Number Plate Recognition Using YOLOv5

Report submitted to the SASTRA Deemed to be University
as the requirements for the course

CSE300 – MINI PROJECT

Submitted by

Kakumani Lakshmi Priya (Reg. No.: 225003074)
Narreddy Pranathi Maha Shiva (Reg. No.: 225003105)
Amirtha C (Reg. No.: 225003201)

May 2024

Department of Computer Science and Engineering
SRINIVASA RAMANUJAN CENTRE
Kumbakonam, Tamil Nadu, INDIA - 612 001

SASTRA

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

Department of Computer Science and Engineering

SRINIVASA RAMANUJAN CENTRE

Kumbakonam, Tamil Nadu, INDIA - 612 001

May 2024

Bonafide Certificate

This is to certify that the report titled **"Number Plate Recognition Using YOLOv5"** submitted as *in partial fulfilment of the requirements for the award of the degree of B.Tech.*, Computer Science and Engineering to the SASTRA Deemed to be University, is a bonafide record of the work done by **Ms. Kakumani Lakshmi Priya (225003074), Ms. Narreddy Pranathi Maha Shiva (225003105), Ms. Amirtha C (225003201)** during the final semester of the academic year 2023-2024, in the Srinivasa Ramanujan Centre, under my supervision. This project report has not formed the basis for the award of any degree, diploma, associateship, fellowship or other similar title to any candidate of any University.

Signature of Project Supervisor :

Name with Affiliation : **Abiramasundari S AP/CSE**

Date : 6/5/24

Mini Project *Viva voce* held on 6/5/2024

Examiner 1 6/5/24

Examiner 2 06/05/24

Acknowledgements

We pay our sincere pranam to **God ALMIGHTY** for his grace and infinite mercy and for showing on us his choicest blessings

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam and Dr.S. Swaminathan,** Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli,** Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. Ramaswamy,** The Dean and **Dr. A. Alli Rani**, Associate Dean, Srinivasa Ramanujan Centre for their constant support and suggestions when required without any reservations.

We express our sincere gratitude to **Dr. V. Kalaichelvi,** Associate Professor, Department of Computer Science and Engineering for her unconditional ss and encouragement for completing the project.

Our guide **Mrs. S. Abiramasundari,** Assistant Professor, Department of Computer Science & Engineering, Srinivasa Ramanujan Centre was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped us in making progress throughout our project work.

We also thank the Project Coordinator **Dr. S. Priyanga,** Assistant Professor, Department of Computer Science & Engineering, Srinivasa Ramanujan Centre for her ever-encouraging spirit and meticulous guidance for the completion of the project.

We also thank the panel members for their valuable comments and insights which made this project better. We would like to extend our gratitude to all the teaching and non-teaching faculties of the Srinivasa Ramanujan Centre who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from our family and friends resulting in the successful completion of this project. We thank you all for providing us with an opportunity to showcase our skills through the project.

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolution Neural Networks
LPR	License Plate Recognition
OCR	Optical Character Recognition
DCP	Dark Channel Prior
CSP	Cross Stage Partial Network
PAN	Path Aggregation Network

ABSTRACT

In modern security and surveillance systems accurate vehicle identification is crucial in the traffic. Different whether conditions like fog and haze cause major difficulties that make number plates to be blurry, therefore accurate Number Plate Recognition is required. By utilizing deep learning methods such as Optical Character Recognition (OCR) for character recognition, Dark Channel Prior (DCP) for haze removal, and YOLOv5 for number plate boundary detection, the system is able to accurately and consistently recognize number plates in real-time images and video streams. The suggested approach provides a reliable approach in hazy environments by precisely detecting number plates, extracting characters, and efficiently eliminating haze.

Key Words:

Number Plate Recognition, YOLOv5, Optical Character Recognition, Dark Channel Prior, Object Detection

Specific Contribution:

Kakumani Lakshmi Priya - Implementation of Dark Channel Prior for haze removal

Narreddy Pranathi Maha Shiva - Training and Testing of YOLOv5 model

Amirtha C - Implementation of OCR for Character Extraction

Technical Limitations & Ethical Challenges faced:

- Training YOLOv5 models is time consuming process
- In OCR, some characters are wrongly recognized

Table of Contents

CHAPTER 1

SUMMARY OF THE BASE PAPER

Title : **Liscense Plate Recognition For Vehicle Identification**

Journal Name : Science Direct

Authors : Alvaro Ramajo-Ballester

Year : 2024

1.1 SUMMARY:

- The project primarily focuses on License Plate Recognition in images using deep learning techniques
- It constructs a specialized architecture that integrates advanced algorithms, including YOLOv5 for object detection and Optical Character Recognition (OCR) for character recognition
- Using the YOLOv5 model, which is known for its speed and accuracy, the system recognizes license plates in pictures under a variety of conditions
- Additionally, OCR algorithms are employed to precisely recognize and classify characters on the detected license plates, enhancing the system's ability to extract text information
- Convolutional Neural Networks (CNNs) serve as the backbone architecture for the YOLOv5 object detection model, facilitating efficient feature extraction from license plate images
- The successful implementation of these deep learning techniques enables accurate and reliable license plate recognition in images

1.2 INTRODUCTION

1.2.1 YOLOv5 MODEL:

The object detection model known as "You Only Look Once version 5," or YOLOv5, was first presented as an advancement of the YOLO (You Only Look Once) architecture. This is a synopsis of YOLOv5's introduction:

Improvements to Performance: Yolov5 aims to attain a higher level of accuracy and speed by building on the qualities of its predecessors. It makes use of recent developments in model architectures and deep learning methods to improve object detection results. **Variations of the Model:** The Yolov5 is available in four sizes: small, medium, big, and extra-large. Each model allows customers to select the one that best meets their unique needs by providing a trade-off between speed and precision.

YOLOv5 has 3 parts:

1.Backbone Network: This network is in charge of extracting features from the input images. Usually, it is made up of hierarchically arranged convolutional layers. The backbone network of YOLOv5, which is frequently built on CSPDarknet53, gathers information at several levels of abstraction from the input image by extracting hierarchical characteristics.

- Convolutional layers make up the structure of a backbone network. Convolutional Neural Networks (CNNs) are normally built up of multiple layers. CNNs are especially good at identifying patterns and spatial hierarchies in images
- Layering: The backbone network's layers are organized in a hierarchical fashion. Accordingly, upper levels collect more abstract and sophisticated features, like shapes and objects, while lower layers catch low-level features, like edges and textures

CSPDarknet53:

- CSP stands for Cross Stage Partial Network, which introduces a novel technique to improve the efficiency and performance of the network
- CSPDarknet53 is essentially a feature extractor that combines the effectiveness of Darknet with the gradient flow augmentation of CSP, hence improving YOLOv5's object detection capabilities

2.Neck:

- The intermediary set of layers between the detecting head and the backbone makes up the neck of the YOLOv5 architecture. Its function is to combine and enhance characteristics from various scales that the backbone has acquired

- The YOLOv5 network incorporates a PANet (Path Aggregation Network) as its neck architecture. PANet helps to aggregate multi-scale features from different stages of the backbone network, enabling the model to better handle objects of various sizes

PANet:

- An essential part of YOLOv5 is PANet, which is smoothly incorporated into the neck between the detecting head and the backbone. By combining multi-scale characteristics from several backbone stages, it improves the model's capacity to identify objects with diverse sizes. Through the use of adaptive feature aggregation and the construction of a feature pyramid, PANet guarantees better object representation at various scales

3.Detection Head:

Using the features that the neck and backbone have extracted and refined, the detection head is in charge of generating predictions. Detection head carries out the following steps:

- **Bounding Box Prediction:** The detection head forecasts the bounding boxes surrounding the objects it has spotted. The position (x, y) and size (width, height) of each bounding box with respect to the image are indicated by its coordinates

- **Objectness Scores:** For every bounding box, YOLOv5 forecasts the objectness scores. These scores represent the probability that an object of interest is contained within the projected bounding box. A high likelihood of object presence is indicated by high objectness scores

- **Anchor Boxes:** Anchor boxes are definite bounding box forms with predetermined dimensions. The size and location of bounding boxes for items that are identified are predicted by the detection head using anchor boxes as references

- **Loss Function:** During training, the model optimizes its parameters by comparing the predicted bounding boxes, objectness scores, and class probabilities with the ground truth annotations. This is done using a combination of localization loss, confidence loss, and classification loss

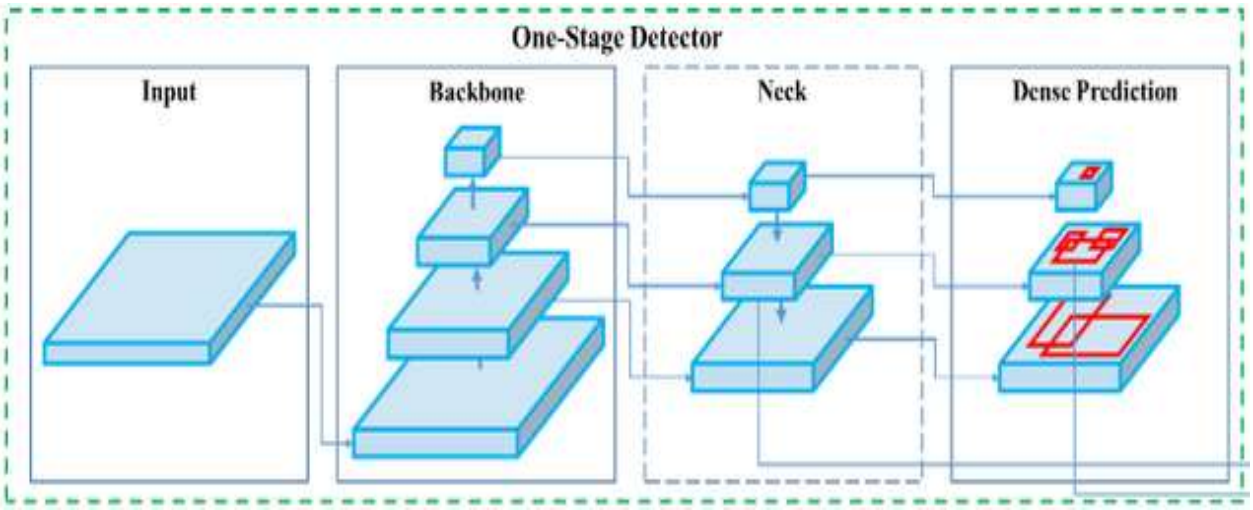


Fig 1.1 YOLOv5 Architecture

1.2.2 OPTICAL CHARACTER RECOGNITION(OCR):

- **Image Detection:** In an OCR system, this is the initial stage. To do this, an object detection model is used to locate and identify required objects inside an image frame. The supplied model locates number plate containing regions in the image by scanning the image

- **Number Plate Recognition:** Bounding boxes are plotted around number plate to indicate their bounds after they have been identified. Bounding boxes are useful for visualising and separating regions of interest for additional processing

- **Text Recognition**: The system can identify and recognize characters from the identified number plates using algorithm like EasyOCR

- **Text Filtering:** A text filtering phase is frequently used to guarantee accuracy. In this step, the recognised text is analysed and any errors or incorrectly detected parts are filtered out. The technique enhances the overall quality of the retrieved text data by eliminating false positives

1.2 PROBLEM DEFINITION

- Overcoming obstacles like great distances and diverse climatic conditions is essential to the efficient operation of Number Plate Recognition technology, particularly in unfavourable circumstances like fog

- Plate recognition becomes more challenging when visibility is limited, highlighting the need for creative solutions. In this conditions by using deep learning techniques number plates are recognized accurately

- We can increase the accuracy and adaptability of Number Plate Recognition by utilising deep learning, optimising sensors, and improving algorithms. This increases its usefulness in a variety of settings and guarantees its effectiveness in traffic management, security, and surveillance

1.3 PROPOSED ARCHITECTURE

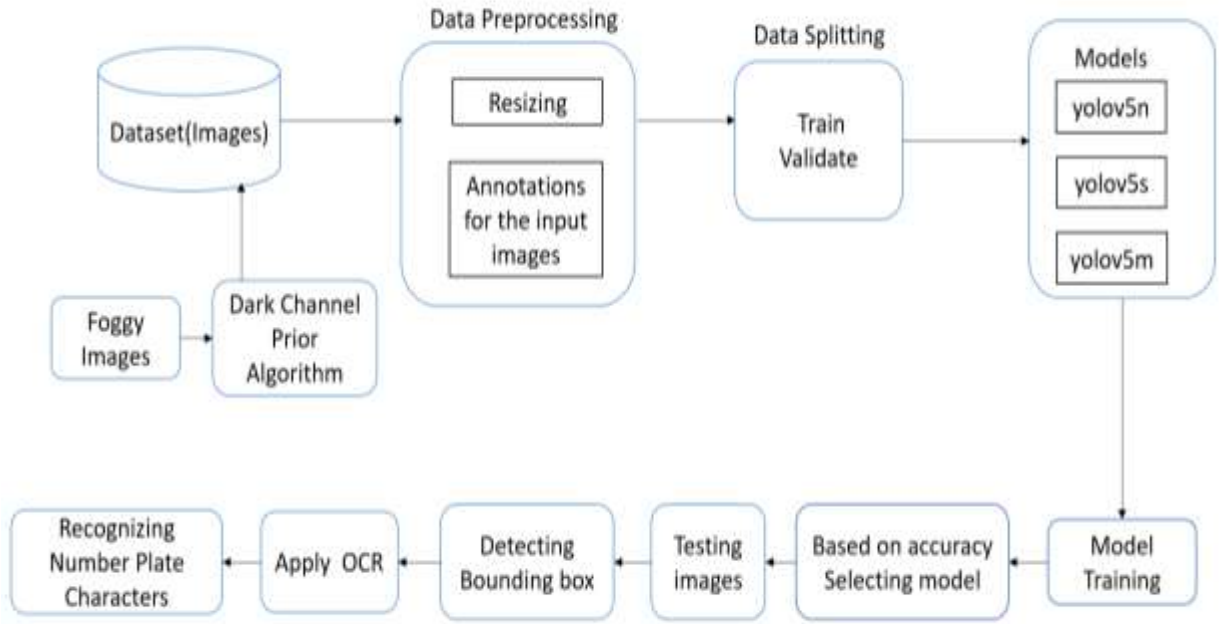


Fig 1.2 Architecture Diagram for Proposed System

1.4 SOLUTION APPROACH

Data Preparation: To maintain consistency and maximize processing effectiveness, resize the images to a consistent size. Annotate the pictures with LabelImg by putting class labels and bounding boxes around the number plates. Store these annotations in the YOLO format, which is essentially a text file with a line for each annotated item that includes the bounding box coordinates and class label.

