License plate detection for road littering

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Aditya Partap Singh (201108)

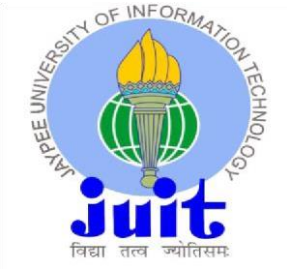
Shrest Agarwal (201110)

Divyan Rajveer Singh Rana (201555)

Under the guidance & supervision of

Prof. Dr. Vivek Kumar Sehgal

Professor and Head, Fellow IEI, SM-IEEE, SM-ACM (SG)



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

Candidate's Declaration

We hereby declare that the work presented in this report entitled **'License plate detection for road littering'** in partial fulfillment ofthe requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology,Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Prof. Dr. Vivek Kumar Sehgal**(Professor and Head, Fellow IEI, SM-IEEE, SM-ACM (SG)).The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Shrest Agarwal
Roll No.:201110

(Student Signature with Date)
Student Name: Aditya Partap Singh
Roll No.: 201108

(Student Signature with Date)
Student Name: Divyan Rajveer Singh Rana
Roll No.: 201555

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Vivek Sehgal

(Supervisor Signature with Date)
Supervisor Name: Prof. Dr. Vivek Kumar Sehgal
Designation: Professor and Head, Fellow IEI, SM-IEEE, SM-ACM (SG)
Department: CSE
Dated:

Candidate's Declaration

We hereby declare that the work presented in this report entitled **'License plate detection for road littering'** in partial fulfillment ofthe requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology,Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Prof. Dr. Vivek Kumar Sehgal**(Professor and Head, Fellow IEI, SM-IEEE, SM-ACM (SG)).The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Shrest Agarwal
Roll No.:201110

(Student Signature with Date)
Student Name: Aditya Partap Singh
Roll No.: 201108

(Student Signature with Date)
Student Name: Divyan Rajveer Singh Rana
Roll No.: 201555

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Vivek Sehgal

(Supervisor Signature with Date)
Supervisor Name: Prof. Dr. Vivek Kumar Sehgal
Designation: Professor and Head, Fellow IEI, SM-IEEE, SM-ACM (SG)
Department: CSE
Dated:

Acknowledgement

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully. We are really grateful and wish to be profoundly indebted to Supervisor Prof Dr. Vivek Kumar Sehgal, Professor and Head, Fellow IEI, SM-IEEE, SM-ACM (SG)Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of our supervisor in the field of "Artificial Intelligence /Information Security" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project. We would like to express our heartiest gratitude to Prof Dr. Vivek Kumar Sehgal Department of CSE, for his kind help to finish our project. We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking. Finally, we must acknowledge with due respect the constant support and patients of our parents.

Aditya Partap Singh
(201108)

Shrest Agarwal
(201110)

Divyan Rajveer Singh Rana
(201555)

Table of Content

List of Figures

List Of Abbreviations

1. YOLO: You Only Look Once
2. CNN: Convolutional Neural Network
3. R-CNN: Region-based Convolutional Neural Network
4. SVM: Support Vector Machine
5. ELA: Error Level Analysis
6. COCO: Common Objects in Context
7. CUDA: Compute Unified Device Architecture
8. GPU: Graphics Processing Unit
9. CPU: Central Processing Unit
10. SSD: Single Shot Detector
11. ReLU: Rectified Linear Unit
12. NMS: Non-Maximum Suppression
13. IOU: Intersection Over Union

Abstract

Compared to conventional transportation vehicles, construction vehicles have various driving circumstances and environmental factors. They consequently encounter particular difficulties when working at construction and evacuation sites. Therefore, even if the learning strategy for construction trucks is the same as that for conventional transportation vehicles like cars, study must be done to overcome these issues while implementing autonomous driving. The aim of this thesis study has been identified as being fulfilled by the following objectives. to determine appropriate and very effective CNN models for tracking and real-time object identification of construction vehicles. Analyse these CNN models' categorization performance. Present the results after comparing them to one another. The experiment and the literature review are appropriate research methods to use in answering the research questions. A survey of the literature has been done to determine the best object detection models for real-time object tracking and recognition. Since then, tests have been conducted to evaluate the performance of the selected object detection models. The literature research has determined that the most appropriate and effective algorithms for real-time detection and tracking of scaled construction trucks are the Faster R-CNN model, YOLO, and Tiny-YOLO. These algorithms' classification performance has been computed and contrasted with one another. The findings have been made public. Among the algorithms, YOLO has been determined to have a greater F1 score and accuracy, followed by Faster R-CNN. Consequently, the optimal algorithm for the real-time recognition and tracking of scaled construction vehicles has been determined to be YOLO. The outcomes resemble the literature's comparison of these three algorithms' classification performance.

