns the picture into binary images. a component analysis is the next stage, and it is used to extract character patterns. This technique is quite helpful since it does OCR on images with black backgrounds and white text [33]. Most likely the first to provide this sort of processing was Tesseract. The outlines are then transformed into Blobs. Text lines are created from blobs, and the lines and regions are then examined for a defined area or an equal text size. There are both definite and fuzzy spaces used to separate words in the text. A two-pass procedure is then initiated for text recognition. Every word from the text is first tried to be recognized. Every word that has been evaluated favorably is sent to an adaptive classifier as training data. The adaptive classifier aims to identify text more precisely. The last step is utilized to address different problems and extract text from pictures since the adaptive classifier has learned something new after receiving some training data.

3.5 PROPOSED METHODS FOR CARS ACCIDENTS DETECTION

Car accidents occupy the ninth order in the list of most 10 reasons that lead to death. Often it occurs daily and causes injuries and maybe deaths and loss of properties. Despite the fact that low- and middle-income nations have around 60% of the world's automobiles, they account for 93% of all traffic deaths, additionally, most nations lose 3% of their gross national product to road accidents. For that reason, this topic attracts the attention of both researchers and developers to design the mechanisms or methods to reduce car accidents and also provide emergency first aid to the victims. since the delay to provide rapid help to the victims of cars accidents leads to deaths in

most cases, some researchers exploit the emergence of artificial intelligence especially deep learning techniques and computer vision to design models capable of detecting car accidents in real-time and calling the nearest emergency units to provide the rapid first aid to the victims which contribute to saving lives and somehow reduce the number of victims also. Our proposed method is basically based on utilizing deep learning techniques such as neural networks and computer vision. We design our model by using (OpenCV) a computer vision library in the python language program with the help of the PyTorch framework. The main objective is to detect car accidents by utilizing two types of detectors which are single-step and two-step detection algorithms and then we compare between these algorithms to evaluate which type is preferred for the aforementioned task. From the single-step algorithm, we chose YOLO v5, and from the other type, we choose Faster R-CNN [7]. there are so many reasons for choosing these algorithms, for the YOLOv5 algorithm consider very easy to train on the custom dataset it accepts anything out of the 80 classes of the COCO dataset, and also it has very good speed and good accuracy compared to other types of object detection algorithms. In general YOLO family considers less complicated in architecture compared to two-step detectors. In contrast, we choose Faster R-CNN [7] just to compare with YOLOv5 in terms of accuracy and speed, and consumption of resources. In the following paragraphs, we will explain our proposed method to detect accidents for each type of detector. In 2015 Redmon [8] et al. publish a new object detection technique called YOLO which stands for (You Look Only Once) it is simple in architecture than previous object detection methods, in this novel approach The object detection is framed by the authors as a regression issue to spatially separated bounding boxes and related class probabilities, Bounding boxes and class probabilities are directly predicted by a single neural network from whole images in a single evaluation [8]. Since YOLO can operate at 45 FPS its capable of processing online videos with a delay of approximately 25ms and also has 57.9 % mAP on the VOC 2012 dataset. Five predictions make up each bounding box: x , y , w , h , and confidence. The centroid of the box concerning the boundaries of the grid cell is represented by the (x, y) coordinates. Relative to the whole image, the width and height are forecasted. Last but not least, the confidence prediction shows the IOU between any ground truth boxes and the forecasted box. Using the PASCAL VOC detection dataset, the author evaluates the performance of the model. While the fully connected layers of the network forecast the output probabilities and coordinates, the first convolutional layers extract features from the image [8]. The GoogLeNet method that was used for image classification served

as a basis for the network design [35]. It includes 2 fully connected layers after 24 convolutional layers. the architecture is simplified by using 1×1 reduction layers followed by 3×3 convolutional layers in replacement of the inception modules utilized by GoogLeNet [8].in the next year Redmon [9] et al. release the second version of YOLO that include multiple enhancements on the previous version. Yolov2 is frequently referred to as YOLO9000 since it can recognize over 9000 different item types. The model can also operate at various sizes, making it simple to balance accuracy and speed. On VOC 2007, YOLOv2 achieves 76.8 mAP at 67 FPS. YOLOv2 achieves 78.6 mAP at 40 FPS, exceeding many techniques like Faster R-CNN with ResNet and SSD [9]. Redmon [9] et al. train YOLOv2 on the COCO detection dataset at the same time together with the ImageNet classification dataset, since the proposed method is capable of jointly training on object detection and classification, the model can predict detections for object classes that don't have labeled detection data. It's worth noting that YOLOv2 utilizes a logistic activation to limit and balance between the predicated coordinates associated with the position of the grid cells and made it range between 0 and 1. Yolov2 solely uses convolutional and pooling layers, and its input resolution is 416×416 . Since the network can change input dimensional size for every ten batches, YOLOv2 can handle images of various sizes [9]. Due to the model's 32-fold downsampling, the data were taken from multiples of 32, such as 320, 352... and 608. 320 by 320 is the smallest choice, while 608 by 608 is the biggest. The network was modified to include that dimension and training continued. YOLOv2 includes a novel technique to deal with small objects by adding 26×26 layer beside the original feature map that has a resolution of 13×13 that deal with big objects to the backbone of the YOLOv2 is DarkNet-19, it has 19 convolutional layers and 5 maximum pooling layers [9]. In 2018 after two years of YOLOv2 releases Redmon [34] et al. publish YOLOv3 with improvements on the prior version that include using independent logistic classifiers and binary cross-entropy loss for the class predictions during training. With the input 320×320 resolution, the model mean average precision is about 28.2 with a runtime of 22 ms .The backbone of the model is called DarkNet-53 since it contains 53 convolution layers each of which has a 3×3 size after that there is a 1×1 convolutional layer. Bochkovskiy [36] et al. in 2020 published a paper about YOLOv4 that contains the implementation of new feature techniques such as Mosaic Data Argumentation, Mish Activation, Self-Adversarial-Training (SAT), Cross-mini-Batch Normalization (CmBN), Weighted-Residual-Connections (WRC), Cross Stage Partial connections (CSP), and DropBlock regularization. The model obtains the average precision of

43.5% on the MS COCO dataset at 65 fps. Bochkovski [36] et al. choose the CSPDarknet53 as a backbone, SPP additional module [43], Path-Aggregation Network (PANet) [39] as the neck, and YOLOv3 head as the rest architecture of the model. After two months to release of the fourth version of YOLO, Glenn Jocher the CEO of the (Ultralytics company) releases YOLOv5 as an open source code on the GitHub website without any paper with too slightly difference from YOLOv4. These differences can be summarized in that YOLOv5 is based on the PyTorch framework, unlike YOLOv4 that based on DarkNet which is written in the C language program. A data loader that augments data online is used by YOLOv5 to feed training data through with each training batch. Scaling, color space changes, and mosaic augmentation are the three types of augmentations that the data loader does. The most innovative of them is mosaic data augmentation, which creates four random-ratio panels from four photos. Mosaic augmentation is particularly helpful for the well-known COCO object detection benchmark because it teaches the model how to deal with the well-known "small object problem," which is the issue that little items are harder to identify than bigger ones [36]. YOLOv5 creates model configuration in .YAML rather than Darknet's .cfg files. The key difference between these two file types is that the YAML file is compressed so that it only contains information about the network's various levels, which are then multiplied by the number of layers in the block. Both YOLOv4 and YOLOv5 utilize the same kind of neck which is PANet [39] that is used to aggregate features.

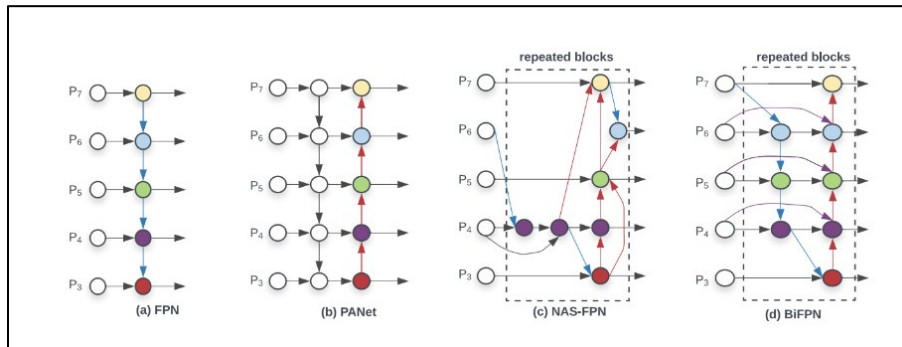


Figure 3.11 Feature network design – (a) FPN [38] introduces a top-down pathway to fuse multi-scale features from level 3 to 7 (P₃ - P₇), (b) PANet [39] adds a bottom-up pathway on top of FPN, (c) NAS-FPN [40] use neural architecture search to find an irregular feature network topology and then repeatedly apply the same block; (d) BiFPN [37] with better accuracy and efficiency trade-offs.

It's worth noting that YOLOv5 takes a fewer time in training than YOLOv4 which the training length is based on the number of iterations. For example, YOLOv5 needs only 14.5 minutes to complete the training stage with 200 epochs.

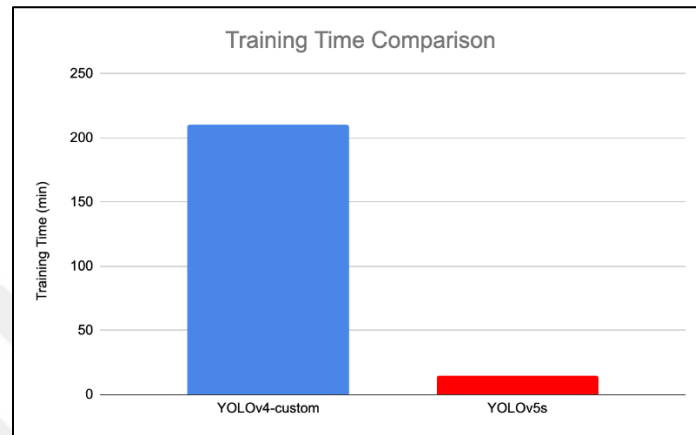


Figure 3.12 Training Time Comparison Between YOLOv4 And YOLOv5

Many kinds of research have been published to compare the performance between YOLOv4 and YOLOv5 in the evaluation stage, and many of them have been proven that both versions have the same mean average precision.

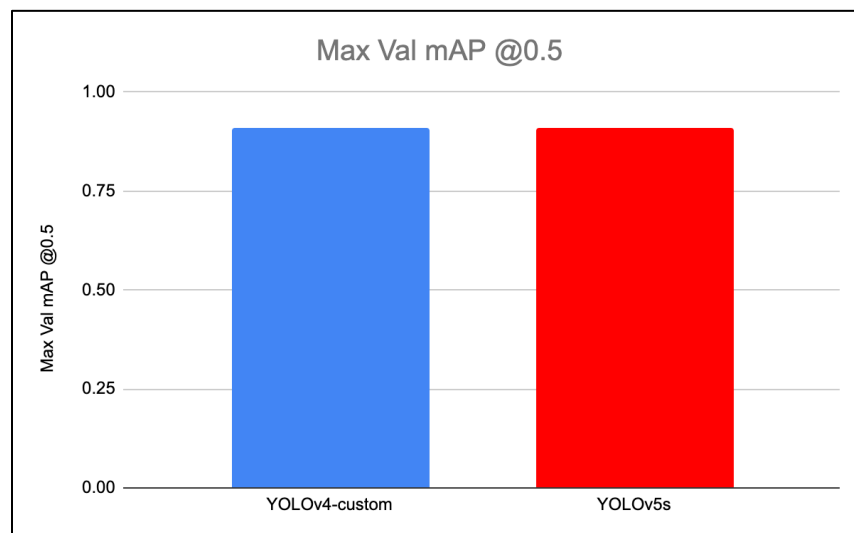


Figure 3.13 Both Models Have Similar mAP

But for the speed, the YOLOv4 outperforms YOLOv5 which shows latency on the MS COCO object detection dataset.