Data Selection: The models are trained on the 80% training set, and their ability to generalize to new data is evaluated on the 20% validation set.

Model Training and Selection: Test out various YOLOv5 model variations, including YOLOv5n, YOLOv5s, and YOLOv5m, to see which one is the most accurate at detecting number plates. When choosing a model for deployment, look for the one that offers the best balance between accuracy.

Model Deployment: Model creates a ".pt" file with the trained parameters. This file can be utilized for inference, which is the process of taking input images and producing bounding boxes and class labels for number plates that are identified. Deploy the model in the desired environment for real-time or batch processing of images.

Text Recognition (OCR): Utilize Optical Character Recognition (OCR) techniques to extract text, including number plate characters, from the identified regions of interest in the images once number plates have been identified. To do OCR on these areas, use libraries such as EasyOCR. These libraries can recognize characters.

Dehazing: The Dark Channel Prior (DCP) algorithm dehazes foggy images to improve visibility. After processing, these photos are combined with the original dataset to improve model adaptability and enhance environmental diversity.

DARK CHANNEL PRIOR (DCP):

The Dark Channel Prior (DCP) algorithm is a widely used method for single image dehazing. It is based on the observation that the dark channel of a haze-free image, which is the minimum intensity value across the RGB channels, contains important information about the scene. The Dark Channel Prior (DCP) algorithm for image dehazing, which involves the following key steps:

- **Dark Channel Calculation:** The dark channel algorithm first computes the minimum intensity value among the three color channels for each pixel, and then applies morphological erosion to find the minimum value within each local patch

- **Atmospheric Light Estimation:** The atmospheric light is estimated using the dark channel by selecting the minimum intensity values from a specified percentage of the image pixels

- **Transmission Map Estimation**: The transmission map is estimated based on the atmospheric light and the dark channel. The transmission map is initialized as a matrix of ones and adjusted using the dark channel to refine the haze removal process

- **Dehazing:** The dehazed image is generated by utilizing the input image, transmission map, and atmospheric light. The dehazing process involves correcting the image using the estimated transmission map and atmospheric light to enhance visibility and reduce haze effects

Dataset: The datasets used for this project is car plate detection from Kaggle contains 433 car images and some images from the real number plates which are collected randomly. For dehazing foggy images has been chosen from different sources.

1.6 TECHNOLOGIES USED:

To implement this project, several technologies and libraries were used, each serving a specific purpose:

NUMPY: It facilitates the effective handling, resizing, normalization of pixel values, and manipulation of bounding box coordinates of input images. NumPy also helps with non-maximum suppression. NumPy functions facilitates the efficient handling and manipulation of image numerical data.

PYTORCH: PyTorch is probably used for loading pre-trained weights, defining the architecture of the YOLO model, and doing inference on videos or images that are brought in. Neural network models can be created, trained, and implemented using PyTorch, the main deep learning package.

EASYOCR: Importing the EasyOCR Python Optical Character Recognition (OCR) library serves the primary objective of improving the text extraction process from scanned documents or photos.

CV2: For computer vision and image processing applications, OpenCV is a common library. For tasks including picture reading and writing, image modification, feature and object detection, and more, it offers a variety of functions and algorithms.

The machine used for this project is Jupyter and Git Command, Kaggle and laptop with following specifications.

- Processor: AMD Ryzen 3 3250U @ 2.60GHz
- Random Access Memory (RAM) : 8GB
- Operating System : Windows 11 Home Single Language

CHAPTER 2

MERITS AND DEMERITS OF BASE PAPER

2.1 LITERATURE SURVEY

- The research paper on "Vehicle Detection and Recognition Technology based on Artificial Intelligence" by Bo Yang. The research paper emphasizes the need of researching AI-based vehicle detection and recognition technology and underlines the expanding application of AI in traffic management. The developed technique may not, however, be sufficiently demonstrated in the paper's field testing or real-world application, which would limit its practical applicability
- The study "Vision-based vehicle detection and counting system using deep learning in highway scenes" introduces a novel method for precisely identifying and counting cars on highways through the use of deep learning. Issues such as different object sizes and high viewing angles are addressed. Its exclusive concentration on highway sceneries, however, may limit its application to other traffic situations, such as city streets or country roads
- The goal of the paper "Research on Vehicle Detection Algorithm Based on Improved YOLOv4" is to increase vehicle detection accuracy by utilizing an enhanced YOLOv4 algorithm. It demonstrates notable gains in accuracy over earlier techniques. The study does, however, recognize that there are issues with computational complexity and scalability that could affect how well it works in large-scale applications
- The base paper “Vehicle Identification and Location Detection at Closed-Circuit Television to Find Vehicles Using OCR and OpenCV” improves accuracy by automatically identifying vehicles and detecting their locations from CCTV images by creatively combining OCR and OpenCV technologies. The correct identification of cars in challenging circumstances, such as dim lighting or partially covered license plates, may provide difficulties, though, and the integration and upkeep of these technologies may present difficulties and support issues for real-world surveillance systems
- High detection accuracy and low false alarm rates are provided by the foundational study, "Research on Vehicle Identification Method based on Computer Vision" which improves vehicle safety. But its application in different lighting circumstances is limited because it mainly concentrates on vehicle identification in specific lighting conditions

2.2 MERITS

- The system has been trained and tested on both public and proposed datasets, demonstrating remarkable performance in license plate detection, character recognition
- Utilizes advanced feature extraction methods from deep learning architectures like YOLOv5 to extract bounding box and their coordinates
- The thorough evaluation of the system's performance showcases its effectiveness in real-world scenarios and highlights its potential for applications in video surveillance and traffic scene analysis
- Accurate and efficient License Plate Recognition is achieved by utilizing modern deep learning techniques, including object detection models (YOLOv5) and OCR algorithms. Convolutional Neural Networks, or CNNs, are integrated into the system to improve feature extraction and better License Plate Recognition performance

2.3 DEMERITS

- Effects of Image Size and Batch Size: Larger batch sizes could result in flatter minima in the optimisation landscape, which would reduce the model's ability to generalise well. This could change the learning dynamics of the model and cause it to converge to a less generalizable sub-optimal solution.
- Size of Input Image: Although finer features can be recovered from higher resolution input images, not many number plates have dimensions larger than 640 pixels.. As a result, enlarging the image may not always result in a more detailed view of the image
- Training Speed and Effectiveness: The effect of batch size on training speed is emphasised, where a smaller batch size might accelerate convergence by increasing the frequency of gradient updates in the model. This may result in the utilisation of computing resources more effectively. Nonetheless, the model's overall performance must be taken into account when evaluating the impact of batch size on training speed
- Model Dimensions: Because they have a propensity to overfit smaller datasets, larger models might not necessarily perform better. Because they can prevent overfitting, smaller models like YOLOv5 can perform better than larger models in specific situations. It should be noted, though, that the smallest model, YOLOv5-n, exhibits substantial underfitting, suggesting that model size is not always a direct correlation of performance

CHAPTER 3

SOURCE CODE

```
In [ ]: import torch

In [4]: import numpy

In [ ]: import cv2

In [ ]: import os
```

CLONING INTO YOLOv5 REPOSITORY

```
In [5]: #!git clone https://github.com/ultralytics/yolov5
```

Preprocessing:

PREPROCESSING ¶

```
In [ ]: def resize_images(input_folder, output_folder, target_size):
            os.makedirs(output_folder, exist_ok=True)
            for filename in os.listdir(input_folder):
                img_path = os.path.join(input_folder, filename)
                img = cv2.imread(img_path)
                resized_img = cv2.resize(img, target_size)
                output_path = os.path.join(output_folder, filename)
                cv2.imwrite(output_path, resized_img)

In [ ]: resize_images('C:/Users/welcome/Desktop/all/number_plate_images_ocr', 'C:/Users/welcome/Desktop/all/number_plate_images_ocr', (41

In [ ]: #SPLITTING DATA INTO TRANING AND VALIDATION

In [ ]: training_imgpath="C:/Users/welcome/Desktop/yolov5-master/yolov5/dataset/images/train"
        validation_imgpath="C:/Users/welcome/Desktop/yolov5-master/yolov5/dataset/images/val"
```

DCP:

```
In [1]: import cv2
        import numpy as np

In [2]: def dark_channel(image, patch_size):
            b, g, r = cv2.split(image)
            min_channel = cv2.min(cv2.min(r, g), b)
            kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (patch_size, patch_size))
            dark = cv2.erode(min_channel, kernel)
            return dark

        def estimate_atmospheric_light(image, dark_channel, percentage):
            pixels = dark_channel.size
            flat_dark = dark_channel.flatten()
            sorted_dark = np.sort(flat_dark)
            threshold_index = int(pixels * percentage / 100)
            atmospheric_light = np.mean(sorted_dark[-threshold_index:])
            return atmospheric_light
        def estimate_transmission(image, atmospheric_light, omega):
            transmission = np.ones_like(image, dtype=np.float32)
            transmission[:, :, 0] = atmospheric_light / image[:, :, 0]
            transmission[:, :, 1] = atmospheric_light / image[:, :, 1]
            transmission[:, :, 2] = atmospheric_light / image[:, :, 2]
            return 1 - omega * dark_channel(transmission, 15)
```

```
def dehaze(image, transmission, atmospheric_light, t0):
    dehazed = np.zeros_like(image, dtype=np.float32)
    for i in range(3):
        dehazed[:, :, i] = (image[:, :, i] - atmospheric_light) / np.maximum(transmission, t0) + atmospheric_light
    return np.clip(dehazed, 0, 255).astype(np.uint8)

def dcp_dehazing(image, patch_size=15, percentage=0.1, omega=0.9, t0=0.1):
    dark = dark_channel(image, patch_size)
    atmospheric_light = estimate_atmospheric_light(image, dark, percentage)
    transmission = estimate_transmission(image, atmospheric_light, omega)
    return dehaze(image, transmission, atmospheric_light, t0)
```

In [13]:
```
# Take image path as input
image_path = r'C:\Users\welcome\Desktop\dark channel\dataset\74.png'

# Load input image
image = cv2.imread(image_path)
```

In [14]:
```
if image is None:
    print("Error: Unable to load the image from the provided path.")
else:
    # Perform DCP-based dehazing
    dehazed_image = dcp_dehazing(image)

    # Save the dehazed image
    output_path = r'C:\Users\welcome\Desktop\dark channel\output\dehazed_image.png'
    cv2.imwrite(output_path, dehazed_image)
```