Chapter 1: Introduction

1.1 Introduction

The process of recognizing objects in videos and images is known as Object recognition. This computer vision technique enables the autonomous vehicles to classify and detect objects in real-time. An autonomous vehicle is an automobile that has the ability to sense and react to its environment so as to navigate without the help or involvement of a human. The object detection and recognition are considered to be one of the most important tasks as this is what helps the vehicle detect obstacles and set the future courses of the vehicle. Therefore, it is necessary for the object detection algorithms to be highly accurate. Selecting the appropriate algorithm for autonomous driving is crucial since it necessitates real-time object detection and recognition, even though there are numerous machine learning and deep learning methods available for object detection and recognition, such as Support Vector Machine (SVM), Convolutional Neural Networks (CNNs), Regional Convolutional Neural Networks (R-CNNs), You Only Look Once (YOLO) model, etc. In order for the vehicle controllers to solve optimisation problems at least once every second, it is imperative that the algorithms be quick, accurate, and able to detect things in an image in real-time, as robots are not as quick as humans to do so.

Littering is a pervasive issue that raises serious issues with the environment, public health, and welfare of local communities. Innovative strategies that not only enforce anti-littering legislation but also increase awareness and encourage responsible conduct are needed to address this issue. We want to automate the process of detecting and holding litterers accountable for their actions by leveraging the power of Python, YOLOv8, and EasyOCR. This project acts as a deterrence to future littering behaviour in addition to offering a more effective method of punishment.

1.2 Problem Statement

Litter buildup in public areas and along roadsides is a widespread problem that poses health, aesthetic, and environmental risks worldwide. Littering is still a major issue even with efforts to reduce it through waste management programmes and public awareness campaigns. Conventional approaches to detecting and cleaning up litter are frequently costly, time-consuming, and labour-intensive. In order to solve the ongoing problems with object detection, this project makes use of You Only Look Once (YOLO), a creative and effective neural network design.

Finding the right balance between speed and accuracy is one of the main problems with object detection. Numerous current solutions sacrifice one for the other, making them less useful in situations when processing in real time is crucial. The research acknowledges this trade-off and looks to YOLO, which is renowned for processing whole images in a single forward pass, as a solution that can successfully balance speed and accuracy. This article focuses on using Python to implement YOLOv8, the most recent version of YOLO that offers improvements in terms of accuracy and computing efficiency.

We suggest a project aimed at creating an automated system for trash detection from moving cars utilising machine learning methods in order to overcome this difficulty. To enable timely and focused cleanup activities, the main goal is to develop a system that can reliably recognise and categorise various kinds of trash items in real-time.

The requirement for solid training datasets that accurately reflect the variety of real-world situations represents another major obstacle. This study is to investigate techniques for efficiently training and fine-tuning YOLOv8 utilising pertinent datasets, acknowledging the significance of both dataset quality and diversity. By doing this, the research hopes to improve the generalisation capabilities of the model, guaranteeing accurate and dependable object recognition in a variety of application cases and domains.

1.3 Objectives

- **Data collection:**

Assemble a large, annotated collection of pictures that show different kinds of trash, a range of environmental circumstances, and varying car speeds. The base for training and assessing machine learning models will be provided by this dataset.

- **Annotating the data:**

After acquiring the images, we annotate the data with accurate labels by manually drawing bounding boxes around the objects in every image in our training dataset.
This facilitates the training and evaluation of our model.

- **Real-time license plate detection:**

Implement real-time license plate detection to enhance the process of digital media content authentication and verification. This includes ensuring the authenticity and reliability of images and videos, particularly in situations where authenticity is crucial.

- **EasyOCR license plate recognition:**

Implement EasyOCR to accurately recognize and extract alphanumeric characters.

- **Litter Detection:**

Develop a model to accurately detect and classify litter.

1.4 Significance and Motivation of the Project Work

An important first step in addressing the pervasive issue of environmental damage brought on by littering is my endeavour to identify and log the license plates of cars that leave trash behind. Littering is more than just an aesthetic problem; it endangers the environment, public health, and welfare of communities. By automating the identification and enforcement process, my project provides a possible answer to this problem by utilizing cutting-edge technologies like Python, YOLOv8, and EasyOCR.

Conventional approaches to policing anti-littering legislation have frequently proven resource-intensive and ineffectual. Nonetheless, my project offers a more effective and possibly more efficient way to execute the law. It keeps people responsible for their actions and discourages future littering behaviour by automatically recording the license plate information of cars that are detected littering. Deterrence plays a critical role in cultivating a culture of accountability and environmental respect.

Through its detection and recording procedure, it highlights the negative effects of littering and raises community awareness of the need to maintain clean, pollution-free public areas. This teaching component is crucial for encouraging individuals to adopt new behaviours over the long run and to feel stewards of the environment.