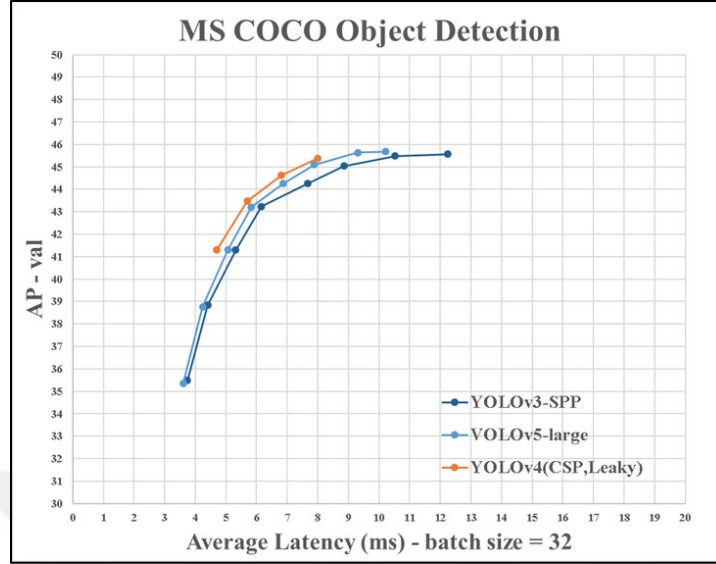


Figure 3.14 Performance Comparison on the MS COCO Benchmark

In contrast, R-CNN is the most common type of two-step detector. It is a set of machine learning models utilized in computer vision and image processing. Any R- CNN's objective is to identify objects in any input picture and define borders around them since it was specifically created for object detection. The R-CNN model uses a process called selective search to collect details about the region of interest from an input picture [28]. The bounds of the rectangles may be used to indicate a region of interest. There can be more than two thousand regions of interest, depending on the circumstance. To create output features, CNN uses this region of interest. The SVM classifier is then used for these output features to classify the given items under a region of interest [28].

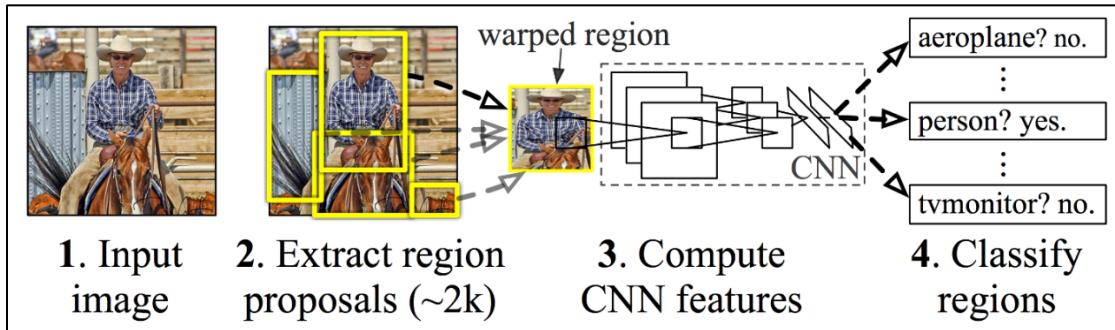


Figure 3.15 R-CNN System Overview [28]

The methods of an R-CNN when utilizing it to detect an item are shown in the figure 3.15 [28]. It includes the use of the region extraction technique to extract regions of interest from a picture using the R-CNN. Up to 2000 more regions may be added. The model controls the size to be fitted for the CNN for each region of interest. The CNN calculates the features of the region, and SVM classifiers classify the items that are shown in the region [28]. Any object detection algorithm may use a variety of object positioning techniques. One method, which is refer to as an exhaustive search approach, involves applying sliding filters of various sizes to the picture in order to extract the objects from it. An exhaustive search approach will need more processing as the number of filters or windows rises. Exhaustive search is used by the selective search algorithm, but in addition to utilizing it by itself, it also works with the segmentation of the colors seen in the picture [28] selective search is a technique that splits objects from a picture by giving the objects distinct colors. This approach begins by creating several little windows or filters and then grows the region using the greedy algorithm. Then it finds the colors that are similar throughout the areas and combines them. An essential element of object localization is the use of selective search techniques. An extracted object will go through the following three processes in object detection after localization which include the Using of warp, a CNN to extract features, and finally classification [28]. Following region selection, the image containing regions is sent through a CNN model, which separates the objects from the region. It takes some time, if not the majority of the time, reshaping the image since its size must be fixed in accordance with CNN's capabilities. In a simple R-CNN, the region is wrapped into images that are $227 \times 227 \times 3$ [28]. The extracted object of 4096 dimensions will be handled from a wrapped input for CNN. An SVM classifier is used in the simplest R-CNN to classify various objects according to their class.

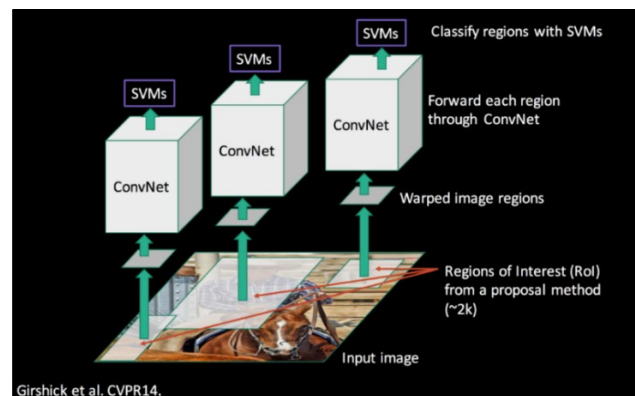


Figure 3.16 The Whole Process of R-CNN [28]

In fast R-CNN, ROI pooling uses a single feature map for all the regions rather than doing maximum pooling. The ROI pooling layer employs maximum pooling to transform the features while warping ROIs into a single layer. A fast R-CNN is thought of as an improvement over the SPPNet since max pooling is also effective and used in Fast R-CNN. It simply creates one layer, not layers in the form of a pyramid in SPPNet [29].

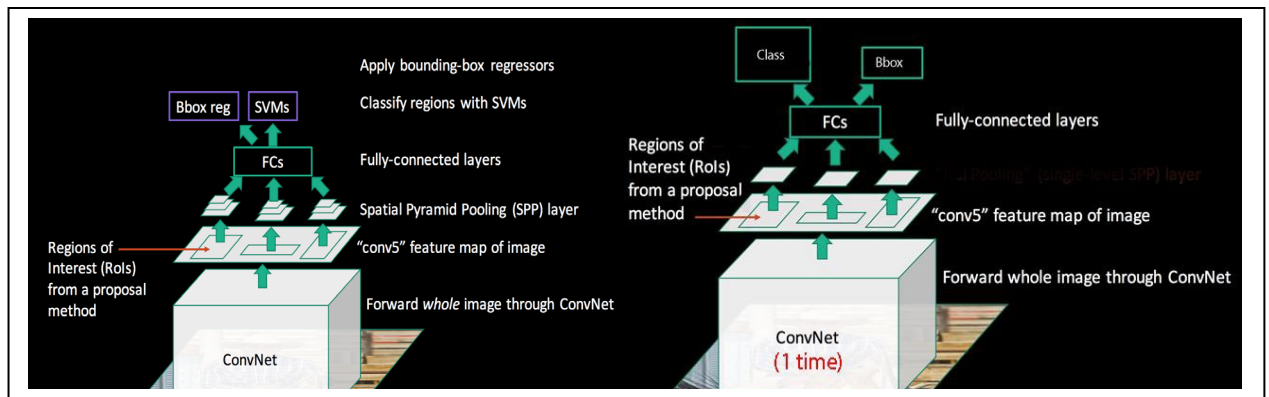


Figure 3.17 SPPNet vs Fast R-CNN [29]

For that reason, Fast R-CNN is faster than SPPNet. It's worth noting that both SPPNet and Fast R-CNN did not use any methods to choose or extract the regions of interest, which is the difference between the aforementioned algorithms and Faster R-CNN. The sets of regions are created using a region proposal approach using faster R-CNN [7]. Faster R-CNN, which is referred to as the regional proposal network, has an additional CNN for acquiring the regional proposal. The proposal network in the training area uses the feature map as input and produces region proposals [7]. And the ROI pooling layer receives these ideas for further action.

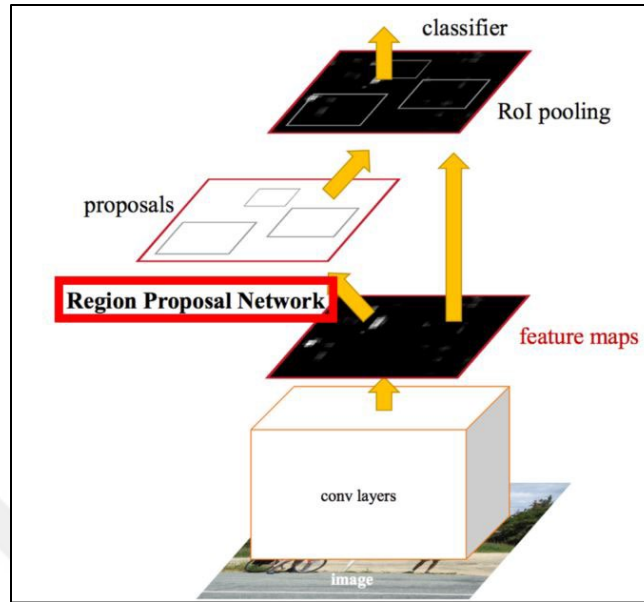


Figure 3.18 Faster R-CNN Approach [7]

Table 3.1: Comparison Between Different Two-Step Detectors Models

Metrics	R-CNN	Fast R-CNN	Faster R-CNN
Region Proposal Method	Selective Search	Selective Search	Region Proposal Network
Prediction Time	40-50 Sec	2 Sec	0.2 Sec
Computation	High Computation Time	High Computation Time	Low Computation Time
The mAP on Pascal VOC 2007 test dataset (%)	58.5	66.9	69.9
The mAP on Pascal VOC 2012 test dataset (%)	53.3	65.7	67

3.5.1 Accidents Detection by YOLO v5

Car accident detection considers a very complicated task because the model that is designed to detect the accidents should be fast and accurate at the same time. In order to increase the accuracy of our model, since it depends on the quality of the dataset and the number of images in the dataset, more images lead to more accuracy for that reason we must take care of our dataset. We choose the YOLOv5 model with Pytorch framework and OpenCV library in python that is responsible for

the computer vision part of deep learning, first of all, we made a dataset consisting of 3000 images including various accidents images with different angles of vision and different situations of accidents, beside that we insert some images of cars without accidents or in normal cases to make our model capable of detecting both car and car accident. Before we start the first step which is the training step, we must modify our dataset which we call it (CAD-DATASET) by using the WORKFLOW website to edit labels on each image and after that we export in the format that suits YOLOv5. We divide our dataset into 75% for training and 15% for validation and 10% for test step.