OCR for images:

```
import torch
import cv2
import time

import re
import numpy as np
import easyocr

EASY_OCR = easyocr.Reader(['en']) ### initiating easyocr
OCR_TH = 0.2



### ------------------------------------ function to run detection ------------------------------------------------
def detectx (frame, model):
    frame = [frame]
    print(f"[INFO] Detecting. . . ")
    results = model(frame)
    labels, cordinates = results.xyxyn[0][:, -1], results.xyxyn[0][:, :-1]

    return labels, cordinates

### ------------------------------------ to plot the BBox and results ----------------------------------------------
def plot_boxes(results, frame,classes):

    labels, cord = results
    n = len(labels)
    x_shape, y_shape = frame.shape[1], frame.shape[0]
```

```
### ---------------------------------- to plot the BBox and results ----------------------------------------------------
def plot_boxes(results, frame,classes):

    labels, cord = results
    n = len(labels)
    x_shape, y_shape = frame.shape[1], frame.shape[0]

    print(f"[INFO] Total {n} detections. . . ")
    print(f"[INFO] Looping through all detections. . . ")



    for i in range(n):
        row = cord[i]
        if row[4] >= 0.55:
            print(f"[INFO] Extracting BBox coordinates. . . ")
            x1, y1, x2, y2 = int(row[0]*x_shape), int(row[1]*y_shape), int(row[2]*x_shape), int(row[3]*y_shape)
            text_d = classes[int(labels[i])]


            coords = [x1,y1,x2,y2]

            plate_num = recognize_plate_easyocr(img = frame, coords= coords, reader= EASY_OCR, region_threshold= OCR_TH)



            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2) ## BBox
            cv2.rectangle(frame, (x1, y1-20), (x2, y1), (0, 255,0), -1) ## for text label background
            cv2.putText(frame, f"{plate_num}", (x1, y1), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(255,255,255), 2)

    return frame
```

```
#### ---------------------------- function to recognize license plate ------------------------------


# function to recognize license plate numbers using Tesseract OCR
def recognize_plate_easyocr(img, coords,reader,region_threshold):

    xmin, ymin, xmax, ymax = coords

    nplate = img[int(ymin):int(ymax), int(xmin):int(xmax)]


    ocr_result = reader.readtext(nplate)



    text = filter_text(region=nplate, ocr_result=ocr_result, region_threshold= region_threshold)

    if len(text) ==1:
        text = text[0].upper()
    return text
```

```python
def filter_text(region, ocr_result, region_threshold):
    rectangle_size = region.shape[0]*region.shape[1]

    plate = []
    print(ocr_result)
    for result in ocr_result:
        length = np.sum(np.subtract(result[0][1], result[0][0]))
        height = np.sum(np.subtract(result[0][2], result[0][1]))

        if length*height / rectangle_size > region_threshold:
            plate.append(result[1])
    return plate




### ------------------------------------------- Main function ---------------------------------------------------

def main(img_path=None, vid_path=None,vid_out = None):

    print(f"[INFO] Loading model... ")


    model =  torch.hub.load('./yolov5', 'custom', source ='local', path='C:/Users/welcome/Desktop/yolov5-master/yolov5/runs/trair

    classes = model.names
```

```python
    ### ---------------- for detection on image ---------------------
    if img_path != None:
        print(f"[INFO] Working with image: {img_path}")
        img_out_name = f"./output/result_{img_path.split('/')[-1]}"

        frame = cv2.imread(img_path)
        frame = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)

        results = detectx(frame, model = model)

        frame = cv2.cvtColor(frame,cv2.COLOR_RGB2BGR)

        frame = plot_boxes(results, frame,classes = classes)


        cv2.namedWindow("img_only", cv2.WINDOW_NORMAL)

        while True:


            cv2.imshow("img_only", frame)

            if cv2.waitKey(5) & 0xFF == ord('q'):
                print(f"[INFO] Exiting. . . ")

                cv2.imwrite(f"{img_out_name}",frame)
                break
```

```
        print(f"[INFO] Clening up. . . ")
        ### releaseing the writer
        out.release()

        ## closing all windows
        cv2.destroyAllWindows()

### -------------------  calling the main function--------------------------------


main(img_path="C:/Users/welcome/Desktop/DATA_SET/25.png") ## for image
```

OCR for videos:

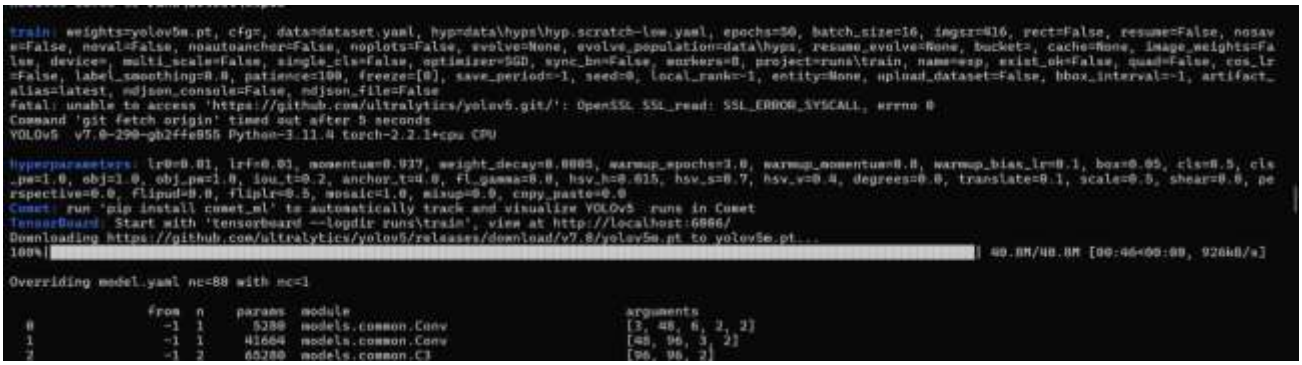
```
import cv2

def process_video(vid_path, vid_out=None, model=None, classes=None):
    if vid_path is not None:
        print(f"[INFO] Working with video: {vid_path}")

        cap = cv2.VideoCapture(vid_path)
        if vid_out:
            width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
            height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
            fps = int(cap.get(cv2.CAP_PROP_FPS))
            codec = cv2.VideoWriter_fourcc(*'mp4v')
            out = cv2.VideoWriter(vid_out, codec, fps, (width, height))

        frame_no = 1
        while True:
            ret, frame = cap.read()
            if ret:
                print(f"[INFO] Working with frame {frame_no} ")

                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                results = detectx(frame, model=model)
                frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

                frame = plot_boxes(results, frame, classes=classes)

                if vid_out:
                    out.write(frame)

                frame_no += 1
```

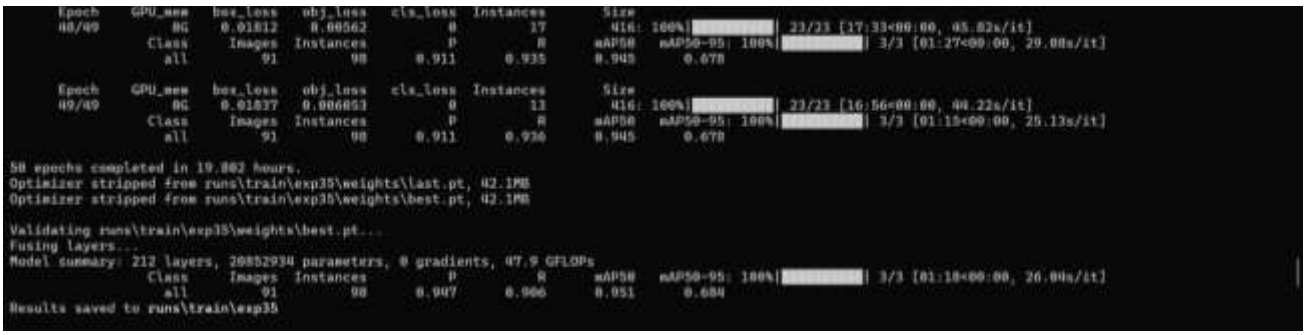
```
            else:
                break

        print(f"[INFO] Cleaning up...")
        if vid_out:
            out.release()

        cap.release()
        cv2.destroyAllWindows()

# Example usage
# Specify your video file paths and other parameters
video_path = "C:/Users/welcome/Downloads/5579747-hd_1920_1080_30fps.mp4"
output_path = "C:/Users/welcome/Downloads/output2.mp4"
model =  torch.hub.load('./yolov5', 'custom', source ='local', path='C:/Users/welcome/Desktop/yolov5-master/yolov5/runs/train/ex
classes = model.names
process_video(video_path, vid_out=output_path, model=model, classes=classes)
```

CHAPTER 4
SNAPSHOTS

Training Models:

YOLOv5m:





Accuracy:

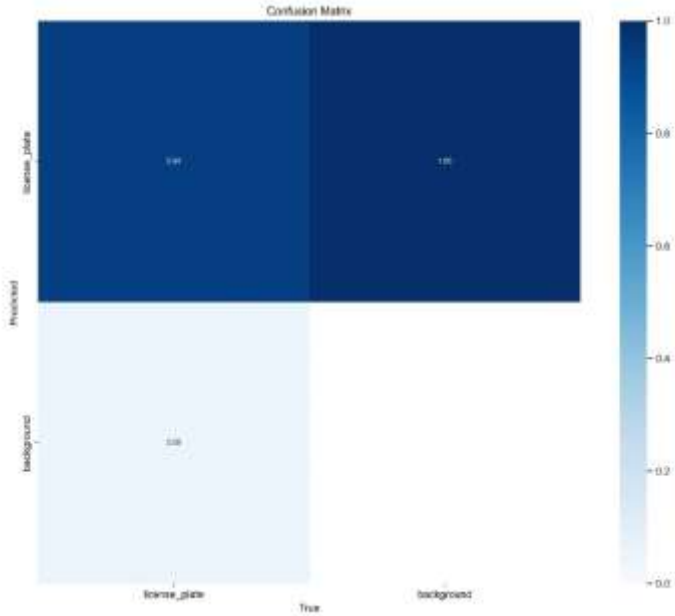


Fig 4.1 Confusion Matrix for YOLOv5m

YOLOv5s:

```
C:\Users\welcome\Desktop\yolov5-master\yolov5>python train.py --img 416 --batch 16 --epochs 50 --data dataset.yaml --weights yolov5s.pt
train: weights=yolov5s.pt, cfg=, data=dataset.yaml, hyp=data\hyps\hyp.scratch-low.yaml, epochs=50, batch_size=16, imgsz=416, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, noplots=False, evolve=None, evolve_population=data\hyps, resume_evolve=None, bucket='', cache=None, image_weights=False, device='', multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=runs\train, name=exp, exist_ok=False, quad=False, cos_lr=False, label_smoothing=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest, ndjson_console=False, ndjson_file=False
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 15 (delta 8), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (15/15), 5.90 KiB | 92.00 KiB/s, done.
From https://github.com/ultralytics/yolov5
 * [new branch]      glenn-jocher-patch-2 -> origin/glenn-jocher-patch-2
github: up to date with https://github.com/ultralytics/yolov5
YOLOv5  v7.0-298-gb24fe055 Python-3.11.4 torch-2.2.1+cpu CPU

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5  runs in Comet
TensorBoard: Start with 'tensorboard --logdir runs\train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=1

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
```

```
     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     48/49         0G    0.02072   0.006352          0         17        416: 100%|██████████| 23/23 [05:29<00:00, 14.33s/it]
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 3/3 [00:33<00:00, 11.10s/it]
                 all         91         98      0.968      0.837      0.922      0.653

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     49/49         0G    0.02104   0.006777          0         13        416: 100%|██████████| 23/23 [05:26<00:00, 14.19s/it]
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 3/3 [00:33<00:00, 11.09s/it]
                 all         91         98      0.972      0.837       0.92      0.659

50 epochs completed in 5.924 hours.
Optimizer stripped from runs\train\exp36\weights\last.pt, 14.3MB
Optimizer stripped from runs\train\exp36\weights\best.pt, 14.3MB

Validating runs\train\exp36\weights\best.pt...
Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%|██████████| 3/3 [00:30<00:00, 10.33s/it]
                 all         91         98      0.972      0.837       0.92      0.658
Results saved to runs\train\exp36
```

Accuracy:

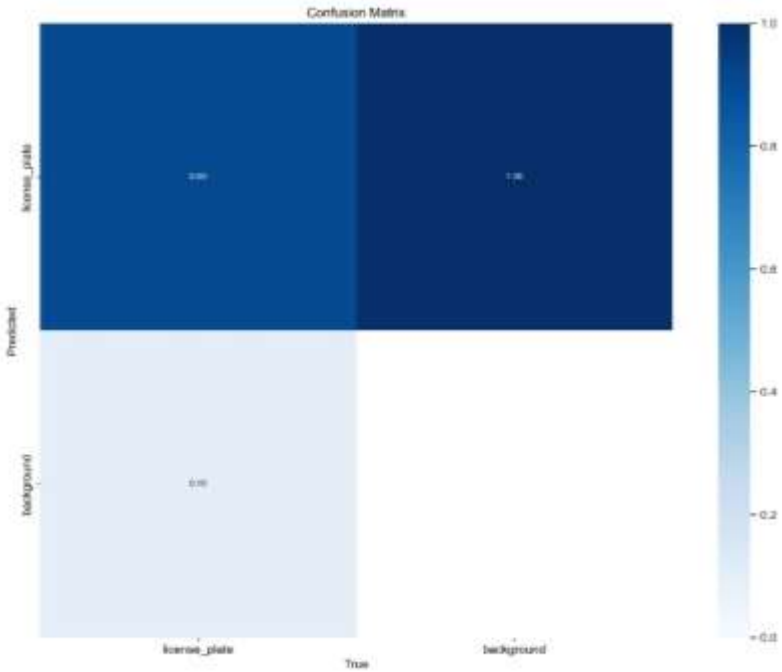
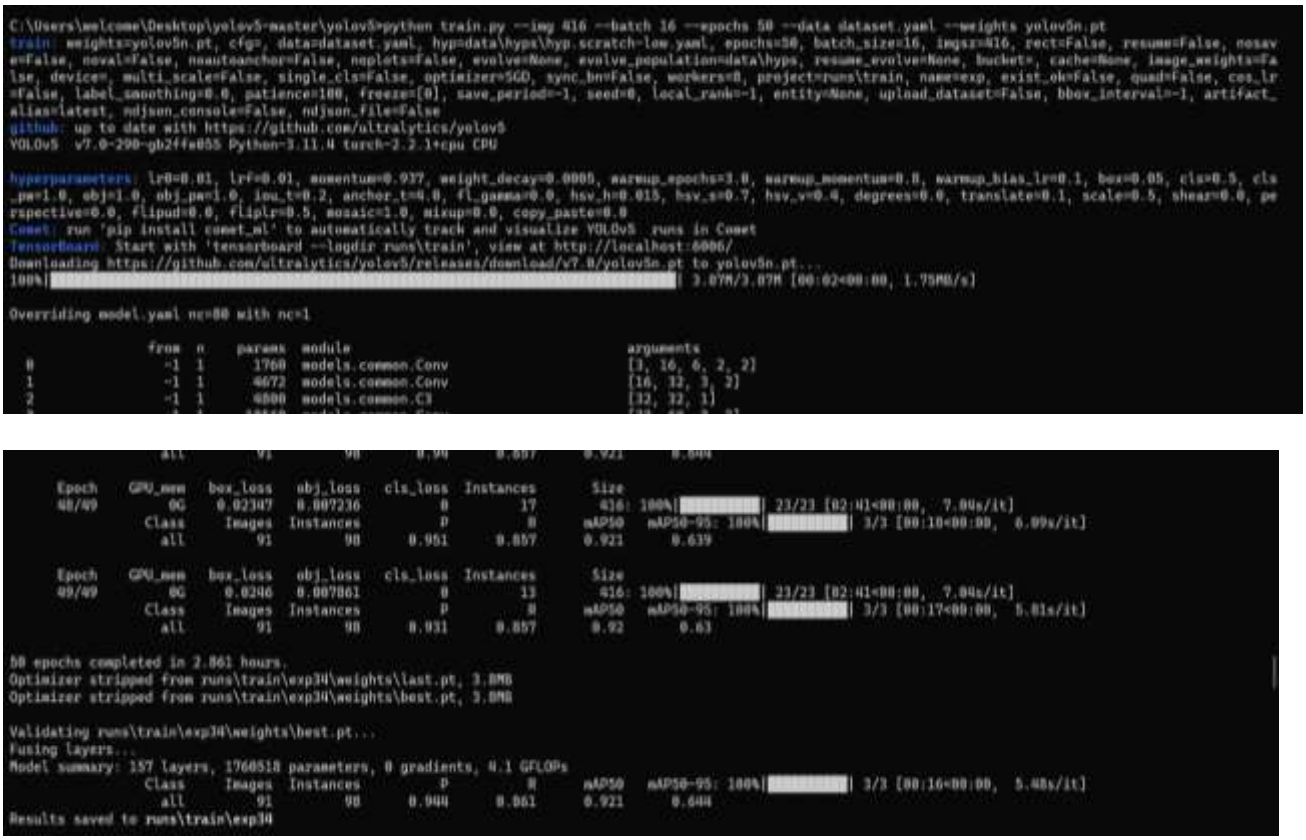