Furthermore, this project can support current environmental programs and laws that try to lessen trash. It can help with focused interventions and initiatives to reduce littering at the local, regional, and national levels by offering useful data on hotspots and trends associated with littering. In this sense, my idea solves the immediate problem of trash while simultaneously making a larger contribution to the cause of healthier and more sustainable societies.

Chapter 2: Literature Survey

2.1 Overview of Relevant Literature

[1] YOLO9000: Better, Faster, Stronger, Joseph Redmon, Ali Farhadi, University of Washington, Allen Institute for AI (2017)

This Paper presents YOLO9000, a cutting-edge, real-time object identification system capable of detecting over 9000 object categories. First, we suggest several enhancements to the YOLO detection approach, both unique and based on previous work. The updated model, YOLOv2, performs admirably on standard detection tasks such as PASCAL VOC and COCO. The same YOLOv2 model may operate at multiple sizes thanks to a revolutionary, multi-scale training strategy, providing an easy compromise between speed and accuracy.

Tools/Technologies:

The YOLOv2 research report will most likely discuss enhancements and alterations made to the original YOLO framework. In this context, common tools and technologies may include:
1.Framework for the Darknet:The Darknet framework, created by Joseph Redmon, is frequently used to achieve YOLO.
2.CNNs (Convolutional Neural Networks): YOLOv2, like its predecessor, is expected to leverage CNNs for effective feature extraction and object detection.
3.CUDA: GPU acceleration through CUDA is commonly employed to enhance the speed of neural network computations.
4.Datasets: The model is likely trained and evaluated on datasets such as COCO (Common Objects in Context) for benchmarking and comparison.

Result:

The real-time detection systems YOLOv2 and YOLO9000 are presented in this paper. YOLOv2 is state-of-the-art and works better than other detection techniques on a variety of detection datasets. Additionally, it can be used with varying image sizes to provide a smooth trade-off between accuracy and speed.

YOLO9000 is a real-time framework that simultaneously optimizes detection and classification to detect over 9000 item categories. We combine our combined optimisation strategy to train on both ImageNet and COCO simultaneously and use WordTree to aggregate data from multiple sources. A major step forward in closing the dataset size difference between detection and classification is the YOLO9000.

[2] YOLOv3: An Incremental Improvement Joseph Redmon Ali Farhadi University of Washington (2018)

This paper has improved YOLO with a slew of minor design tweaks. In this paper they trained this fantastic new network. It's a little larger this time, but it's more accurate. It's still fast, don't worry. At 320X320, YOLOv3 runs in 22 ms at 28.2 mAP, which is as precise as SSD but three times faster. When compared to the old.5 IOU mAP detection metric, YOLOv3 is quite good. It reaches 57.9 AP50 in 51 ms on a Titan X, compared to RetinaNet's 57.5 AP50 in 198 ms, which is equivalent performance but 3.8 faster.

Tools/Technologies:

The tools and technologies utilised in YOLOv3 are anticipated to have comparable aspects as its predecessors, with some potential enhancements or adjustments. Common components may include:

1.Darknet Framework: YOLO models are typically implemented using the Darknet framework, which was developed by Joseph Redmon. Darknet is a neural network framework written in C and CUDA, and it is specifically designed for YOLO.

2.Convolutional Neural Networks (CNNs): YOLOv3, like its predecessors, heavily relies on convolutional neural networks for feature extraction and object detection tasks.

3.CUDA: GPU acceleration through CUDA is commonly used to speed up the training and inference processes of deep neural networks, including YOLO models.
4.Python and associated libraries: While the paper may not explicitly mention programming languages, it's common to use Python along with deep learning libraries such as TensorFlow or PyTorch for research and implementation.
5.Datasets: YOLOv3 is likely trained and evaluated on benchmark datasets such as COCO (Common Objects in Context) or others relevant to object detection tasks.
6.pre-trained models: Transfer learning may be employed, using pre-trained models on large datasets to initialize the weights of the YOLOv3 network before fine-tuning on specific object detection datasets.

Results:

The improvements made by YOLOv3 over its predecessors would be described in detail in the paper's results section. Metrics like speed, accuracy, and efficiency of object detection could be included. In order to highlight the model's advantages and disadvantages, it might also discuss how the model performed on particular issues or datasets.

[3] YOLOv4: Optimal Speed and Accuracy of Object Detection Alexey Bochkovskiy, Chien-Yao Wang∗ Institute of Information Science Academia Sinica, Taiwan Hong-Yuan Mark Liao, Institute of Information Science Academia Sinica, Taiwan(2020)

The goal of this study is to improve Convolutional Neural Networks' (CNNs') object detection capability in real time and accuracy. The study investigates a wide range of characteristics intended to increase CNN accuracy, taking into account both general features that can be applied to various models and tasks and specialised features that are specific to particular models and circumstances. Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalisation (CmBN), Self-adversarial-training (SAT), and Mish-activation are some of the prominent universal features.