Figure 3.19 Instances from CAD-DATASET

For the training step, we choose the batch size equal to 32 and with 300 epochs, the number of samples that are processed before the model is changed is the batch size. The quantity of complete iterations across the training dataset is the number of epochs. A batch must have a minimum size of one and a maximum size that is less than or equal to the number of samples in the training dataset. so instead of processing one image per one epoch, we can make more than one image for example 4 or 16, or 32 to be processed in one iteration, this will lead to reducing the computations and make the training step shorter in time. We training step complete the model will be capable to detect car accidents if occur in any frame of the video.

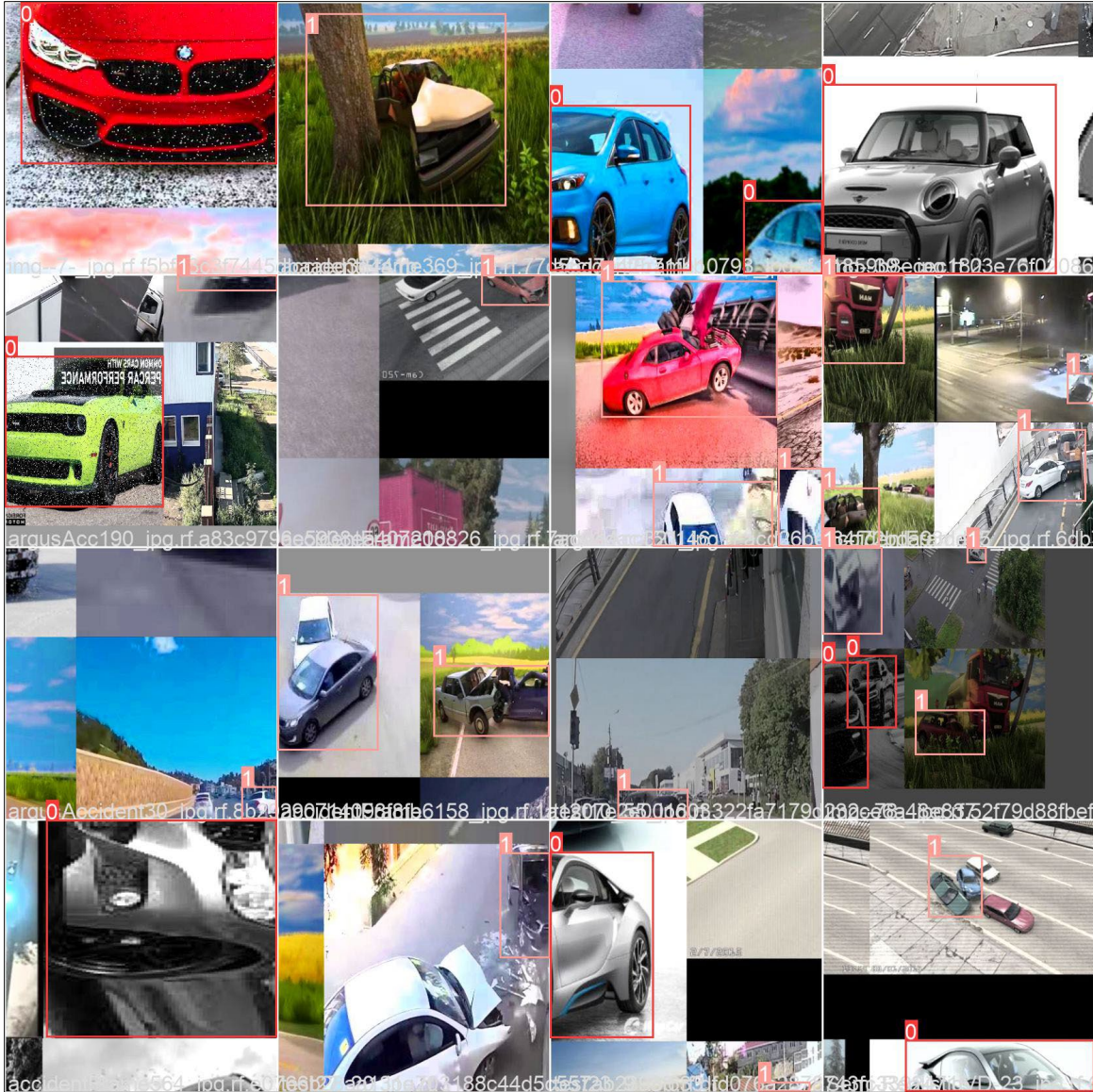


Figure 3.21 Train Batch (1)

Before starting to explain the detection step, it's worth noting that we choose the YOLOV5 v6.0 model. The v6.0 backbone is released by Ultralytics also on the GitHub website. In general, the model consists of three-part which are: a New CSP Darknet -53 as the backbone (feature extractor) that contain 53 convolution layer and the second part include the SPPF, a New CSP-PANet as the neck of the model which is responsible for feature fusion, and the final part is the header the same YOLOv3 head that produce the detection result [42]. This backbone eliminates the redundant gradient information seen in big backbones and incorporates gradient change into feature maps, which speeds up inference, improves accuracy, and shrinks the size of the model by reducing the

parameters [42]. It boosts the information flow by using the path aggregation network (PANet) as a neck. A novel feature pyramid network (FPN) with several bottom-up and top-down layers is adopted by PANet. This enhances the model's transmission of low-level features. Lower layer localization is improved by PANet, increasing the object's accuracy in its localization[42].YOLOv6.0 model includes the implementation of SPPF that consider an improved version of the original SPP [43], but it is mathematically identical with fewer FLOPs. The original SPP is utilized to get rid of the network's fixed size restriction, this we will get the same size of output with different input image sizes [42]. While C3 is made up of a module cascaded by many bottlenecks and three convolution layers. The head of YOLO5 produces three distinct feature map outputs in order to perform multiscale forecasting. Additionally, it improves the model's ability to anticipate tiny to big objects effectively.

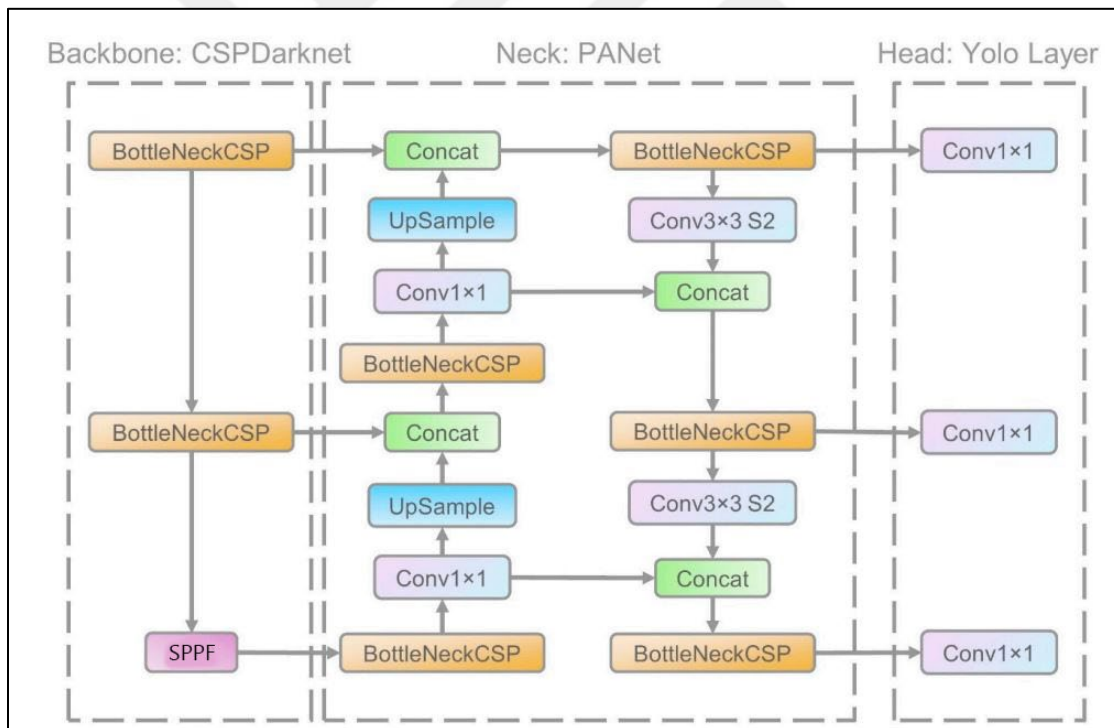


Figure 3.22 New CSP DarkNet Backbone

In figure 3.22 the bottleneck is added to reduce the number of feature maps in the network, which, otherwise, tend to increase in each layer, this is achieved by using 1x1 convolutions with fewer output channels than input channels [42]. The preceding layer fusion is upsampled at the closest

node using upsample. Concat, a layer that slices other layers, is used to slice the layer before it. The last three Conv2d are detecting modules utilized at the network's head.

After the training and validation step to evaluate our model performance, its detection step turns, in this step after the car is detected and the model creates a bounding box around the object (cars) we will use the criteria to detect the accidents. If there is any intersection between bounding boxes and the intersection was more than 30% and below 85% the model will indicate that there is an accident in the frame between the cars. Through the following equations, the model will conclude if there is an accident by calculating the intersection between the bounding boxes.

$$(2 \times |c1.x - c2.x| < c1.w + c2.w) \wedge (2 \times |c1.y + c2.y| < c1.h + c2.h) \quad (3.1)$$

In the aforementioned equation, (c1) and (c2) refer to two bounding boxes of two objects (cars), and the (x) and (y) refer to the centroid position of the bounding boxes while (w) and (h) refer to the dimensions of the bounding boxes. So, after the cars were detected by YOLOv5 and surrounded by the predicated bounding box if any intersection occur at any frame and the intersection rate is between 30% and 85% this means an accident occur in the frame.

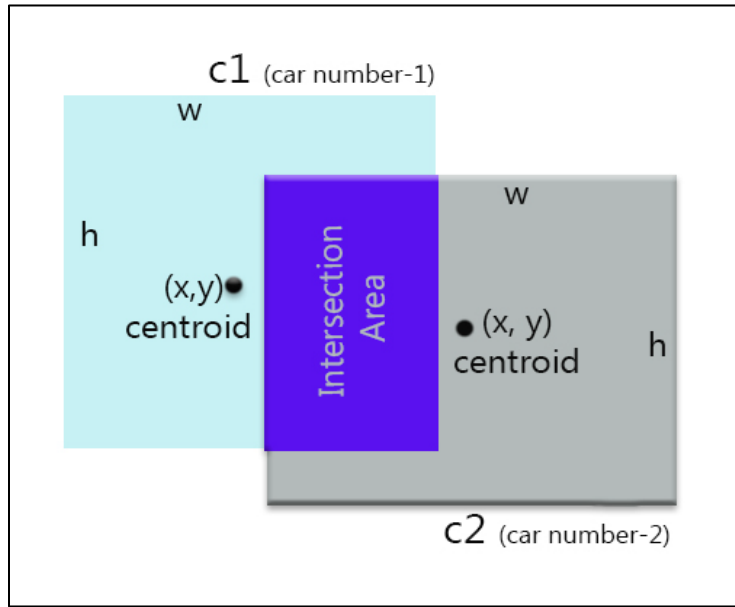


Figure 3.23 Proposed Intersection Between Bounding Boxes

In brief, our model uses two methods to detect cars accidents one of them is by the trained YOLOv5 itself that trained on the dataset contains the images of cars accidents and cars in normal

situations and by processing a sequence of frames, the model will detect the accident. While the second method will exploit the intersection between predicated bounding boxes around the detected vehicles to conclude if there is an accident or not. So, if one of the methods fails to detect the accident the second method will detect it, which increases the accuracy of our proposed model.