Fig 4.2 Confusion Matrix for YOLOv5s

YOLOv5n:



Accuracy:

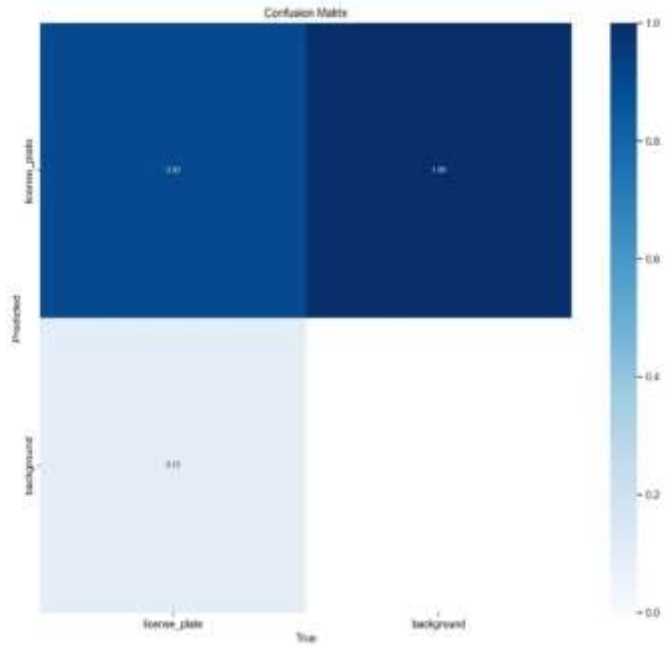


Fig 4.3 Confusion Matrix for YOLOv5n

Annotated Vehicle Images:

Fig4.4 Annotated Number Plates

Validation:

Fig 4.5 Validated Number Plates

CHAPTER 5

RESULTS AND DISCUSSIONS

5.1 Result Analysis

- Based on the accuracy we choose YOLOv5m
- Image undergoes object detection using YOLOv5, resulting in bounding boxes around detected objects
- Extract text from image by OCR
- After implementing DCP algorithm on fog images we will extract characters by using OCR

5.2 Results

Fig 5.1 Image

Fig 5.2 Detecting Bounding Box

Fig 5.3 Fog Image

Fig 5.4 Dehazed Image

Fig 5.5 Recognize Characters on Number Plate

CHAPTER 6
CONCLUSION AND FUTURE PLANS

6.1 CONCLUSION

This methodology combines YOLOv5,OCR, and the Dark Channel Prior algorithm can be used in real-world situations. The system is capable of handling a wide range of tasks by utilising the ability to recognise objects with YOLOv5, text extraction capabilities of OCR, and the haze removal method offered by the Dark Channel Prior Algorithm .

The technique of haze removal improves the quality of images, making it possible to identify objects like number plate more accurately even in fog environment.

This model usefulness can be used for extracting characters from number plates from both photos and videos. Due to its versatility, it may be applied to a variety of sectors and collecting number plate data from surveillance footage or traffic monitoring cameras. Its adaptability to a range of real-world situations, such as image and video analysis, highlights its potential influence on a wide range of sectors and applications.

6.2 FUTURE PLAN

In future we planned to use different deep learning techniques designed for diverse whether conditions, in order to expand number plate recognition beyond foggy surroundings. To increase model performance in foggy environments,the dataset should contains different density of haze images.

Furthermore, improving OCR performance to identify characters in situations with blurry photos and at longer distances will be given priority in order to increase the accuracy of number plate detection. These developments will guarantee reliable number plate detection in a range of environmental circumstances, improving the overall efficacy and security of the system.

REFERENCES

[1] Carrasco, D.P.; Rashwan, H.A.; Garcia, M.A.; Puig, D. T-YOLO: Tiny vehicle detection based on YOLO and multi-scale convolutional neural networks. IEEE Access 2021.

[2] Song, H., Liang, H., Li, H. et al. Vision-based vehicle detection and counting system using deep learning in highway scenes. Eur. Transp. Res. Rev. 11, 51 (2019).

[3] Chong Y, Chen W, Li Z, et al. Integrated Real-Time Vision-Based Preceding Vehicle Detection in Urban Roads[M]. Advanced IntelligentComputing. Springer Berlin Heidelberg, 2011:144-149.

[4] Mingzhi Xu1, Wei Cui2, Jing Xu1 and Wenhan Zhang2 Published under licence by IOP Publishing Ltd Journal of Physics: Conference Series, Volume 2400, 2022 4th International Conference on Artificial Intelligence Technologies and Applications (ICAITA 2022) 18/08/2022 - 21/08/2022 Changchun, China

[5] Li, Q.; Ding, X.; Wang, X.; Chen, L.; Son, J.; Song, J. Detection and Identification of Moving Objects at Busy Traffic Road based on YOLO v4. J. Inst. Internet Broadcasting Commun. 2021, 21, 141–148.

[6] C. Henry, S.W. Lee, Multinational license plate recognition using generalized character sequence detection, IEEE Access 8 (2020) 35185–35199, http://dx.doi.org/10.1109/ACCESS.2020.2974973.

[7] Moghimi M M, Nayeri M, Pourahmadi M, et al. Moving Vehicle Detection Using AdaBoost and Haar-Like Feature in Surveillance Videos[J]. 2018

[8] BoYang,Research on Vehicle Detection and Recognition Technology Based on Artificial Intelligence,Microprocessors and Microsystems,2023,104937,ISSN 0141-9331

[9] Yan, Zhou Deming, Yuan Jun, Zhou 2019/04/28140 978-1-4503-6090-6 Research on Vehicle Identification Method Based on Computer Vision 10.1145/3335656.3335700 ICDMML 2019: Proceedings of the 2019 International Conference on Data Mining and Machine Learning

[10] Samantaray, Milan Biswal, Anil Kumar,Singh, Debabrata Samanta, Debabrata 2022/01/21 Optical Character Recognition (OCR) based Vehicle's License Plate Recognition System Using Python and OpenCV

CHAPTER 7
APPENDIX A Base Paper

Robotics and Autonomous Systems 172 (2024) 104608

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Dual license plate recognition and visual features encoding for vehicle identification

Álvaro Ramajo-Ballester *, José María Armingol Moreno, Arturo de la Escalera Hueso

Intelligent Systems Lab, Universidad Carlos III de Madrid, Spain

ARTICLE INFO

Keywords:
Deep learning
Public dataset
ALPR
License plate recognition
Vehicle re-identification
Object detection

ABSTRACT

This work presents an improved version of a new approach for vehicle identification, which comprises a dual identification system based on license plate recognition and visual encoding. To support this proposal, two new datasets have been created: UC3M-LP for license plate detection and character recognition and UC3M-VRI for vehicle re-identification. The main contributions of this research are the publication of the two open-source datasets and the validation of the dual approach for a reliable vehicle recognition. Precisely, the UC3M-LP dataset is unique, as it fills the gap of European license plates public datasets, becoming the largest of its kind and the first ever for Spanish plates. The proposed dual identification system provides a more robust solution, as it is less sensitive to the variability of image conditions. Performance has been evaluated both on public and the proposed datasets using a multi-network architecture and achieving remarkable results. This strategy opens up new research opportunities in the field of vehicle identification, and the generated datasets may serve as a benchmark for future research. The datasets are publicly available at https://github.com/ramajoballester/UC3M-LP and https://github.com/ramajoballester/UC3M-VRI.

1. Introduction

The development and use of computer vision systems in real-life has increased significantly as a result of recent developments in deep learning, which have substantially improved the accuracy of these systems in comparison to conventional algorithms. In this context, new applications emerge in areas including object and person recognition [1], image-based medical diagnosis [2] or traffic scene analysis in smart cities [3]. The current work concentrates on the latter as the recognition of license plates has been a subject of great interest due to its relevance in several fields, such as security and traffic control.

However, this task is challenging due to variations in lighting, view angle, distance or occlusions, among other factors. These problems make the development of reliable and efficient license plate recognition and vehicle identification systems an ongoing research topic in the field of computer vision. In recent years, machine learning techniques have shown promising results in this area, improving the accuracy and robustness of these systems.

To address these difficulties, the main contributions of this work are the proposed dual vehicle identification system and the open-source datasets to validate the system performance. This dual methodology can perform license plate detection and recognition, as well as visual vehicle identification. The system uses two newly introduced datasets: UC3M-LP for license plate detection and character recognition (OCR) and UC3M-VRI for vehicle re-identification.

The UC3M-LP dataset, in particular, is unique of its kind. It becomes one of the largest publicly available European license plate dataset, as it contains over 20 times more samples than the OpenALPR dataset [4] and a wide variety of conditions. In addition, the labels offer rich information about the scene and a plate polygonal annotation, allowing to refine the detection with a geometric transformation or rectification process.

The proposed dual vehicle identification system provides a more robust solution since it is based on license plate recognition when conditions are favorable and visual identification otherwise. It constitutes an improved version of [5], with the novelty of custom training for all the object detection models. This system has significant implications for video surveillance and traffic scene analysis and can contribute to the development of intelligent infrastructures in smart cities.

It has been trained and tested on the public and proposed datasets, showing remarkable performance in license plate detection, character recognition, and vehicle re-identification.

To provide a comprehensive overview of this work, each section will focus on specific aspects of this research. After putting it in the current state-of-the-art context in Section 2, Section 3 delves into the

* Corresponding author.
E-mail address: aramajo@ing.uc3m.es (Á. Ramajo-Ballester).

https://doi.org/10.1016/j.robot.2023.104608

Available online 12 December 2023

details of the multi-network architecture. After that, Section 4 exposes the methodology and the privacy-related issues of the collected datasets and Section 5 covers all the trainings and hyperparameters tuning. Finally, the results are shown in Section 6.

2. Related work

A thorough state of the art review has been conducted in the fields of Automatic License Plate Recognition (ALPR) and visual vehicle identification.

This review will be presented in two sections: one dedicated to ALPR, which includes a detailed analysis of the latest techniques for license plate detection, segmentation, and optical character recognition; and another one dedicated to visual vehicle identification, which will cover recent developments in the use of deep learning to detect the correspondence of vehicle images to the same identity.

It will also address the challenges associated with these tasks, such as variations in lighting conditions, license plate styles, and vehicle orientations, and discuss the latest trends and open research questions in the field.

2.1. ALPR

Attracting an increasing interest, Automatic License Plate Recognition (ALPR) systems have found their applicability in intelligent transportation systems across many countries for various purposes, including but not limited to traffic law enforcement and traffic monitoring. Moreover, ALPR systems are also implemented to manage the entrance and exit of vehicles in parking areas, collect toll payments, and enforce security measures in restricted areas.

"The general definition of a license plate is 'a metal or plastic plate attached to a vehicle that helps to identify them uniquely'. Yet, this definition is not comprehended by a machine." [6].

Many different research lines have been studied during the past decades to achieve that goal, such as edge-based [7,8], color-based [9–11], character-based [12] or texture-based [13] plate detection tasks. Similarly, regarding to license plate recognition, some of the existing work has focused on character segmentation [14–16] and character recognition [17,18].

Nonetheless, almost all classic techniques have been subdued by the greater precision and speed of deep learning-based approaches, either for plate detection [19–21] and recognition [22,23]. Both tasks are highly correlated since the majority of the top-performance systems rely on object detection models.

These new methods require a great amount of training data, so many efforts have been put into collecting high-quality datasets. On this subject, some of the most used ones include examples from different countries, light conditions and a variety of view angles, like GAP-LP [24], from Tunisia; UFPR-ALPR [19], SSIG-SegPlate [25] and RodoSol-ALPR [26], from Brazil (and Mercosur); PKU Dataset [27], from China; AOLP [28], from Taiwan and other smaller datasets from various locations, such as OpenALPR-EU [4] and CD-HARD [29]. Special mention to CCPD [30] for its 250k images, becoming the largest dataset of this kind and KarPlate [31] for including the largest variety from multiple countries, although is not currently available. All these datasets and their characteristics are shown in Table 1.

However, none of these offer character-wise annotations of European (Spanish) license plates, which constitutes a literature void that this work aims to fill. This will allow to develop precise and effective license plate identification systems specifically designed and trained with European plates format.

2.2. Visual vehicle identification

The necessity of a finer and more rigorous feature extraction process arises when applying visual re-identification, as the variations between different classes of vehicles are comparatively minor in contrast to other common objects, such as different colors and shapes.

Despite the challenges posed by this task, there are several works that have demonstrated high performance, thanks to relying on deep learning techniques. Some of them are based on convolutional neural networks (CNN) and support vector machines (SVM) [32], CNN and long-short term memory (LSTM) bidirectional loop [33], residual networks [34], group sensitive triplet embedding (GSTE) [35], using a two-stage progressive learning approach [36] and with local graph feature aggregation [37].

In this context, some of the widely used datasets are CompCars [38], VehicleID [39], VeRi-776 [40], VeRi-Wild [41], BoxCars21k [42], BoxCars116k [43], Toy Car ReID [33], VRID-1 [44], PKU-VD1 and PKU-VD2 [45], CityFlow [46], Stanford-Cars [47]. Their characteristics are exposed in Table 2.

3. Model architecture

After reviewing some of the current state-of-the-art solutions, the next step is to analyze and explain the bimodal re-identification system architecture. This section will describe the implementation details of each submodule that constitutes the system. All the object detection models are based on the YOLOv5 [48] models family with custom training, which will be presented in Section 5. This is one of the main contributions on the improvement of previous work [5].

The proposed system is composed of four different models and structured in a cascade style to achieve an optimal processing speed and avoid unnecessary forward passes when no vehicle or license plate is detected in previous stages. The overall architecture is shown in Fig. 1.

- Vehicle detection model: the initial object detector processes the input image and produces the bounding boxes for all vehicles that are present in it. It is based on the original YOLOv5-m [48], where only 4 categories (car, motorcycle, bus and truck) are used.
- LP detection model: once the vehicle regions have been extracted, the first parallel branch performs the license plate recognition. The first model in the branch detects the license plate and crops it. A custom object detector has been trained for that purpose.
- OCR recognition model: after plate cropping, another custom YOLOv5 model identifies and sorts its characters to complete the license plate recognition.
- Visual identification: the second parallel branch encodes the previously vehicle image region according to its visual characteristics into a feature vector. The similarity between different vehicles will be measured by the Euclidean distance between their vectors. This metric is used to discriminate if two vehicle images correspond to the same identity or not, performing then the visual identification.

The system workflow is depicted in Fig. 2. The visual re-identification branch performs inference on the vehicle region only when license plate character recognition yields no favorable results. Blocks color code matches models in Fig. 1.

3.1. Vehicle detection

The detection of regions of interest (ROI) is manifold in this work. Firstly, the aim is to identify the rectangular boxes within the image that correspond to a vehicle and to search for the cutout of the vehicle that corresponds to the license plate and its characters afterwards.

Even so, and since the main objective is the bimodal processing of already obtained vehicle images, vehicle detection optimization was out of the scope of this work. With that in mind, the original and pre-trained YOLO model has been used for this task.

Table 1
License plate identification dataset comparison.

Dataset	Location	Images	Year	Notes
AOLP [28]	Taiwan	2,049	2013	3 different sets, depending on pan, tilt angles and distance
SSIG-SegPlate [25]	Brazil	2,000	2016	14k available characters, some have been blurred for privacy constrains
OpenALPR-EU [4]	Europe	108	2016	Only European dataset
PKU Dataset [27]	China	3,977	2017	Mostly frontal images
CCPD [30]	China	250,000	2018	Includes view angle annotations and 720 × 1160 resolution
UFPR-ALPR [19]	Brazil	4,500	2018	1920 × 1080 image resolution
CD-HARD [29]	International	102	2018	Oblique view, plate-centered crops
GAP-LP [24]	Tunisia	9,175	2019	Contains Arabic and Latin chars
KarPlate [31]	International	8,613	2020	Not available due to legal reasons
RodoSol-ALPR [26]	Brazil/Mercosur[a]	20,000	2022	Multiple countries and 1280 × 720 resolution

[a] Mercosur includes Argentina, Brazil, Paraguay, Uruguay and Venezuela.

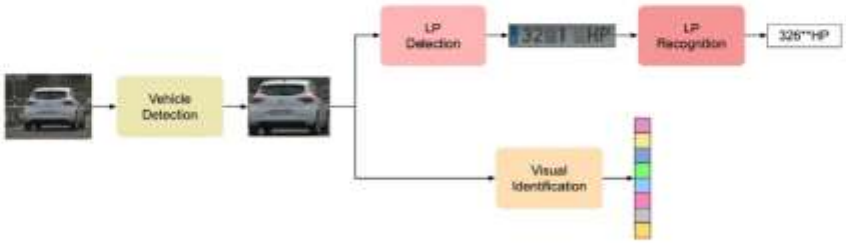


Fig. 1. Multi-network system architecture.

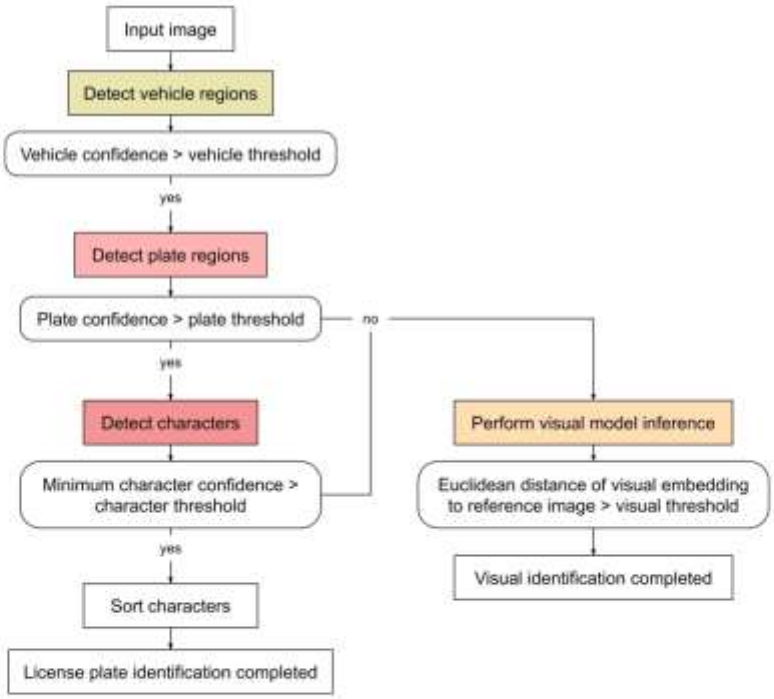


Fig. 2. System workflow diagram.

3.2. License plate recognition

As mentioned before, the first parallel branch carries out the plate recognition for all detected vehicles in the previous stage. There are two specific tasks that need to be solved. The first one, is to correctly locate and define its bounding box. The second, detect, classify and sort each character to conform a unique identification number.

Detection in both cases is carried out by custom versions of YOLOv5 [48]. The main reason for its selection are its speed and accuracy over other models such as YOLOv4 [49], YOLOv3 [50],

Table 2
Vehicle re-identification datasets comparison.

Dataset	Cameras	Images	IDs	Year
Stanford-Cars [47]	–	16,185	196	2013
CompCars [38]	–	136,713	4,701	2015
VehicleID [39]*	2	221,567	26,328	2016
VeRi-776 [40]	20	49,357	776	2016
BoxCars-21k [42]	–	63,750	21,250	2016
VRID-1 [44]	326	10,000	1,000	2017
PKU-VD1 [45]	1	1,097,649	1,232	2017
PKU-VD2 [45]	1	807,260	1,112	2017
Toy Car ReID [33]b	50	30,000	200	2018
VeRi-Wild [41]	174	416,314	40,671	2019
BoxCars-116k [43]	137	116,286	27,496	2019
CityFlow [46]	40	56,277	666	2019

* [36] points out that the real number of images differs slightly from the original publication [39].
b Synthetic dataset.

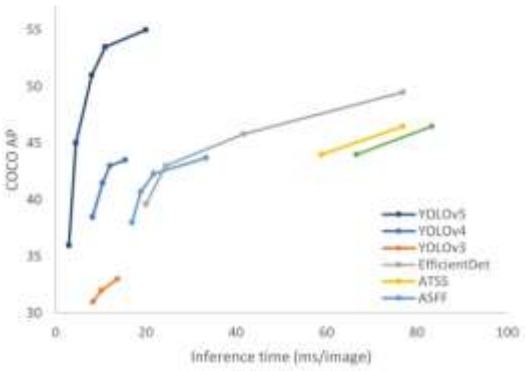


Fig. 3. Inference time and average precision in detection models.

EfficientDet [51], ATSS [52], ASFF [53] or CenterMask [54], as it can be seen in Fig. 3.

3.3. Visual identification

Once the vehicle's license plate number has been verified, the system proceeds to identify the vehicle using a visual recognition model. This is done by comparing the feature vectors of each image processed with that of the target vehicle. The system calculates the Euclidean distance between the vectors to successfully re-identify the vehicle.

The main objective is being able to discern whether two vehicle identities are the same from an infrastructure point of view and with different cameras.

4. Datasets acquisition and labeling

The acquisition high-quality datasets is crucial in the development of machine learning models for object detection and recognition tasks. In this project, two datasets were obtained and manually labeled, one for license plate detection and character recognition, and another one for vehicle visual identification. The UC3M-LP dataset was used for license plate detection and character recognition tasks, while UC3M-VRI for vehicle visual identification. This section will describe the datasets acquisition process, including the data collection and annotation methods used.

The treatment and publication of images that contain personal information is not straight-forward, so a particular emphasis was placed on implementing robust protocols and procedures to meticulously anonymize the data, thereby eliminating any potential traces of personally identifiable information. These efforts were aimed at safeguarding the privacy and anonymity of individuals whose data was included in the dataset, and will be described in Sections 4.1 and 4.2. The labeling process has been done with Labelme [55], for both polygonal and bounding box annotations.

Table 3
UC3M-LP dataset features.

Images	1975	
Labels	Plates	Plate characters
- Type	polynomial	bounding box
- Count	2547	12757
Lighting	Day	Night
- Count	2185	362
Distance	1m – 20m	
Perspective	Various: frontal and oblique (up to 70° angle)	

4.1. License plate recognition: UC3M-LP dataset

The proposed UC3M-LP dataset for license plate recognition has been gathered from a multitude of sources, including both smartphone cameras and professional cameras. This dataset encompasses a good variety of perspectives, light conditions, distances, and resolutions, resulting in a significant challenge when it comes to effectively processing and interpreting the images, as it is summarized in Table 3.

Preserving the anonymity of data in this context is not trivial, since blurring the entire license plate is not an option — as there would be no dataset at all. The privacy measures that have been taken include blurring one digit and one character from the plate and distorting the image by adding color noise. Fig. 4 shows a sample of this dataset.

Having samples with different difficulty allows the model to learn more complex scenarios and increase the overall performance. Bottom images in Fig. 4 show some of the most complicated examples.

4.1.1. Detection

Through the use of polygonal annotation, the first type of the dataset annotations has been tailored to suit the training of both detection systems and post-processing models, such as the well-known Spatial Transformer Networks (STN) [56]. In this way, it has been crafted to optimize the accuracy and efficiency of license plate detection.

In Fig. 5, it is shown how the polygonal label allows, for instance, the neural networks training to perform the rectification of plates applying an affine transformation.

It comprises 2547 license plates from 1975 images of different resolutions, with 2185 out of them are from daytime (D) and 362 from nighttime (D). Fig. 6 exposes the license plate variety in the Spanish typology:

- Type A: 2498 samples of the most common long, one row with white background (Fig. 6(a)).
- Type B: 31 samples of motorcycle double row and white background (Fig. 6(b)).
- Type C: 1 sample of light motorcycle one row with yellow background (Fig. 6(c)).
- Type D: 11 samples of taxis and VTC (Spanish acronym for private hire vehicle) with blue background (Fig. 6(d)).
- Type E: 6 samples of trailer tows with black characters and red background (Fig. 6(e)).

This information is encoded in the label as a prefix to the plate number. For instance, a AN-000**ZZ would mean type A (one row, white background) and N (nighttime).

Fig. 4. Proposed UC3M-LP dataset sample.

Fig. 5. Polygonal annotation format and rectification process.

Fig. 6. License plate types.

4.1.2. Optical character recognition (OCR)

The license plate character labels comprise the second type of annotations in this dataset. These labels are intended to facilitate object detection-based optical character recognition (OCR). The label format for this type is an orthogonal bounding box. The 2547 plates include 12757 different labels for the 37 different letters and numbers, enabling the training of models for effective license plate identification.