3.5.2 Accidents Detection by Faster R-CNN

To determine which type of detector is better to perform our task which is car accident detection we will implement the two-step detector Faster R-CNN. This type is more accurate but slower than a single-step detector. As a procedure, we will follow the same procedures in the methods based on YOLOv5, we will train Faster R-CNN on the dataset (CAD-DATASET) with changing the format of the dataset to make it suitable for the Faster R-CNN, and then we will use again intersection between bounding boxes to detect the accident. It's worth noting that we use two types of backbones ResNet50 with FPN and the other one is MobileNet v3 large with FPN. The purpose behind training two types of backbone for Faster R-CNN is to estimate which is useful and more accurate for our task. MobileNet v3 is the third version of the architecture that supports the image processing abilities of many well-known mobile apps. Additionally, the design has been implemented into well-known frameworks like TensorFlow Lite [44]. MobileNetV3's primary innovation is the use of AutoML to identify the ideal neural network design for a particular issue, this contrasts with the hand-crafted architecture of early versions. MobileNetV3 specifically makes use of MnasNet and NetAdapt as two AutoML approaches. In order to choose the best configuration from a collection of discrete options, MobileNetV3 first looks for a coarse architecture using MnasNet. The model then uses NetAdapt, a complementary method that eliminates underutilized activation channels incrementally, to fine-tune the design [44]. Squeeze-and-excitation blocks being included in the basic design of MobileNetV3 is another innovative concept. The main goal of the squeeze-and-excitation blocks is to explicitly express the relationships between the channels of a network's convolutional features in order to enhance the quality of the representations that are generated by the network [44]. In order to do this, a provided feature recalibration technique that enables the network to learn how to selectively highlight informative features and suppress less helpful ones by learning to utilize global information. Squeeze-and-excitation blocks were included in the search space for MobileNetV3's design, which results in more robust architectures than MobileNetV2's [44].

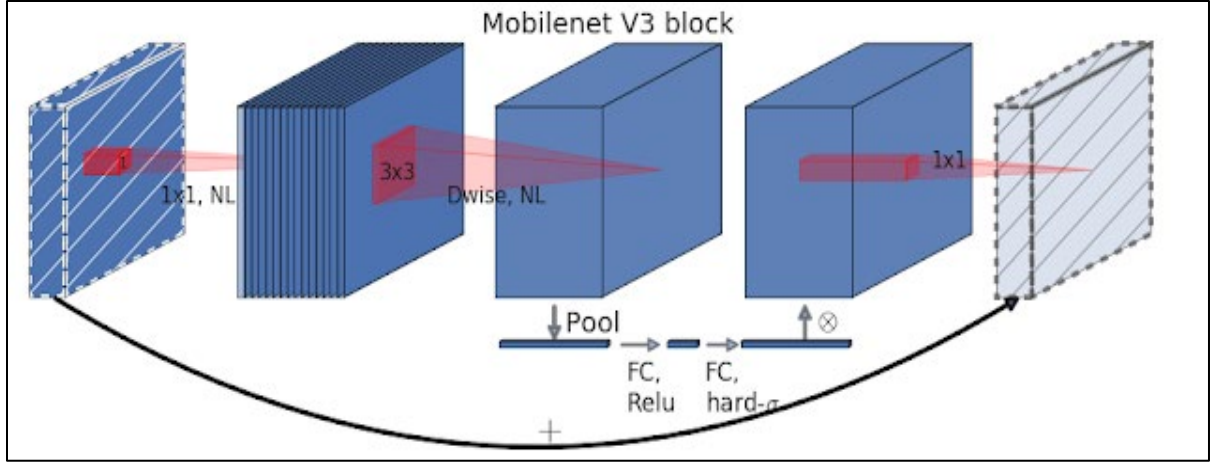


Figure 3.24 Architecture of MobileNet v3 [44]

The redesigning of a few of the costly architectural layers in MobileNetV3 was an interesting optimization. Some of the layers in MobileNetV2 were essential to the models' accuracy but also added latency-related issues. MobileNetV3 was able to eliminate three costly layers of its previous design without reducing accuracy by implementing several fundamental enhancements [44].

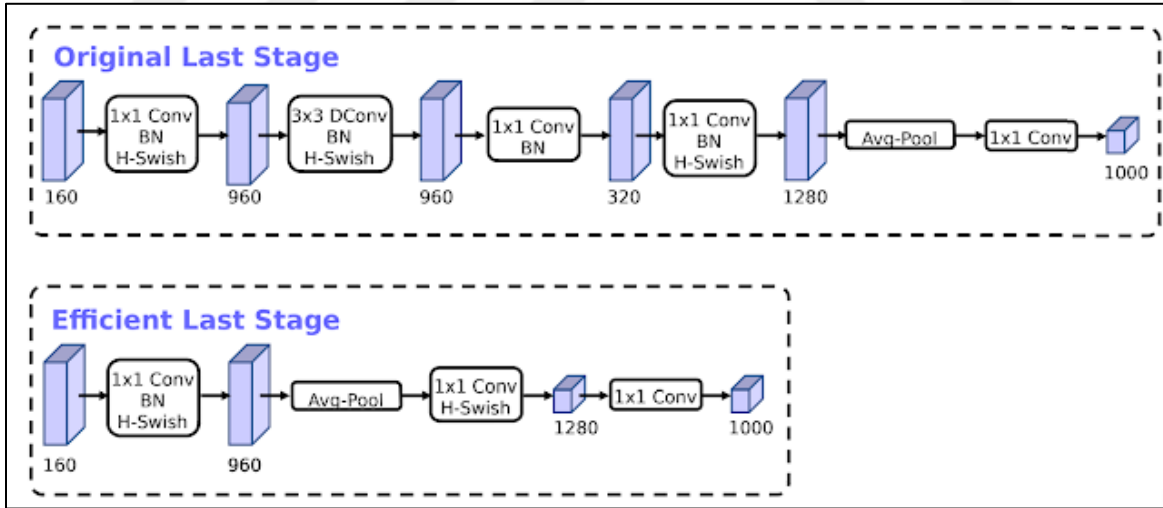


Figure 3.25 Original last stage in MobileNet v2 and efficient last stage in MobileNetv3 [44]

We choose MobileNet v3 large as our backbone. The following table (3.2) illustrates the components of the MobileNet v3 large where (SE) refers to Squeeze-And-Excite in that block, while NL refers to the type of nonlinearity that is used, whereas HS, RE, NBN,s refer to h-switch and ReLU, no batch normalization, and stride successively [44].

Table 3.2: Components of MobileNet v3 [44]

Input	Operator	Exp size	out	SE	NL	S
$224^2 \times 3$	Conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	Used	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	Used	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	Used	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	Used	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	Used	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	Used	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	Used	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	Used	HS	1
$7^2 \times 160$	Conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	Pool, 7x7	-	-	-	-	1
$1^2 \times 960$	Conv2d, 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	Conv2d, 1x1, NBN	-	K	-	-	1

In 2015 He [45] et al. introduce the novel neural network called ResNet which stands for Residual Network. Deep residual networks are convolutional neural networks (CNNs) with more than 50 layers, like the well-known ResNet-50 model, ResNet is a kind of Artificial Neural Network (ANN) that builds a network by layering residual blocks on top of one another [45]. This model was very effective, as shown by the fact that its ensemble took first place in the ILSVRC 2015

classification competition with a little 3.57% error. There are other versions of ResNet that use the same basic idea but have varying amounts of layers. ResNet50 can operate with 50 neural network layers. The strength of this kind of neural network is because of the idea of "skip connections," which is at the core of the residual blocks [45]. These skip connections operate in two different ways. First, they resolve the problem of the disappearing gradient by creating a different shortcut for the gradient to use. They also provide the model with the ability to learn an identity function. By doing this, it is made sure that the model's upper layer doesn't perform any worse than its lower ones. In summary, the layers learn identity functions much faster due to the residual blocks. ResNet hence reduces the number of errors while increasing the effectiveness of DNN with additional neural layers. To put it another way, the skip connections combine the results of earlier layers with the results of stacked layers, enabling the training of far deeper networks than was previously feasible [45]. The difference between ResNet50 and prior models like ResNet34 is the modification in the bottleneck structure. The building block in this instance was changed into a bottleneck design. Instead of the preceding 2 layers, ResNet50 utilized a stack of 3 [45]. In order to create the Resnet 50 design, each of the 2-layer blocks in Resnet34 was changed to a 3-layer bottleneck block. Compared to the 34-layer in the ResNet34 model, ResNet50 has substantially greater accuracy. The performance of the 50-layer ResNet is 3.8 billion FLOPS.

We will train Faster R-CNN with each type of backbones (MobileNet v3 large, ResNet50-FPN) by the same dataset (CAD-DATASET). It's worth noting that ResNet Backbone includes the FPN techniques as a feature extractor that help us to solve the problem of occlusion and also enables the construction of higher resolution layers from a semantically rich layer top down. we assign max iteration equal to 1500 and the step is scaled to (1000,1500) and about the (model of ROI, batch size per image) which is a parameter that gathers a subset of a proposal from RPN to evaluate the classes and register the losses that occur through the training process, we set the aforementioned parameter equal to 64. So, after the training finishes our model will be able to detect the accident in any frame. If the Faster R-CNN will miss the accident, we will reuse the intersection between bounding boxes in this model also to help the pre-trained model to detect the accidents by employing the same principle in the equation (3.1). So, if the Faster R-CNN doesn't detect the accident the intersection between bounding boxes will detect it. That's will increase the accuracy of the model.

3.6 PROPOSED METHODS FOR CARS PLATE NUMBER RECOGNITION

3.6.1 Car Plate Number Recognition by Using Easy OCR

Its traditional methods to extract characters and numbers from the image. Easy OCR consists of many kinds of processes to analyze the image and locate the character in the image. To perform our task, we need to utilize python and OpenCV library in addition to Easy OCR library in python and Imutils library which include a series of functions to perform image processing tasks such as rotation, transformation, detecting edges, and many other functions. The first step of our model to recognize cars plate numbers is by converting the input image from colored RGB image into a gray image, which facilitates the operation of the next processes. The second step is using Bilateral Filter to reduce the noise in the images and made the edges more sharp and clear Canny edge detector [47] to detect the edge in the image by using a multistage algorithm including the finding intensity gradient of the image and applying non-max suppression to remove unnecessary pixels that don't contain any edges and after that implementing Hysteresis Thresholding that decides which are all edges are really edges and which are not [47].



Figure 3.26 Input Image



Figure 3.27 Gary Image (step 1)



Figure 3.28 The Effect of Bilateral Filter and Canny Edge Detection (step 2)

In the third step finding and capturing and storing the contours is done by utilizing both OpenCV and Imutils libraries. Whereas the fourth step includes the use of shape descriptors by implementing the Contour Approximation function which approximates a contour shape to another shape with fewer number of vertices depending upon the precision we specify that specify a contour around the plate number. The next is to apply a mask to the image by using a matrix of zeros that converts the image into binary. After that in the next step draw contours that we get it from the previous step by using the OpenCV library. after drawing the contours, we will apply an arithmetic logic operation to the resulting drawing contours, to make all the regions around the specified contours (plate number) turn in black color, we choose the inverse of (AND) operation on the drawing contours to make the plate numbers more clear and easy to recognize.