This has been the most delicate part since data protection and privacy considerations have been taken into account. For this reason, several characters have been blurred from each license plate as well as faces and other personal information, as mentioned in Section 4.1, following the current European regulations. The anonymized version of the dataset is the one used in this work. A good example of this label format can be seen in Fig. 7.

The data distribution of these characters is categorized in frequency per class in Fig. 8. As there are less digit classes and more appearances

Fig. 7. OCR bounding box annotation.

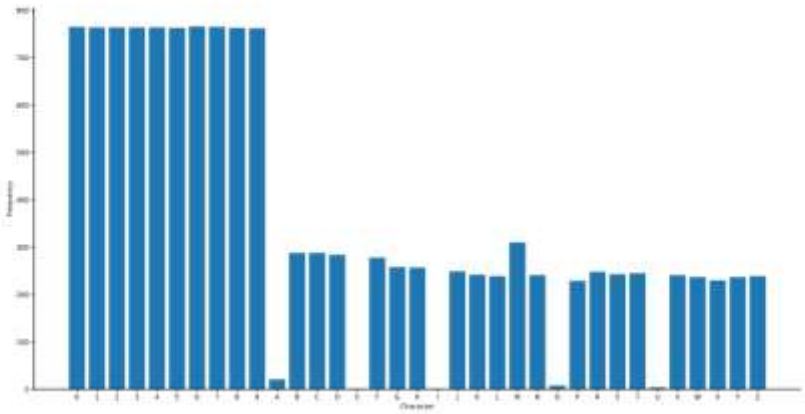


Fig. 8. UC3M-LP dataset data distribution.

Table 4
UC3M-VRI dataset features.

Images	1611	
Label type	Bounding box	
Lighting	Day	
Locations	**Road**	**Intersection**
Images	458	1153
IDs	201	85
Distance	5m – 7m	10m – 20m
Input cameras	1	2
Perspective	Rear oblique	All vehicle orientations

of them, their frequency is higher than the characters classes. An important aspect to point out in this data distribution is that there are few examples of vowels, since they only appear in rare occasions in old plates. Since the data protection protocols required the blurring of several plate characters, it has been done with the complementary aim of balancing the dataset classes as much as possible with a best-effort strategy.

4.2. Vehicle visual identification: UC3M-VRI

To minimize the difference between the evaluation settings of the system and the actual production environment, a custom dataset, namely UC3M-VRI, has been gathered and manually labeled to assess the performance of the visual re-identification system. Two different sets constitute this dataset, each of which have distinct features such as vehicle similarity, perspective, illumination, and the number of input cameras, as shown in Table 4.

Given the specific use of this dataset, blurring license plates is not a restriction, so that has been the chosen anonymization method.

4.2.1. Road set

The initial dataset comprises images captured from a pole on a highway, from an elevated, oblique, and rear angle. This set is characterized by a high degree of similarity between images of the same category, as they exhibit the same perspective, uniform lighting, and no obstructions. It is intended to serve as the first stage in the evaluation process, as it is comparatively less challenging and expected to yield more positive outcomes. The dataset encompasses 458 images depicting 201 vehicle models, and a selection of these images is presented in Fig. 9.

4.2.2. Intersection set

This second dataset comprises traffic scenes captured at intersections. It is divided into two distinct recording locations (v1 and v2)

Fig. 9. Road Dataset sample.

and offers a variety of perspectives and occlusions between vehicles and vegetation. Each scene has been simultaneously captured by two cameras (c1 and c2) and represents a high degree of difficulty as it depicts a typical operational environment. Having two input sources enables searching for annotated vehicles from one camera in the other with a different perspective, which is the primary objective of this study. The dataset includes a total of 1153 images and 85 classes with slightly different annotation criteria. In v1, all vehicle appearances, including very distant and partial views, are included, whereas in v2 only complete vehicles with a minimum recognizable size are annotated (Fig. 10).

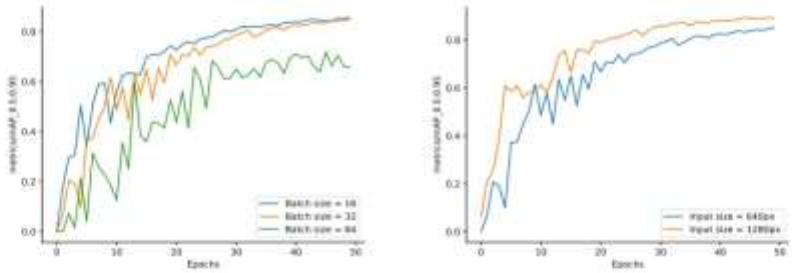
5. Neural networks training

The training process can be influenced by several factors, such as the choice of optimizer, learning rate, batch size, and architecture, among others.This section will discuss the training process and highlight the key factors that affect the models performance.

5.1. Object detection models

As both plate detection and OCR are based on object detection methods, similar training procedures have been followed in each of them. Several trainings have been tested to understand the effect of hyperparameters tuning in the validation results. Sections 5.1.1 and

Fig. 10. Intersection Dataset sample.



(a) Batch size (b) Input image size

Fig. 11. Hyperparameters tuning effect for LP detection.

5.1.2 present this repercussions in the validation set according to each hyperparameter.

5.1.1. License plate detection

Fig. 11 points out the impact of batch size and input image size. A bigger batch size it not always desirable, since the generalization ability can be compromised in several ways, as Fig. 11(a) suggests. Increasing the batch size can lead to changes in the learning dynamics of the model. For example, larger batch sizes can lead to flatter minima in the optimization landscape, which may cause the model to converge to a sub-optimal solution that is less generalizable. The input size does improve the model performance, as the higher the resolution, the finer the details that can be extracted.

Fig. 12(a) reflect the performance difference depending on the optimization process and the model size. Regarding the optimizer choice, SGD [57] performs slightly better than AdamW [58], which in turn performs a bit better than Adam [59] (Fig. 12(a)). This trend may be due to the different strategies employed by each one. SGD, for instance, uses a simpler strategy of gradient descent, while Adam and AdamW use more advanced techniques such as adaptive learning rates and momentum, which in this particular case with a small dataset may not deploy their potential. However, the performance differences were relatively small, indicating that the choice of optimizer may not be as critical as other factors. This amount of data may be the reason why larger models do not perform better, as it is depicted in Fig. 12(b). In these cases, the models tend to overfit and being surpassed by the smaller YOLOv5-s. Nonetheless, the smaller is not the better since YOLOv5-n shows a clear underfitting.

Apart from the performance, YOLOv5-s is a small enough model to run easily in real time, even in low-end devices.

5.1.2. Character recognition

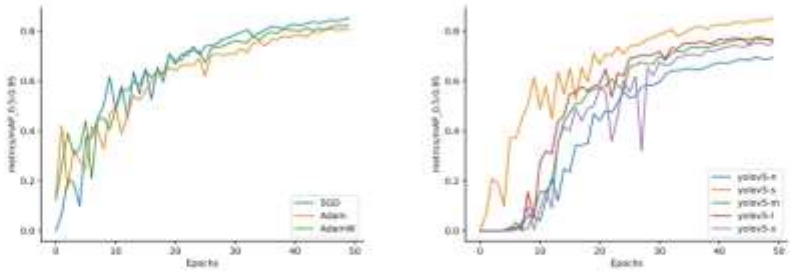
The same training trials as in Section 5.1.1 were applied to the OCR task. In Fig. 13, the impact of batch size and image input size is shown. The effect of batch size is similar than in detection, although it is affects the training speed, as the final results are pretty similar. A reduced batch size can cause the model to converge faster, as the gradients are updated more frequently, which can lead to more efficient use of computational resources. In terms of image input size, there is no difference, since not many license plates have a bigger dimensions of 640 pixels, so increasing its size does not provide a more detailed view of the image.

On the contrary of Section 5.1.1, Fig. 14(a) highlights a performance drop when using SGD in favor of adaptive optimizers. As it is widely known, this kind of optimization strategies tend to be faster than Gradient Descent. In the previous case, as images were larger, SGD may be a better solution since it shows a bit greater robustness and less tendency to overfit with larger batch sizes or learning rates. On the other hand, Fig. 14(b) exhibits no significant difference when comparing different model sizes, apart from the underfitting of the smallest YOLOv5-n.

5.2. Visual identification model

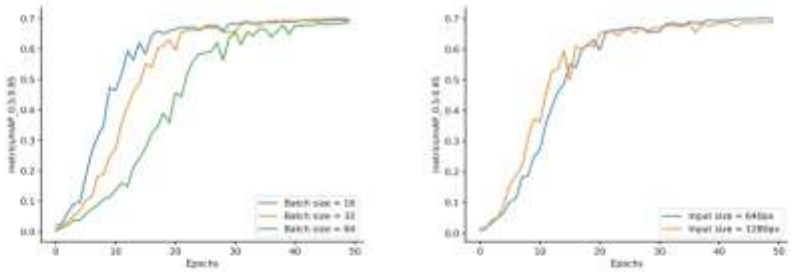
In order to achieve the visual identification, a comparison between some state-of-the-art models and several trained backbone models has been tested. The FastReid Toolbox [60] provides pre-trained and optimized architectures specifically for vehicle identification purposes. To meet the desired standards, different training strategies were employed with the EfficientNet [61] family of neural networks. One notable feature of this architecture is its ability to precisely scale network dimensions, which enhances performance. Fig. 15 illustrates that standard convolutional networks increase the width (b) of the feature map to improve performance, whereas others opt to add more intermediate layers (c) or higher resolution images (d) to make the network deeper. These models were used as the backbone during the training process, with max pooling and convolutional layers appended to their output for fine tuning.

The initial training was carried out with the Stanford-Cars dataset [47], which is widely used in the current state of the art as discussed in the previous section. The first training was conducted with a reduced version of the dataset (approximately 10%) to adjust the



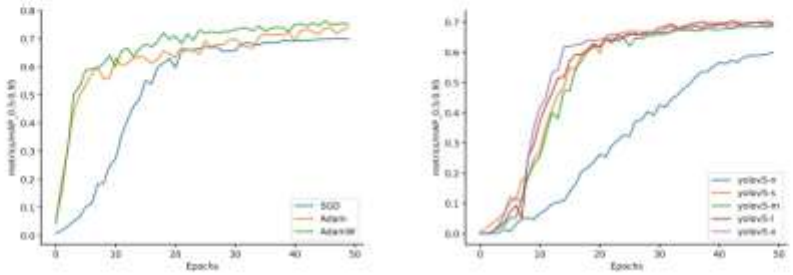
(a) Optimizer (b) Model size

Fig. 12. Hyperparameters tuning effect for LP detection (2).



(a) Batch size (b) Input image size

Fig. 13. Hyperparameters tuning effect for LP OCR.



(a) Optimizer (b) Model size

Fig. 14. Hyperparameters tuning effect for LP OCR (2).

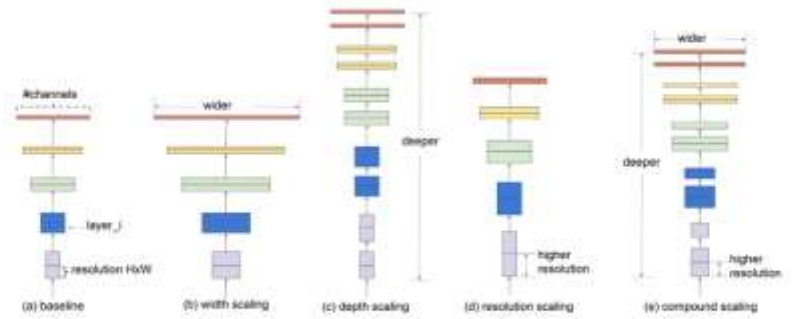


Fig. 15. EfficientNet architecture [61].

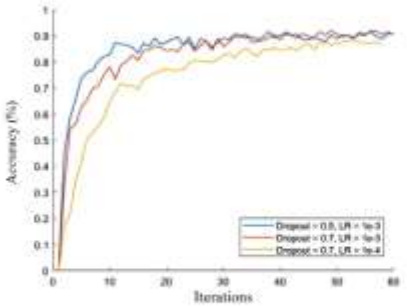


Fig. 16. Dropout and learning rate effect.

dropout and learning rate. Dropout refers to the ratio of neural networks in certain layers that are randomly "turned off" during training, which aids in the extraction of characteristics through several routes and helps the model generalize better. The learning rate determines the speed at which the weights are updated. A reduced value allows for the addition of more weights, but at the cost of a longer training time, hence, it is advisable to optimize it. All the aforementioned tests are depicted in Fig. 16.

The graph presents two notable results. Firstly, the networks with a dropout of 0.7 exhibit slightly better generalization than those with 0.5. Despite taking slightly longer in the initial epochs, the trend favors the former as it reaches a significantly lower maximum with 0.5. Additionally, a learning rate of $1e^{-3}$ is found to be the most suitable as it maximizes precision more quickly. Several other tests were conducted but are omitted to avoid prolonging the training demonstration. Following these tests, the performance of three versions of the EfficientNet network (B0, B3, and B7) were evaluated. The output was configured with maximum global pooling for each output filter and a dense classification layer with the previously adjusted dropout. The results are presented in Fig. 17(a).