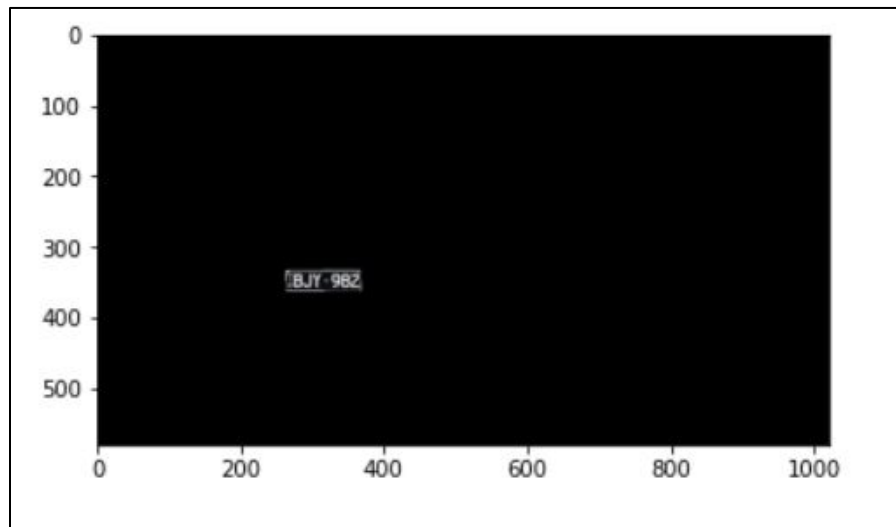


Figure 3.29 The Resulted Image After (step3, 4)

After that in step five, we must make a matching process between the resulting image and the original image to locate the position of the plate number in the original image, the purpose behind that is to the region surrounded by the drawing contour or doesn't contain a black pixel to produce and cropped image that contains only plate number. This is very important since Easy OCR cannot usually extract characters from the whole image or image with a big size.

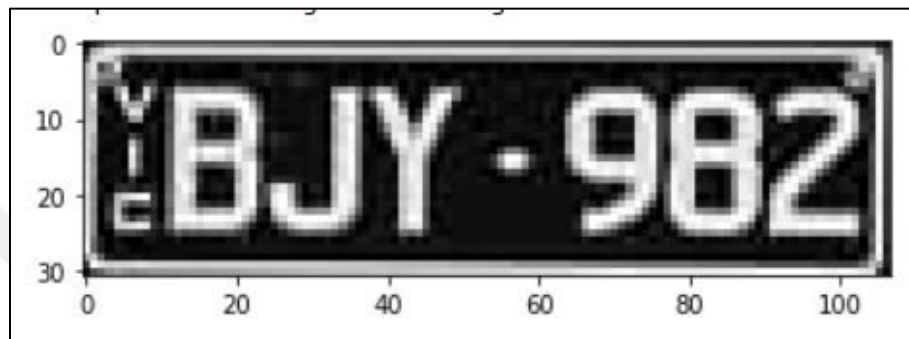


Figure 3.30 The Resulted Image After (step 5)

The final step includes the employ of the Easy OCR function in python to analyze and extract the characters from the corrupted image, then after the characters are extracted, we will use the OpenCV library to put and write the extracted characters and then draw a bounding box around the plate number in the original image.

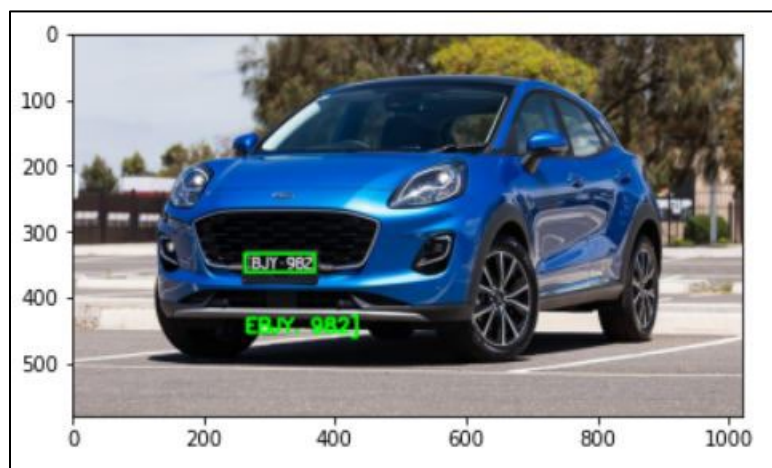


Figure 3.31 final resulting image

3.6.2 Car Plate Number Recognition by Using YOLOv7 with Pytesseract

In this task, we will implement two models together one of them is YOLOv7 which we use it to detect the plate number in the input image or frame, and the second model is responsible for extracting and recognizing the characters from the detected plate number by using Pytesseract in python.

In July 2022 Wang [48] et al. released YOLOv7 which outperforms both YOLOv5 and YOLOv6 in terms of speed and accuracy. The improvements in YOLOv7 include Extended Efficient Layer Aggregation Network (E-ELAN) and Model Scaling for Concatenation based Models in the architectural side beside the improvement in the trainable (Bag of freebies) that include use of Planned Re-Parameterized convolution and Coarse for auxiliary and fine for lead loss [48]. Bag of Freebies term was first introduced in [36] which means a series of steps that can be taken to improve the model's performance without increasing latency at inference time. By using expand, shuffle, and merge cardinality to accomplish the capacity to constantly increase the learning ability of the network without breaking the original gradient path, the YOLOv7 E-ELAN architecture helps the model learn better [48]. Compound model scaling for a concatenation-based model is introduced in YOLOv7. The model's original design attributes may be preserved by using the compound scaling approach, keeping the ideal structure unchanged. Compound model scaling operates as follows: For instance, changing a computational block's depth factor necessitates altering the block's output channel. The transition layers are then subjected to width factor scaling with the same degree of modification [48]. Re-parameterization is a technique used after training to improve the model it increases the training time but improves the inference results, there are two types of re-parametrizations used to finalize models, which are: first-way model level by using different training data but the same settings and taking the average of the weights of models at different epochs. And also train multiple models and then average their weights to obtain the final model, the second way is module level ensemble the model training process is split into multiple modules, and the outputs are ensembled to obtain the final model [48]. YOLOv7 isn't restricted to just one head. It includes the lead head in charge of producing the output while the auxiliary head is utilized to support middle-layer training. additionally, and to improve deep network training, a Label Assigner method was developed that assigns soft labels after taking into account ground truth and network prediction outcomes [48]. Reliable soft labels use calculation and optimization

methods that also take into account the quality and distribution of prediction output along with the ground truth, as opposed to traditional label assignment that directly refers to the ground truth to generate hard labels based on prescribed rules [48].

First of all, as we mentioned before we will use YOLOv7 to detect the plate number. We train YOLOv7 on our dataset that we call it (CLPN) which contains 3525 images consisting of images of plate numbers with different angles. We divide our dataset into 75% for training, 15% for validation, and 10% for the test step. We trained the YOLOv7 with 300 epochs and a batch size equal to 16. After we complete the training, we get a model capable of detecting the plate number.



Figure 3.32 Output Image from YOLOv7

After YOLOv7 detects the plate number, the second model takes the output image from YOLOv7 to extract the characters from the detected plate number by using Pytesseract. Since Pytesseract cannot extract characters from the whole image or high-resolution image, so we need to pre-process the image. First, we need to convert the image from an RGB image into a colored image, and then by using OpenCV we find the region of interest that includes the bounding box (plate number) by finding the contours of the detected object.



Figure 3.33 Region of Interest Image

After that Pytesearct receives the region of interest image and applies the dilation and erosion process with the help of the OpenCV library to remove noise. After that convert the grayscale image into an inversed binary image to make the edge of the characters more sharpened and then it turns the Pytesearct to extract and convert the character from image to string and then print the extracted character.

```
C:\Users\alimo\PycharmProjects\pythonProject1\venv\  
--- Start recognize text from image ---  
VAVACARS  
  
----- Done -----
```

Figure 3.34 Extracted Characters from Image

4. EXPERIMENTS AND DISCUSSIONS

4.1 CAR ACCIDENT DETECTION RESULTS AND DISCUSSIONS

As we mention before, we train two different types of object detectors one of them is single step detector (YOLOv5) and the other is a two-step detector (Faster R-CNN) on the same dataset (CAD-DATASET), the purpose is to choose which type is suitable and better for this task. We use python to design our model especially the computer vision library which is responsible for image processing by deep learning and other artificial intelligence techniques. First of all, we choose Darknet-53 as the backbone for the YOLOv5 to extract features and produce a feature map. As we mention before we train YOLOv5 with 300 epochs and batch size =32. After training, we use the Tensor Board tool to evaluate and plot our model performance and precision. The following table (4.1) illustrates the achieved values of precision and recall for each class and also the overall precision and recall of the model for the training step.

Table 4.1: Performance Evaluation of the Training Step for YOLOv5

	Precision	Recall	mAP@0.5
All classes	0.933	0.835	0.923
car	0.982	0.857	0.956
Car-accident	0.885	0.812	0.89

After the training step, as in the table (4.1) we can see that our overall model achieves a mean average precision of 92% at (IoU) threshold equal to 0.5. For the car-accident class, our model achieves 98% mAP with precision equal to 88%. It is worth noting that the precision equation is:

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \quad (4.1)$$

Whereas the recall value can be calculated from the following equation:

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \quad (4.2)$$

Finally, the F1 score is defined as the harmonic mean between precision and recall, we can get the F1 score of the training step from the following equation:

$$F1 = \frac{2 * (precision * recall)}{precision + recall} \quad (4.3)$$

It's worth noting that our model achieves an F1 score equal to 0.9 after the training step based on the values from the table (4.1). The following figure (4.1) shows our model precision at each epoch during both the training and validation step.

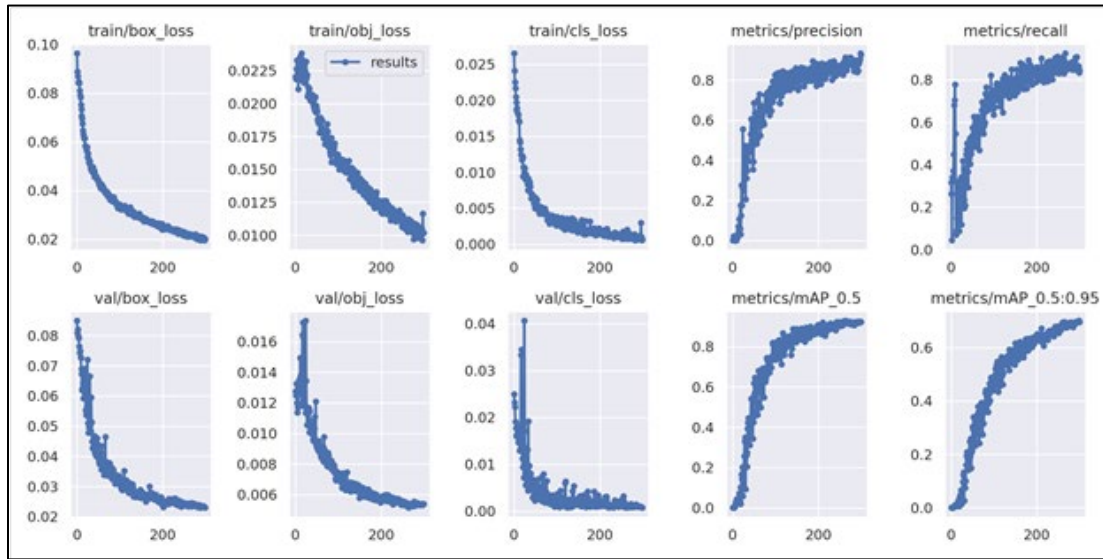


Figure 4.1 Results of Training and Validation Step

From figure (4.1) we can conclude that the precision of our model increases with an increased number of epochs this means more number of epochs lead to more precision, whereas the losses of bounding boxes and object classes losses decreased with the increased number of epochs.