The graph illustrates that the performance of all three models (B0, B3, and B7) is comparable. Given the increased size and slower processing speed of the larger models, it makes sense to choose the B0 model for use as the characterization network, especially since a real-time system is desired.

Moreover, the VeRi-776 dataset [62] was used to perform another training round with the same networks. However, to simplify the training process, the output classes were slightly modified. The network was originally designed for classification purposes, so the last softmax layer was removed, letting the model encode the input image with the penultimate layer output. The results are presented in Fig. 17(b), although the comparative evaluation will be shown in Section 6.

As it can be seen, the accuracy is similar between the three models, so EfficientNetB0 is chosen for the same reasons. The same procedure has been tested with a new manually labeled dataset, and the results will be shown in Section 6.

6. Results and discussion

This section will show the final results of the training process, as well as a thorough analysis of the effect of key factors that contribute to a optimal performance.

6.1. License plate recognition

The following results and analysis have been carried out with the proposed UC3M-LP dataset.

Table 5
Result metrics for license plate detection (sorted by mAP@0.5:0.95).

Model	mAP@0.5:0.95	mAP@0.5	precision	recall	F1
Image size = 1280 px	**0.893**	**0.988**	**0.961**	**0.965**	**0.963**
Batch size = 16	0.856	0.985	0.953	0.959	0.956
Baseline*	0.852	0.982	0.941	**0.965**	0.953
AdamW	0.821	0.975	0.922	0.952	0.937
Adam	0.809	0.971	0.935	0.939	0.937
YOLOv5-l	0.774	0.956	0.924	0.911	0.918
YOLOv5-m	0.771	0.944	0.920	0.911	0.916
YOLOv5-x	0.753	0.943	0.907	0.899	0.903
YOLOv5-n	0.695	0.917	0.930	0.867	0.897
Batch size = 64	0.684	0.968	0.929	0.948	0.938

* Baseline model is a YOLOv5-s architecture, batch size = 32, SGD optimizer and input image size of 640 px.

Table 6
Result metrics for license plate recognition (sorted by mAP@0.5:0.95).

Model	mAP@0.5:0.95	mAP@0.5	precision	recall	F1
AdamW	**0.764**	**0.976**	0.943	**0.972**	**0.957**
Adam	0.740	0.962	0.979	0.914	0.946
YOLOv5-l	0.704	0.926	0.988	0.884	0.933
Baseline*	0.701	0.911	0.988	0.889	0.936
YOLOv5-x	0.699	0.922	0.989	0.885	0.934
Batch size = 16	0.696	0.904	**0.990**	0.890	0.937
Batch size = 64	0.689	0.901	0.986	0.885	0.933
Image size = 1280 px	0.689	0.901	0.985	0.882	0.931
YOLOv5-m	0.688	0.909	0.988	0.878	0.930
YOLOv5-n	0.601	0.816	0.737	0.818	0.775

* Baseline model is a YOLOv5-s architecture, batch size = 32, SGD optimizer and input image size of 640 px.

6.1.1. Detection

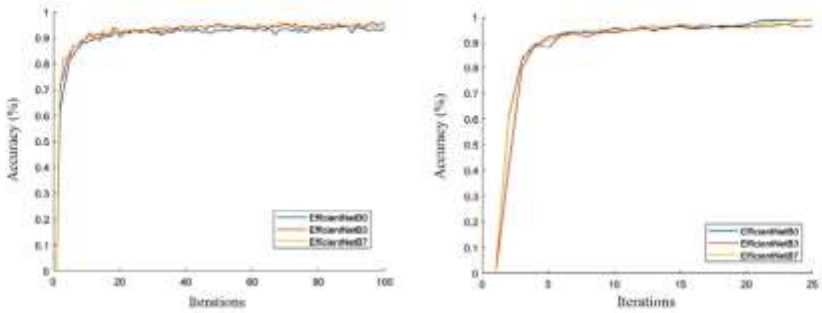
The results of the license plate detection models exhibit a range of performances across the different configurations tested (Table 5). The best performing model achieved a mean average precision (mAP) of 0.893 at an intersection over union (IoU) threshold of 0.5 to 0.95 and a mAP of 0.988 at an IoU of 0.5. Notably, the image size of 1280 px produced the best results, while the larger batch size of 64 resulted in the worst performance. This suggests that smaller batch sizes may be more effective in training license plate detection models. Interestingly, the baseline model performed similarly to the Adam, AdamW, and YOLOv5-l models, indicating that more complex models and optimization strategies may not necessarily lead to improved performance in this task. Overall, these results highlight the importance of carefully selecting model configurations and optimizing hyperparameters for achieving optimal performance in license plate detection.

Since a real-time performance was in the scope of this work, the inference time is shown for each model. For the license plate detection stage it is 15.6 ms. However, it is noted that it was performed in a NVIDIA RTX 3090 GPU. The inference time was computed with a single image batch, so a multiple image batch would exhibit a lower time per image.

6.1.2. Character recognition

The OCR results in Table 6 show that the AdamW model outperformed the other models in terms of mAP at 0.5:0.95, achieving a score of 0.764. The Adam model also performed well with an mAP of 0.74. The YOLOv5-l, Baseline, and YOLOv5-x models achieved similar mAP scores, with values ranging from 0.699 to 0.704. As it can be seen in the table, increasing the batch size to 16 or 64 and the image size to 1280 px did not result in significant improvements in performance, as it was explained in Section 5.1.2. The YOLOv5-n model had the lowest mAP score of 0.601, as a consequence of underfitting problems due to its reduced size. Overall, the results suggest that the models with adaptive momentum optimizers are the most effective for license plate character recognition.

The license plate detection takes approximately 10.7 ms on a single image batch.



(a) Stanford-Cars dataset (b) VeRi-776 dataset

Fig. 17. EfficientNet B0-B3-B7 results.

Table 7
Accuracy in positive–negative pair test in public datasets.

Model	Stanford-Cars	VeRi-776	VeRi-Wild
EfficientNetB0 (Stanford-Cars)	**0.835**	0.625	0.727
EfficientNetB0 (VeRi-776)	0.591	0.772	0.797
FastReid (VeRi-776)	0.606	**0.968**	0.908
FastReid (VeRi-Wild)	0.676	0.905	**0.995**

Table 8
Accuracy in positive–negative pair test in custom datasets.

Model	Road	Int. v1c1	Int. v2c1	Int. v1	Int. v2
EfficientNetB0 (Stanford-Cars)	0.851	0.736	0.841	0.625	0.619
EfficientNetB0 (VeRi-776)	0.966	0.882	0.916	0.715	0.781
FastReid (VeRi-776)	**0.979**	**0.940**	**0.966**	**0.878**	**0.915**
FastReid (VeRi-Wild)	0.967	0.909	0.897	0.788	0.825

6.2. Vehicle visual identification

Table 7 shows the accuracy of the two trained models (EfficientNetB0) versus the pre-trained FastReid models on the public datasets. This evaluation corresponds to the accuracy of a positive–negative pair test. Each positive–negative pair has been created with each image from the evaluation set, an image from its class (positive) and a random image from the rest of the classes (negative). From these results it can be extracted that the FastReid model pre-trained with VeRi-Wild, which is a bigger dataset and with fewer constraints than VeRi-776, is a better candidate.

Nevertheless, once the evaluation is performed with the proposed UC3M-VRI dataset, which is much closer to the real production environment, the metrics favor the FastReid model pre-trained with VeRi-776, achieving the best results, as shown in Table 8.

This last visual identification stage carries most of the processing time, with an average of 31.8 ms, totaling 58.1 ms for the complete system.

In the introduction, it was noted that recognizing license plates is a challenging task that requires favorable imaging conditions. Nevertheless, it is important to recognize that not all images can meet these requirements. That is where the visual recognition system comes in. This system offers greater flexibility in terms of operational constraints and can deliver outstanding performance in more adverse situations. By analyzing the visual characteristics of the whole vehicle, including its shape and color, this system is less sensitive to distance, which means it can extend the valid recognition range from 15 to 40 m in four lanes of a wide-angle camera view. Moreover, this system allows us to track vehicles in subsequent video frames based on their similarity to previous ones, which is a unique feature of our approach. In short, the proposed system provides a more robust solution than traditional approaches, making it an exciting breakthrough in vehicle identification.

7. Conclusions

In this work, two new datasets for license plate detection and character recognition (OCR) and vehicle re-identification, namely UC3M-LP and UC3M-VRI, respectively are proposed. These datasets have been designed to give a more extensive and challenging benchmark for evaluating license plate recognition systems and vehicle identification methods. They feature a wide variety of scenarios, including challenging lighting conditions, different camera angles, and partial occlusions, which better simulate real-world scenarios.

Particularly, the UC3M-LP dataset becomes the largest publicly available dataset dedicated to European license plate, as it is more than 20 times bigger than the popular OpenALPR [4], and the first ever to Spanish ones specifically.

Moreover, this work provides a dual vehicle identification system that is capable of providing more robust identification results. This system is based on license plate recognition when the conditions are favorable, while visual identification is used in other situations. Since the license number is widely considered the most effective method of vehicle identification, visual recognition adds another layer of confidence to the system's capabilities. The proposed visual method extracts the visual characteristics of the whole vehicle, including its shape and color, to identify vehicles in the absence of a readable license plate. This method offers a distinctive advantage over classic approaches, as it offers a more versatile solution.

In summary, the proposed datasets and the dual vehicle identification system constitute a significant contribution to the field of computer vision. Our datasets enable researchers to benchmark their algorithms on more challenging scenarios, while the proposed dual vehicle identification system provides a more versatile solution for vehicle identification. These contributions may lead to further developments in the field, ultimately improving the performance and reliability of license plate recognition and vehicle identification systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be publicly released. Link to data in the paper.

Acknowledgments

Grant PID2019-104793RB-C31, PDC2021-121517-C31, PDC2022-133684-C31 and PID2021-124335OB-C21 funded by MCIN/AEI/10.13039/5011000 11033 and by the European Union "NextGenerationEU/PRTR".

References

[1] Y. Xu, Z. Piao, S. Gao, Encoding crowd interaction with deep neural network for pedestrian trajectory prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5275–5284, https://openaccess.thecvf.com/content_cvpr_2018/html/Xu_Encoding_Crowd_Interaction_CVPR_2018_paper.html.

[2] A.S. Lundervold, A. Lundervold, An overview of deep learning in medical imaging focusing on MRI, Z. Med. Phys. 29 (2) (2019) 102–127, http://dx.doi.org/10.1016/j.zemedi.2018.11.002.

[3] S. Yamin Siddiqui, M. Adnan Khan, S. Abbas, F. Khan, Smart occupancy detection for road traffic parking using deep extreme learning machine, J. King Saud Univ., Comput. Inf. Sci. 34 (3) (2022) 727–733, http://dx.doi.org/10.1016/j.jksuci.2020.01.016.

[4] OpenALPR, OpenALPR-EU Dataset, 2016, https://github.com/openalpr/benchmarks/tree/master/endtoend/eu.

[5] Á. Ramajo Ballester, J. González Cepeda, J.M. Armingol Moreno, Deep learning for robust vehicle identification, in: D. Tardioli, V. Matellán, G. Heredia, M.F. Silva, L. Marques (Eds.), Proceedings of ROBOT2022: Fifth Iberian Robotics Conference, in: Lecture Notes in Networks and Systems, Springer International Publishing, Cham, 2023, pp. 346–358, http://dx.doi.org/10.1007/978-3-031-21065-5_29.

[6] J. Shashirangana, H. Padmasiri, D. Meedeniya, C. Perera, Automated license plate recognition: a survey on methods and techniques, IEEE Access 9 (2021) 11203–11225, http://dx.doi.org/10.1109/ACCESS.2020.3047929.

[7] D. Zheng, Y. Zhao, J. Wang, An efficient method of license plate location, Pattern Recognit. Lett. 26 (15) (2005) 2431–2438, http://dx.doi.org/10.1016/j.patrec.2005.04.014.

[8] M. Sarfraz, M.J. Ahmed, S.A. Ghazi, Saudi Arabian license plate recognition system, in: 2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings, 2003, pp. 36–41, http://dx.doi.org/10.1109/GMAG.2003.1219663.

[9] S. Yohimori, Y. Mitsukura, M. Fukumi, N. Akamatsu, N. Pedrycz, License plate detection system by using threshold function and improved template matching method, in: IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS '04, Vol. 1, 2004, pp. 357–362, http://dx.doi.org/10.1109/NAFIPS.2004.1336308.

[10] W. Jia, H. Zhang, X. He, Q. Wu, Gaussian weighted histogram intersection for license plate classification, in: 18th International Conference on Pattern Recognition, ICPR'06, vol. 3, 2006, pp. 574–577, http://dx.doi.org/10.1109/ICPR.2006.596.

[11] F. Wang, L. Man, B. Wang, Y. Xiao, W. Pan, X. Lu, Fuzzy-based algorithm for color recognition of license plates, Pattern Recognit. Lett. 29 (7) (2008) 1007–1020, http://dx.doi.org/10.1016/j.patrec.2008.01.026.

[12] J. Matas, K. Zimmermann, Unconstrained licence plate and text localization and recognition, in: Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005, 2005, pp. 225–230, http://dx.doi.org/10.1109/ITSC.2005.1520111.

[13] Y.-R. Wang, W.-H. Lin, S.-J. Horng, A sliding window technique for efficient license plate localization based on discrete wavelet transform, Expert Syst. Appl. 38 (4) (2011) 3142–3146, http://dx.doi.org/10.1016/j.eswa.2010.08.106.

[14] T. Nukano, M. Fukumi, M. Khalid, Vehicle license plate character recognition by neural networks, in: Proceedings of 2004 International Symposium on Intelligent Signal Processing and Communication Systems, 2004. ISPACS 2004, 2004, pp. 771–775, http://dx.doi.org/10.1109/ISPACS.2004.1439164.

[15] I. Paliy, V. Turchenko, V. Koval, A. Sachenko, G. Markowsky, Approach to recognition of license plate numbers using neural networks, in: 2004 IEEE International Joint Conference on Neural Networks, IEEE Cat. No.04CH37541, Vol. 4, 2004, pp. 2965–2970, http://dx.doi.org/10.1109/IJCNN.2004.1381137.

[16] J. Tian, R. Wang, G. Wang, J. Liu, Y. Xia, A two-stage character segmentation method for Chinese license plate, Comput. Electr. Eng. 46 (2015) 539–553, http://dx.doi.org/10.1016/j.compeleceng.2015.02.014.

[17] P. Hu, Y. Zhao, Z. Yang, J. Wang, Recognition of gray character using Gabor filters, in: Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002., IEEE Cat.No.02EX5997, Vol. 1, 2002, pp. 419–424, http://dx.doi.org/10.1109/ICIF.2002.1021184.

[18] D. Llorens, A. Marzal, V. Palazón, J.M. Vilar, Car license plates extraction and recognition based on connected components analysis and HMM decoding, in: J.S. Marques, N. Pérez de la Blanca, P. Pina (Eds.), Pattern Recognition and Image Analysis, in: Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2005, pp. 571–578, http://dx.doi.org/10.1007/11492429_69.

[19] R. Laroca, E. Severo, L.A. Zanlorensi, L.S. Oliveira, G.R. Gonçalves, W.R. Schwartz, D. Menotti, A robust real-time automatic license plate recognition based on the YOLO detector, in: 2018 International Joint Conference on Neural Networks, IJCNN, 2018, pp. 1–10, http://dx.doi.org/10.1109/IJCNN.2018.8489629.

[20] S.M. Silva, C.R. Jung, Real-time license plate detection and recognition using deep convolutional neural networks, J. Vis. Commun. Image Represent. 71 (2020) 102773, http://dx.doi.org/10.1016/j.jvcir.2020.102773.

[21] L. Xie, T. Ahmad, L. Jin, Y. Liu, S. Zhang, A new CNN-based method for multi-directional car license plate detection, IEEE Trans. Intell. Transp. Syst. 19 (2) (2018) 507–517, http://dx.doi.org/10.1109/TITS.2017.2784093.

[22] R. Laroca, L.A. Zanlorensi, G.R. Gonçalves, E. Todt, W.R. Schwartz, D. Menotti, An efficient and layout-independent automatic license plate recognition system based on the YOLO detector, IET Intell. Transp. Syst. 15 (4) (2021) 483–503, http://dx.doi.org/10.1049/itr2.12030.

[23] C.N.E. Anagnostopoulos, I.E. Anagnostopoulos, V. Loumos, E. Kayafas, A license plate-recognition algorithm for intelligent transportation system applications, IEEE Trans. Intell. Transp. Syst. 7 (3) (2006) 377–392, http://dx.doi.org/10.1109/TITS.2006.880641.

[24] Y. Kessentini, M.D. Besbes, S. Ammar, A. Chabbouh, A two-stage deep neural network for multi-norm license plate detection and recognition, Expert Syst. Appl. 136 (2019) 159–170, http://dx.doi.org/10.1016/j.eswa.2019.06.036.

[25] G.R. Gonçalves, S.P.G. da Silva, D. Menotti, W.R. Schwartz, Benchmark for license plate character segmentation, JEI-J. Electron. Ind. 25 (5) (2016) 053034, http://dx.doi.org/10.1117/1.JEI.25.5.053034.

[26] R. Laroca, E.V. Cardoso, D.R. Lucio, V. Estevam, D. Menotti, On the cross-dataset generalization in license plate recognition, in: Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2022, pp. 166–178, http://dx.doi.org/10.5220/0010846800003124, arXiv:2201.00267.

[27] Y. Yuan, W. Zou, Y. Zhao, X. Wang, X. Hu, N. Komodakis, A robust and efficient approach to license plate detection, IEEE Trans. Image Process. 26 (3) (2017) 1102–1114, http://dx.doi.org/10.1109/TIP.2016.2631901.

[28] G.-S. Hsu, J.-C. Chen, Y.-Z. Chung, Application-oriented license plate recognition, IEEE Trans. Veh. Technol. 62 (2) (2013) 552–561, http://dx.doi.org/10.1109/TVT.2012.2226218.

[29] S.M. Silva, C.R. Jung, License plate detection and recognition in unconstrained scenarios, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 580–596, https://openaccess.thecvf.com/content_ECCV_2018/html/Sergio_Silva_License_Plate_Detection_ECCV_2018_paper.html.

[30] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, L. Huang, Towards end-to-end license plate detection and recognition: A large dataset and baseline, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 255–271, https://openaccess.thecvf.com/content_ECCV_2018/html/Zhenbo_Xu_Towards_End-to-End_License_ECCV_2018_paper.html.

[31] C. Henry, S.Y. Ahn, S.-W. Lee, Multinational license plate recognition using generalized character sequence detection, IEEE Access 8 (2020) 35185–35199, http://dx.doi.org/10.1109/ACCESS.2020.2974973.

[32] S. Naseer, S.M.A. Shah, S. Aziz, M.U. Khan, K. Iqtidar, Vehicle make and model recognition using deep transfer learning and support vector machines, in: 2020 IEEE 23rd International Multitopic Conference, INMIC, 2020, pp. 1–6, http://dx.doi.org/10.1109/INMIC50486.2020.9318063.

[33] Y. Zhou, L. Liu, L. Shao, Vehicle re-identification by deep hidden multi-view inference, IEEE Trans. Image Process. 27 (7) (2018) 3275–3287, http://dx.doi.org/10.1109/TIP.2018.2819820.

[34] H.J. Lee, I. Ullah, W. Wan, Y. Gao, Z. Fang, Real-time vehicle make and model recognition with the residual SqueezeNet architecture, Sensors 19 (5) (2019) 982, http://dx.doi.org/10.3390/s19050982.

[35] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, L.-Y. Duan, Group-sensitive triplet embedding for vehicle reidentification, IEEE Trans. Multimed. 20 (9) (2018) 2385–2399, http://dx.doi.org/10.1109/TMM.2018.2796240.

[36] Z. Zheng, T. Ruan, Y. Wei, Y. Yang, T. Mei, VehicleNet: Learning robust visual representation for vehicle re-identification, IEEE Trans. Multimed. 23 (2021) 2683–2693, http://dx.doi.org/10.1109/TMM.2020.3014488.

[37] A.M.N. Taufique, A. Savakis, LA&Net: Local graph aggregation network with class balanced loss for vehicle re-identification, Neurocomputing 463 (2021) 122–132, http://dx.doi.org/10.1016/j.neucom.2021.07.082.

[38] L. Yang, P. Luo, C. Change Loy, X. Tang, A large-scale car dataset for fine-grained categorization and verification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3973–3981, https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Yang_A_Large-Scale_Car_2015_CVPR_paper.html.

[39] H. Liu, Y. Tian, Y. Yang, L. Pang, T. Huang, Deep relative distance learning: Tell the difference between similar vehicles, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2167–2175, https://openaccess.thecvf.com/content_cvpr_2016/html/Liu_Deep_Relative_Distance_CVPR_2016_paper.html.

[40] X. Liu, W. Liu, T. Mei, H. Ma, A deep learning-based approach to progressive vehicle re-identification for urban surveillance, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, in: Lecture Notes in Computer Science, Springer International Publishing, Cham, 2016, pp. 869–884, http://dx.doi.org/10.1007/978-3-319-46475-6_53.

[41] Y. Lou, Y. Bai, J. Liu, S. Wang, L. Duan, VERI-Wild: A large dataset and a new method for vehicle re-identification in the wild, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3235–3243, https://openaccess.thecvf.com/content_CVPR_2019/html/Lou_VERI-Wild_A_Large_Dataset_and_a_New_Method_for_Vehicle_CVPR_2019_paper.html.

[42] J. Sochor, A. Herout, J. Havel, BoxCars: 3D boxes as CNN input for improved fine-grained vehicle recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3006–3015, https://openaccess.thecvf.com/content_cvpr_2016/html/Sochor_BoxCars_3D_Boxes_CVPR_2016_paper.html.

[43] J. Sochor, J. Š.paňhel, A. Herout, BoxCars: Improving fine-grained recognition of vehicles using 3-D bounding boxes in traffic surveillance, IEEE Trans. Intell. Transp. Syst. 20 (1) (2019) 97–108, http://dx.doi.org/10.1109/TITS.2018.2799228.

[44] X. Li, M. Yuan, Q. Jiang, G. Li, VRID-1: A basic vehicle re-identification dataset for similar vehicles, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems, ITSC, 2017, pp. 1–8, http://dx.doi.org/10.1109/ITSC.2017.8317817.

[45] K. Yan, Y. Tian, Y. Wang, W. Zeng, T. Huang, Exploiting multi-grain ranking constraints for precisely searching visually-similar vehicles, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 562–570, https://openaccess.thecvf.com/content_iccv_2017/html/Yan_Exploiting_Multi-Grain_Ranking_ICCV_2017_paper.html.

[46] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, J.-N. Hwang, CityFlow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 8797–8806, https://openaccess.thecvf.com/content_CVPR_2019/html/Tang_CityFlow_A_City-Scale_Benchmark_for_Multi-Target_Multi-Camera_Vehicle_Tracking_and_CVPR_2019_paper.html.

[47] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3D object representations for fine-grained categorization, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2013, pp. 554–561, https://www.cv-foundation.org/openaccess/content_iccv_workshops_2013/W19/html/Krause_3D_Object_Representations_2013_ICCV_paper.html.

[48] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, TaoXie, J. Fang, imyhxy, Lorna, Z. Yifu, C. Wong, V. Abhiram, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, V. Sonck, tkianai, yxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, M. Jain, Ultralytics/YoloV5: V7.0 - YOLOv5 SOTA Realtime Instance Segmentation, 2022, http://dx.doi.org/10.5281/zenodo.7347926, Zenodo.

[49] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, YOLOv4: Optimal speed and accuracy of object detection, 2020, http://dx.doi.org/10.48550/arXiv.2004.10934, arXiv:2004.10934.

[50] J. Redmon, A. Farhadi, YOLOv3: An incremental improvement, 2018, http://dx.doi.org/10.48550/arXiv.1804.02767, arXiv:1804.02767.

[51] M. Tan, R. Pang, Q.V. Le, EfficientDet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10781–10790, https://openaccess.thecvf.com/content_CVPR_2020/html/Tan_EfficientDet_Scalable_and_Efficient_Object_Detection_CVPR_2020_paper.html.

[52] S. Zhang, C. Chi, Y. Yao, Z. Lei, S.Z. Li, Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9759–9768, https://openaccess.thecvf.com/content_CVPR_2020/html/Zhang_Bridging_the_Gap_Between_Anchor-Based_and_Anchor-Free_Detection_via_Adaptive_CVPR_2020_paper.html.

[53] S. Liu, D. Huang, Y. Wang, Learning spatial fusion for single-shot object detection, 2019, http://dx.doi.org/10.48550/arXiv.1911.09516, arXiv:1911.09516.

[54] Y. Lee, J. Park, CenterMask: Real-time anchor-free instance segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13906–13915, https://openaccess.thecvf.com/content_CVPR_2020/html/Lee_CenterMask_Real-Time_Anchor-Free_Instance_Segmentation_CVPR_2020_paper.html.

[55] K. Wada, Labelme: Image polygonal annotation with Python, 2023, http://dx.doi.org/10.5281/zenodo.5711226.

[56] M. Jaderberg, K. Simonyan, A. Zisserman, K. Kavukcuoglu, Spatial transformer networks, in: Advances in Neural Information Processing Systems, vol. 28, Curran Associates, Inc., 2015, pp. 2017–2025, https://proceedings.neurips.cc/paper/2015/hash/33ceb07bf4eeb3da587e268d663aba1a-Abstract.html.

[57] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (3) (1951) 400–407, http://dx.doi.org/10.1214/aoms/1177729586.

[58] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, 2019, http://dx.doi.org/10.48550/arXiv.1711.05101, arXiv:1711.05101.

[59] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, http://dx.doi.org/10.48550/arXiv.1412.6980, arXiv:1412.6980.

[60] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, T. Mei, FastReID: A Pytorch toolbox for general instance re-identification, 2020, https://dx.doi.org/10.48550/arXiv.2006.02631, arXiv:2006.02631.

[61] M. Tan, Q. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019, pp. 6105–6114, https://proceedings.mlr.press/v97/tan19a.html.

[62] X. Liu, W. Liu, H. Ma, H. Fu, Large-scale vehicle re-identification in urban surveillance videos, in: 2016 IEEE International Conference on Multimedia and Expo, ICME, 2016, pp. 1–6, http://dx.doi.org/10.1109/ICME.2016.7553002.

Álvaro Ramajo-Ballester is a researcher and PhD candidate in the field of deep learning applied to 3D environment perception for autonomous vehicles using LiDAR and images at Universidad Carlos III de Madrid. His previous works include real-time license-plate and visual vehicle identification systems.