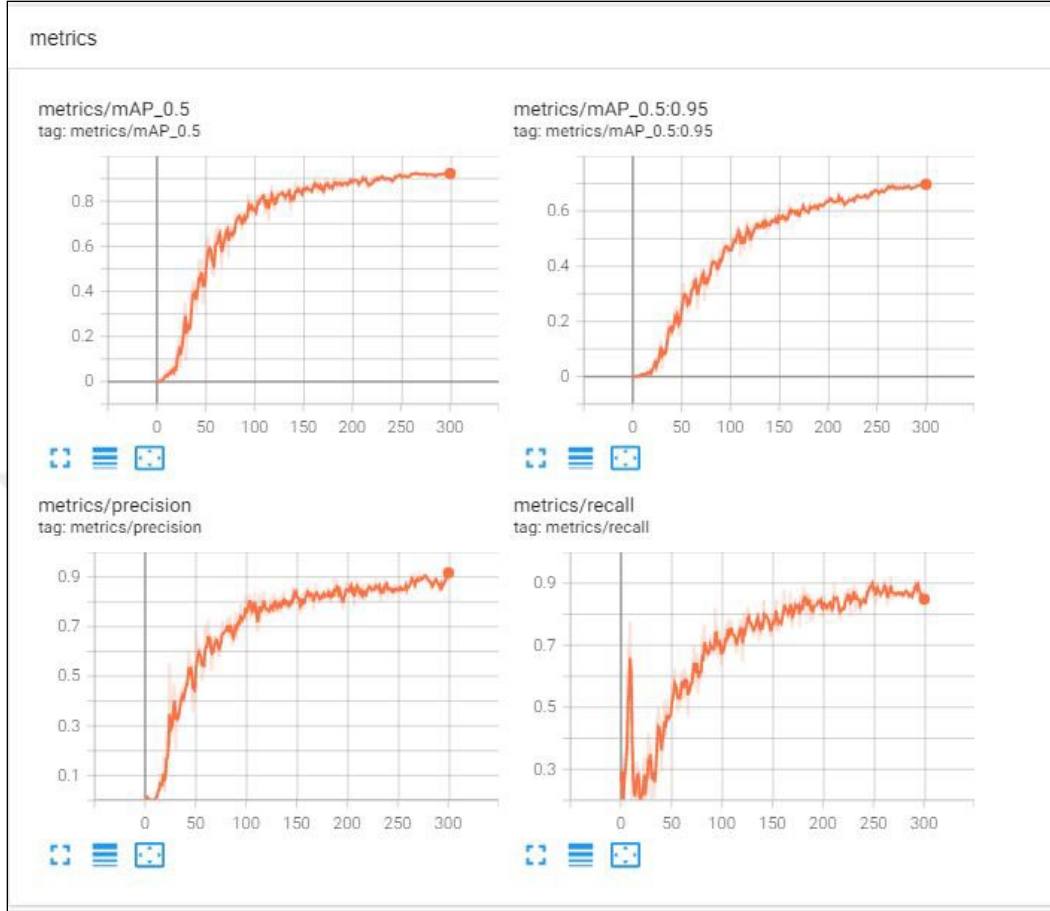


Figure 4.2 metric achieved after training and validation step with 300 epochs

Accuracy shows how many times the model was correct overall. Precision is how good the model is at predicting a specific category. Recall shows how many times the model was able to detect a specific category. In the detection step, we adopt the following equation to calculate the accuracy of our model.

$$accuracy = \frac{\text{number of corret predication}}{\text{total number of samples}} \quad (4.4)$$

So, from this equation, we calculate the accuracy of our model which was equal to 76.2% after training the model with 300 epochs. So, if the trained YOLOv5 failed to detect the accident the intersection between bounding boxes will detect the accident this will increase our model performance. The following figures show the detection of accidents by both the pre-trained model itself or by the intersection of bounding boxes

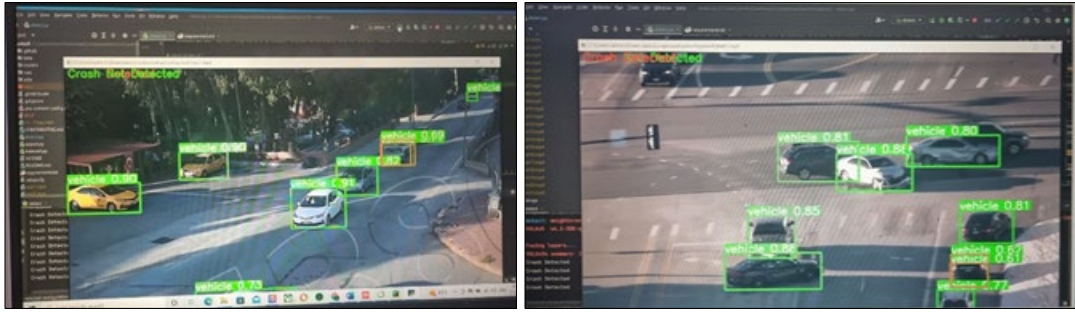


Figure 4.3 Accident Detected by Intersection Between Bounding Box



Figure 4.4 Accident Detected by YOLOv5 Itself

For the second model which is Faster R-CNN, we train the model with two different backbones which are (MobileNet v3 large) and (ResNet50-FPN) by using the same dataset. We train our model with 1500 iteration steps for Faster R-CNN with ResNet50-FPN and 2000 iteration steps for Faster R-CNN with MobileNet v3 large since MobileNet v3 need more step than ResNet to get better performance. We get the following results after the training step

Table 4.2: Performance Evaluation of the Training Step for Faster R-CNN

	AP@0.50:95	AP@0.5	Recall@0.50:95
ResNet 50-FPN	0.977	0.977	0.989
MobileNet v3 large	0.9	0.967	0.923

From table (4.2) we can conclude that ResNet50 outperforms MobileNet v3 in terms of average precision and recall. Even though the iteration steps for MobileNet v3 are more than the iteration steps for ResNet50 in the training stage. The following figures (4.5) show the metric for each backbone.

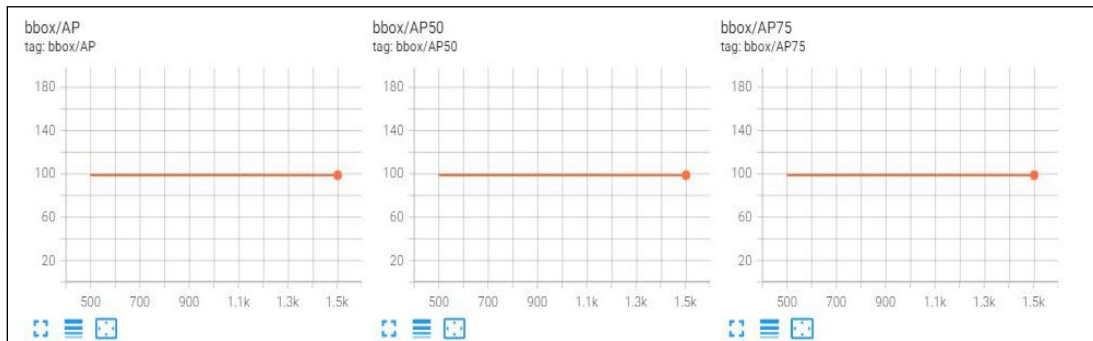


Figure 4.5 Metric of Faster R-CNN with ResNet50-FPN

From the figure (4.5) we can see that the average precision value remains stable from iteration step 500 to the last step of the training at different IoU values. This makes the model more accurate and more stable in the detection step. figure (4.6) and figure (4.7) show that both recall and mAP values become close to 1 at the final iteration in the training step for MobileNet v3 large backbone.

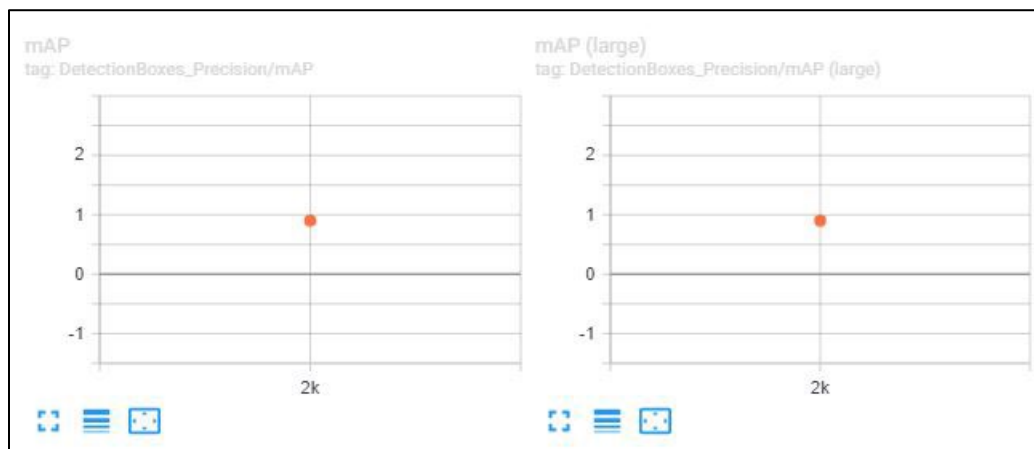


Figure 4.6 Overall mAP and mAP for Large Object for Faster R-CNN (mobilev3 large)

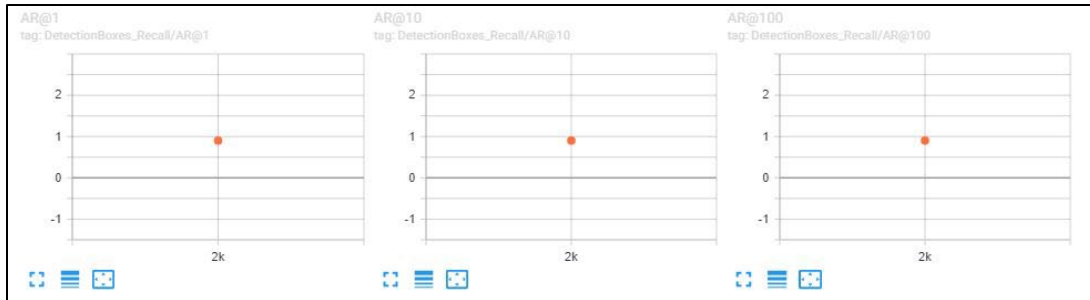


Figure 4.7 Recall at different thresholds

The following figure (4.8) show the accidents that detect by the pre-trained Faster R-CNN itself and the accident that detected by intersection between bounding boxes.

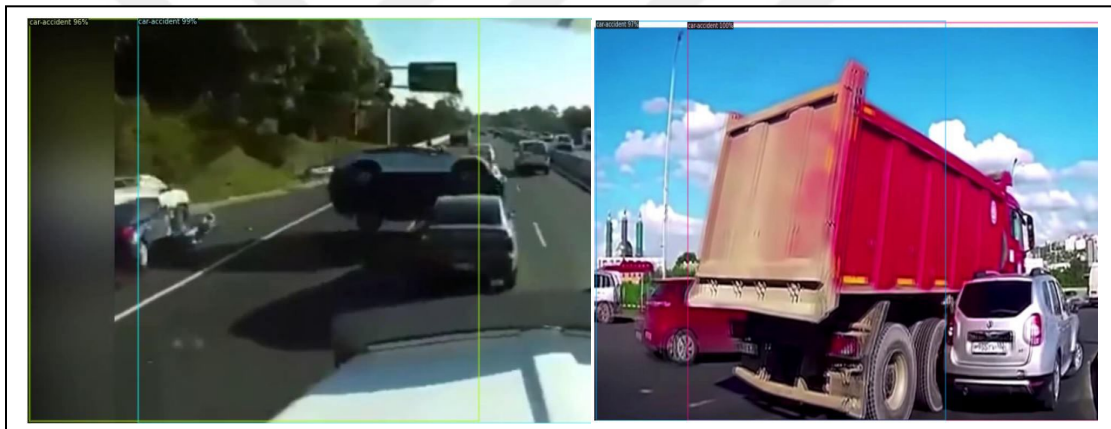


Figure 4.8 Accidents Detect By Faster R-CNN Itself

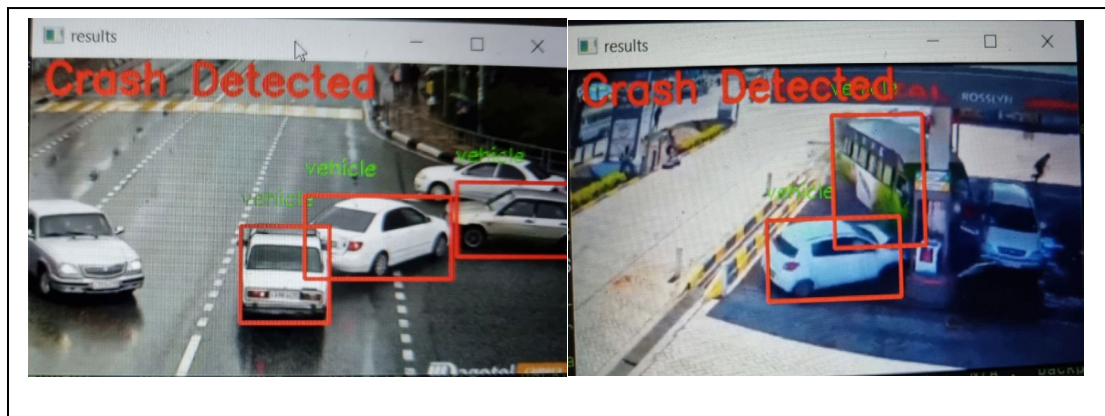


Figure 4.9 Accidents Detect By Intersection Between Bounding Boxes

We use equation (4.4) to calculate the accuracy of both models, Faster R-CNN with ResNet50-FPN achieves an accuracy of 81% while Faster R-CNN with MobileNet v3 large achieves an accuracy of 78%

in the detection step. The following table compares the accuracy of the three models that we train on our dataset (CAD-DATASET).

Table 4.3: Comparison between YOLOv5 and Faster R-CNN on CAD-DATASET

Model	Backbone	Accuracy
YOLOv5	Darknet-53	76.2%
Faster R-CNN	ResNet50-FPN	81.5%
Faster R-CNN	MobileNet v3 Large	78.0%

From the tables (4.3), we can see that Faster R-CNN with both backbone types outperforms YOLOv5.

4.2 CAR PLATE NUMBER RECOGNITION RESULTS AND DISCUSSION

We use two different ways in this task which are traditional EasyOCR and the second way is YOLOv7 with Pytessearct engine. We implement python and computer vision library in this task for both ways, but we use the Tesseract engine in a second way. Of course, in the two ways, we need first to pre-process images before the main function (OCR or Tesseract) works to extract characters as we mention in the previous chapter. The following figures show how EasyOCR extracts the characters from the images or frames of videos. Most EasyOCR methods are similar in some steps but in general most of the methods firstly .

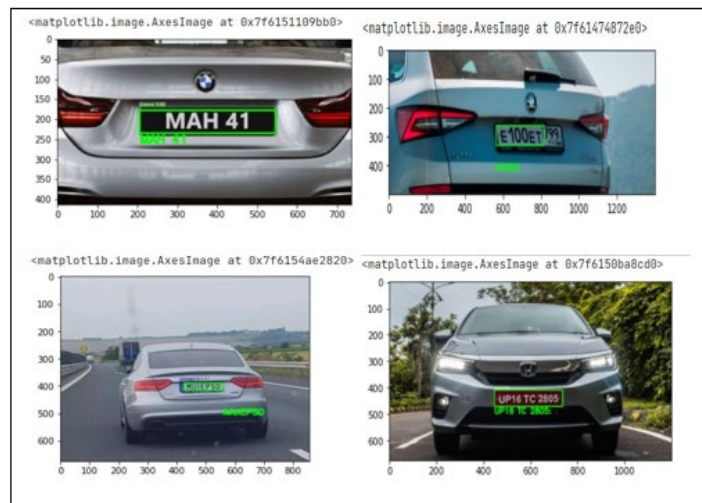


Figure 4.10 Plate Number Recognition by EasyOCR

Include the conversion of colored image to grayscale image and then remove noise from the image by using for example Gaussian blur or Bilateral Filter or any other filter to enhance the image which facilitates the work of the following step. And all of the methods used mainly EasyOCR to extract characters. Our model consists of six steps, achieving an accuracy of 96 %.

Whereas the second model includes the use of YOLOv7 to detect the plate number and then isolate the region that contains the detected plate number and finally the turn of the Tesseract engine comes to extract the characters and print them. The following figures show the evaluation of YOLOv7 performance to detect only the plate number.

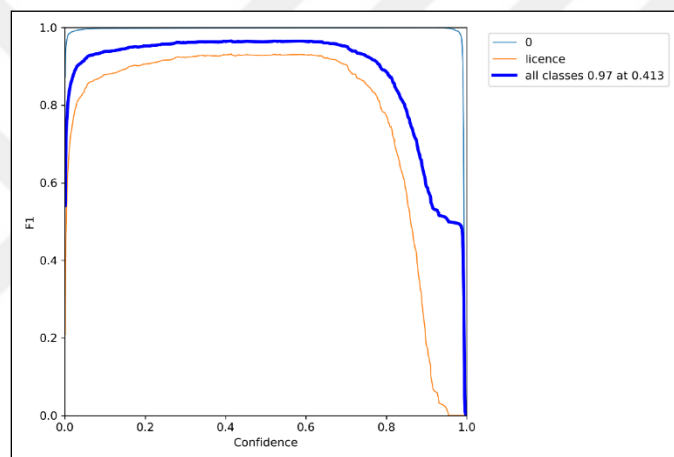


Figure 4.11 F1 Score for YOLOv7

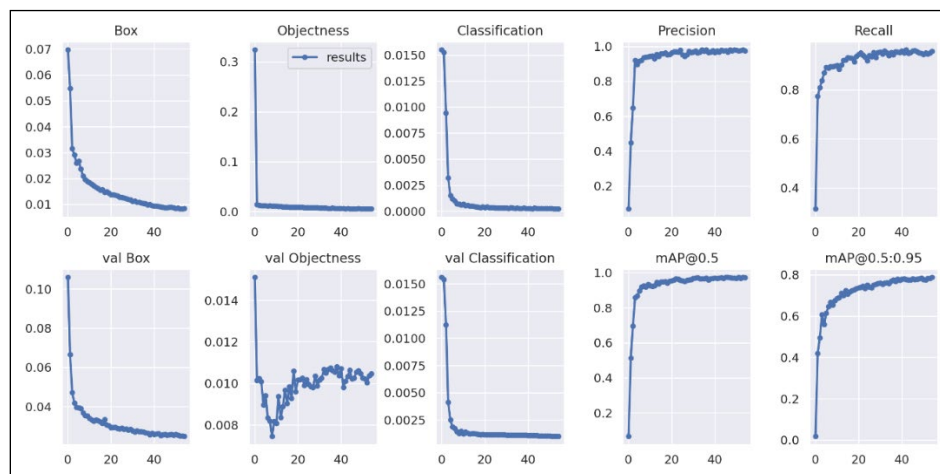


Figure 4.12 Metrics for YOLOv7

From the previous figure (4.12) we can conclude that our model mAP value increased sharply corresponding to the increase in the number of epochs starting from zero until it reaches the maximum value at the 40th epoch, which is equal to 0.99, this is similar to recall value which is reaching the maximum value at the 40th epoch too which is equal to 0.97. As we can see our model during small number can easily detect the plate number with high mAP, the reason for that is the rectangular shape of all plate number that YOLOv7 easily detect it. This makes our model achieve a high F1 score equal to 0.97 or 97% in the training step. The following figures show the detected plate number



Figure 4.13 Plate Number Detected By YOLOv7

The detected plate number will facilitate the operation of the following step. The following figure shows the result of the second step which includes the Isolation of the region of the plate number.



Figure 4.14 samples of isolated plate numbers by the second step

It's worth noting this step depends on the created bounding box by the previous step to focus and isolate the required region. We need to extract and isolate the region that contains the plate number since the Tesseract cannot extract the character from the whole image so we need to isolate the region that contains the plate number. After that, the Tesseract engine starts to extract characters from the image and print it by using a sequence of processes before extracting the character such as finding contours and converting the image to inversed binary image to make the edge of the character sharper to facilitate the grabbing required character from the background the following figures show the operation of Tesseract.

```
C:\Users\alimo\PycharmProjects\pythonProject1\venv\  
--- Start recognize text from image ---  
VAVACARS  
  
----- Done -----
```

```
C:\Users\alimo\PycharmProjects\pythonProject1\venv\  
--- Start recognize text from image ---  
CEQ 511  
  
----- Done -----
```

```
--- Start recognize text from image ---  
licence 0.82,  
  
MAH 41 |
```

Figure 4.15 samples of isolated plate numbers by the second step.

Finally, the accuracy of this model is equal to 94.3%. Easy OCR only extracts the characters from the images and but the YOLOv7 model detects the plate number with characters extraction.

5. CONCLUSION

The problem of car accidents is a contemporary problem that is difficult to avoid and control, especially in developing countries, and thus causes large numbers of deaths and severe injuries annually, in addition to property destruction and material losses, as it ranks ninth in the list of the top 10 causes of death annually. Therefore, this problem attracted the attention of researchers in most fields. For that reason, we endeavor to design a model for detecting car accidents and also recognize the plate numbers for security and safety purposes. For the car accident detection task we choose the YOLOv5 object detection algorithm with a rule of intersection between bounding and the second model was Faster R-CNN with two types of Backbones. Then we compare these models to evaluate which one is better for this task. In terms of accuracy Faster R-CNN with both types of backbone outperforms YOLOv5 since Faster R-CNN with ResNet50 backbone achieves accuracy (81.5%) more than Faster R-CNN with MobileNet v3 backbone that achieves accuracy (78%) which in turn more than YOLOv5 that achieves accuracy (76.2). But in terms of speed or runtime YOLOv5 is faster and outperforms Faster R-CNN. although that YOLOv5 in some practical experiments losses the object (cars) in some frames but Faster R-CNN shows higher false alarms in a detect accident than YOLOv5. In terms of resource consumption and the number of parameters, YOLOv5 surpasses Faster R-CNN, since YOLOv5 performs both classification and detection in one stage which makes it faster and has fewer parameters. Car accidents consider a real-time task that often occurs in a fraction of a second, so it requires a fast model with reasonable accuracy and a false alarm rate, so through the experiments, we think that YOLOv5 is better to the aforementioned task since its speed in detection and creating bounding box beside the ease of training and short training time. Our model of YOLOv5 encounters many drawbacks such as occlusion in some frames. Also shows a slight error in the accident videos that occur in bad illumination (snow, night) but in contrast, shows a great performance to detect accidents in various angles and positions, for example, it can detect accidents in traffic cameras, chopper cameras, phone camera, house monitor camera. For car plate number recognition, we implement traditional Easy OCR and on the other side, we implement YOLOv7 with Pytesseacr engine to evaluate which one is suitable for the task. First of all, Easy OCR achieves a slightly higher accuracy rate (96%) than the YOLOv7 model which achieves (94.3%), but in terms of speed through experiments YOLOV7 model in most cases faster than Easy OCR. Also, YOLOv7 shows great

performance to detect and extract characters in various angles opposite to the second algorithm, which showed relative weakness in some images that contain a plate number in various angles. In addition, Easy OCR shows some errors in high-resolution RGB images whereas the YOLOv7 model doesn't show any errors. finally, for both tasks, the matter of deciding which algorithm is suitable for the task remains a matter of preference, prefer accuracy over speed or vice versa.



REFERENCES

- [1] S. Gollapudi, “Deep Learning for Computer Vision,” in Learn Computer Vision Using OpenCV, 2019. doi: 10.1007/978-1-4842-4261-2_3.
- [2] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning An MIT Press Book, vol. 29, no. 7553. 2016.
- [3] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” Electron. Mark., vol. 31, no. 3, 2021, doi: 10.1007/s12525-021-00475-2.
- [4] M. Awad and R. Khanna, Efficient learning machines: Theories, concepts, and applications for engineers and system designers. 2015. doi: 10.1007/978-1-4302-5990-9.
- [5] G. Montavon, W. Samek, and K. R. Müller, “Methods for interpreting and understanding deep neural networks,” Digital Signal Processing: A Review Journal, vol. 73. 2018. doi: 10.1016/j.dsp.2017.10.011.
- [6] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11. 2019. doi: 10.1109/TNNLS.2018.2876865.
- [7] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Advances in Neural Information Processing Systems, 2015, vol. 2015-January.
- [8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, vol. 2016-December. doi: 10.1109/CVPR.2016.91.
- [9] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017, vol. 2017-January. doi: 10.1109/CVPR.2017.690.
- [10] Jan Erik Solem, “Programming Computer Vision with Python,” Program. Comput. Vis. with Python, 2012.

- [11] J. G. Choi, C. W. Kong, G. Kim, and S. Lim, "Car crash detection using ensemble deep learning and multimodal data from dashboard cameras," *Expert Syst. Appl.*, vol. 183, 2021, doi: 10.1016/j.eswa.2021.115400.
- [12] W. H. Organization, "WHO global status report on road safety 2018," *World Heal. Organ.*, no. 1, 2018.
- [13] S. Robles-Serrano, G. Sanchez-Torres, and J. Branch-Bedoya, "Automatic detection of traffic accidents from video using deep learning techniques," *Computers*, vol. 10, no. 11, 2021, doi: 10.3390/computers10110148.
- [14] M. S. Pillai, G. Chaudhary, M. Khari, and R. G. Crespo, "Real-time image enhancement for an automatic automobile accident detection through CCTV using deep learning," *Soft Comput.*, vol. 25, no. 18, 2021, doi: 10.1007/s00500-021-05576-w.
- [15] D. Tian, C. Zhang, X. Duan, and X. Wang, "An Automatic Car Accident Detection Method Based on Cooperative Vehicle Infrastructure Systems," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2939532.
- [16] S. Ghosh, S. J. Sunny, and R. Roney, "Accident Detection Using Convolutional Neural Networks," 2019. doi: 10.1109/IconDSC.2019.8816881.
- [17] W. J. Chang, L. B. Chen, and K. Y. Su, "DeepCrash: A deep learning-based internet of vehicles system for head-on and single-vehicle accident detection with emergency notification," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2946468.
- [18] H. Hozhabr Pour et al., "A Machine Learning Framework for Automated Accident Detection Based on Multimodal Sensors in Cars," *Sensors*, vol. 22, no. 10, p. 3634, May 2022, doi: 10.3390/s22103634.
- [19] M. Samantaray, A. K. Biswal, D. Singh, D. Samanta, M. Karuppiah, and N. P. Joseph, "Optical Character Recognition (OCR) based Vehicle's License Plate Recognition System Using Python and OpenCV," 2021. doi: 10.1109/ICECA52323.2021.9676015.

- [20] N. M. Dipu, S. A. Shohan, and K. M. A. Salam, "Bangla Optical Character Recognition (OCR) Using Deep Learning Based Image Classification Algorithms," 2021. doi: 10.1109/ICCIT54785.2021.9689864.
- [21] H. Rajput, T. Som, and S. Kar, "An automated vehicle license plate recognition system," *Computer (Long. Beach. Calif.)*, vol. 48, no. 8, 2015, doi: 10.1109/MC.2015.244.
- [22] C. Gou, K. Wang, Y. Yao, and Z. Li, "Vehicle License Plate Recognition Based on Extremal Regions and Restricted Boltzmann Machines," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, 2016, doi: 10.1109/TITS.2015.2496545.
- [23] B. A. Hussain and M. S. Hathal, "Developing Arabic license plate recognition system using artificial neural network and canny edge detection," *Baghdad Sci. J.*, vol. 17, no. 3, 2020, doi: 10.21123/bsj.2020.17.3.0909.
- [24] C. Henry, S. Y. Ahn, and S. W. Lee, "Multinational License Plate Recognition Using Generalized Character Sequence Detection," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.2974973.
- [25] Y. Zou et al., "A Robust License Plate Recognition Model Based on Bi-LSTM," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3040238.
- [26] J. Jordan, "An overview of object detection: one-stage methods.," Jeremy Jordan, 2018.
- [27] L. Du, R. Zhang, and X. Wang, "Overview of two-stage object detection algorithms," in *Journal of Physics: Conference Series*, 2020, vol. 1544, no. 1. doi: 10.1088/1742-6596/1544/1/012033.
- [28] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014. doi: 10.1109/CVPR.2014.81.
- [29] R. Girshick, "【Fast R-CNN】 Fast R-CNN," *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, 2015.

- [30] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in Proceedings of the IEEE International Conference on Computer Vision, 2017, vol. 2017-October. doi: 10.1109/ICCV.2017.322.
- [31] S. K. Pradhan, J. Thirupathi, N. V. K. Rao, J. Aluru, K. Suraparaju, and P. R. T, "MANUAL CHARACTER RECOGNITION WITH OCR," Turkish J. Physiother. Rehabil., vol. 32, no. 3, 2021, doi: 10.13140.
- [32] K. Smelyakov, A. Chupryna, D. Yeremenko, A. Sakhon, and V. Polezhai, "Braille Character Recognition Based on Neural Networks," 2018. doi: 10.1109/DSMP.2018.8478615.
- [33] C. Patel, A. Patel, and D. Patel, "Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study," Int. J. Comput. Appl., vol. 55, no. 10, 2012, doi: 10.5120/8794-2784.
- [34] J. Redmon and A. Farhadi, "YOLO v.3," Tech Rep., 2018.
- [35] C. Szegedy et al., "GoogLeNet Going Deeper with Convolutions," arXiv Prepr. arXiv1409.4842, 2014.
- [36] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4," CVPR Work. Futur. Datasets Vis., 2020.
- [37] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," 2020. doi: 10.1109/CVPR42600.2020.01079.
- [38] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017, vol. 2017-January. doi: 10.1109/CVPR.2017.106.
- [39] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," 2018. doi: 10.1109/CVPR.2018.00913.
- [40] G. Ghiasi, T. Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in Proceedings of the IEEE Computer Society Conference

- on Computer Vision and Pattern Recognition, 2019, vol. 2019-June. doi: 10.1109/CVPR.2019.00720.
- [41] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, “GhostNet: More features from cheap operations,” 2020. doi: 10.1109/CVPR42600.2020.00165.
- [42] U. Nepal and H. Eslamiat, “Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs,” *Sensors*, vol. 22, no. 2, 2022, doi: 10.3390/s22020464.
- [43] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, 2015, doi: 10.1109/TPAMI.2015.2389824.
- [44] A. Howard et al., “Searching for mobileNetV3,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, vol. 2019-October. doi: 10.1109/ICCV.2019.00140.
- [45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-December. doi: 10.1109/CVPR.2016.90.
- [46] C. Wang and C. Zhong, “Adaptive Feature Pyramid Networks for Object Detection,” *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3100369.
- [47] D. Han, T. Zhang, and J. Zhang, “Research and implementation of an improved canny edge detection algorithm,” *Key Eng. Mater.*, vol. 572, no. 1, 2014, doi: 10.4028/www.scientific.net/KEM.572.566.
- [48] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” *arXiv Prepr. arXiv2207.02696*, 2022.
- [49] C. Li et al., “YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications,” *arXiv Prepr. arXiv2209.02976*, 2022.
- [50] A. L. S. Saabith, T. Vinothraj, and M. Fareez, “POPULAR PYTHON LIBRARIES AND THEIR APPLICATION DOMAINS,” *Int. J. Adv. Eng. Res. Dev.*, vol. 7, no. 11, 2020.

- [51] I. Santos, L. Castro, N. Rodriguez-Fernandez, Á. Torrente-Patiño, and A. Carballal, “Artificial Neural Networks and Deep Learning in the Visual Arts: a review,” *Neural Computing and Applications*, vol. 33, no. 1. 2021. doi: 10.1007/s00521-020-05565-4.
- [52] S. Yu, K. Wickstrom, R. Jenssen, and J. Principe, “Understanding Convolutional Neural Networks with Information Theory: An Initial Exploration,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 32, no. 1, 2021, doi: 10.1109/TNNLS.2020.2968509.
- [53] A. Distanto and C. Distanto, *Handbook of Image Processing and Computer Vision*. 2020. doi: 10.1007/978-3-030-42378-0.
- [54] D. Mishra, B. Naik, R. M. Sahoo, and J. Nayak, “Deep Recurrent Neural Network (Deep-RNN) for Classification of Nonlinear Data,” in *Advances in Intelligent Systems and Computing*, 2020, vol. 1120. doi: 10.1007/978-981-15-2449-3_17.
- [55] M. S. Nixon and A. S. Aguado, *Feature extraction and image processing for computer vision*. 2019. doi: 10.1016/C2017-0-02153-5.
- [56] H. Pal and B. Narwal, “A Novel Approach to Optimize Deep Neural Network Architectures,” in *Advances in Intelligent Systems and Computing*, 2021, vol. 1199. doi: 10.1007/978-981-15-6353-9_26.
- [57] V. Wiley and T. Lucas, “Computer Vision and Image Processing: A Paper Review,” *Int. J. Artif. Intell. Res.*, vol. 2, no. 1, 2018, doi: 10.29099/ijair.v2i1.42.