Several unique features, including DropBlock regularisation, CIoU loss, and Mosaic data augmentation, are introduced and combined in this research. The objective is to develop a cutting-edge object identification model that may be used in a variety of applications by running in real-time on traditional GPUs. With a real-time speed of about 65 frames per second on a Tesla V100 GPU, the suggested model, dubbed YOLOv4, produces impressive results, displaying a 43.5% Average Precision (AP) and 65.7% AP50 for the MS COCO dataset.

In order to train a quick and precise object detector, users with ordinary GPUs, such the 1080 Ti or 2080 Ti, can use the developed model, which is highlighted in the article as having practical relevance. In order to make state-of-the-art procedures more effective and appropriate for single-GPU training, the authors tweak and optimise them. They also explore the impact of novel Bag-of-Freebies and Bag-of-Specials methods during detector training.

Among the contributions of this work are the creation of an effective object detection model that can be applied widely, an investigation into the effects of cutting-edge object detection techniques, and adjustments to improve the effectiveness of currently available single-GPU training methods. The goal of the research is to advance object detection capabilities across multiple domains by offering a robust and useful real-time object detection solution. The suggested model's source code is made accessible to the general public for additional research and application.

Tools and Technologies:

Tools and Technologies Used: YOLOv4's tools and technologies are anticipated to incorporate elements comparable to its predecessors, with potential enhancements or adjustments. The following are examples of common components:

1.Framework for the Darknet: The Darknet framework is frequently used to implement YOLO models, particularly YOLOv4. Darknet is a neural network framework written in C and CUDA that was created exclusively for YOLO. It enables GPU acceleration via CUDA, which contributes to the model's ideal performance and efficiency.

2.CNNs (Convolutional Neural Networks): As a deep learning model, YOLOv4 is built on the foundations of convolutional neural networks. CNNs play an important role in picture feature extraction and object detection applications.

3.CUDA: GPU acceleration using CUDA is widely used to speed up neural network computations. Parallel processing on GPUs greatly benefits YOLO models, notably YOLOv4.

4.Python and its dependencies: Python is commonly used for research and implementation, along with deep learning packages such as TensorFlow or PyTorch. Although these technologies are not specifically mentioned in the study, they are commonly utilised in the deep learning community.

5Datasets: YOLOv4 was most likely trained and assessed using object detection benchmark datasets such as COCO (Common Objects in Context) or others.

6.Models that have already been trained: Transfer learning can be used to initialise the weights of the YOLOv4 network using pre-trained models on broad datasets before fine-tuning on individual item detection datasets.

Result:

For inference time verification, we run YOLOv4 on widely used GPUs of the Maxwell, Pascal, and Volta systems and compare it with other state-of-the-art techniques because different approaches use different GPU architectures.

[4] YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors Chien-Yao Wang1, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, Institute of Information Science, Academia Sinica, Taiwan(2022)

YOLOv7, an advanced object detection model, is presented in this paper. It outperforms all known real-time object detectors across a wide range of frames per second (FPS) and excels in both speed and accuracy. With an Average Precision (AP) of 56.8%, YOLOv7 outperforms other real-time object detectors running at 30 frames per second or more on the GPU V100

in terms of accuracy. Significantly, in terms of speed and accuracy, the YOLOv7-E6 variant outperforms transformer-based detectors like SWIN-L Cascade-Mask R-CNN and convolutional-based detectors like ConvNeXt-XL Cascade-Mask R-CNN.

The goal of the research is to enable real-time object detection across a wide range of computing environments and devices, from mobile GPUs to GPUs in edge and cloud environments. In contrast to mainstream approaches that focus mainly on optimising architecture, YOLOv7 emphasises training process optimization without raising inference costs by introducing trainable bag-of-freebies methods.

In this paper, new problems in dynamic label assignment and model re-parameterization are discussed and useful solutions are suggested. In particular, it presents a coarse-to-fine lead-guided label assignment technique and a planned re-parameterized model to improve training efficiency for models with several output layers.

The development of trainable bag-of-freebies techniques, the recognition and handling of novel obstacles in the evolution of object detection, the introduction of "extend" and "compound scaling" strategies for effective parameter and computation utilisation, and the accomplishment of a 40% parameter reduction and a 50% computation reduction for cutting-edge real-time object detectors are among the contributions.

With its increased detection accuracy and faster inference speed, YOLOv7 represents a major breakthrough in real-time object detection. The source code is released for public use in order to facilitate additional research and development.

Tools and Technologies:

The specific tools and technologies utilised in YOLOv7 would be mentioned in the paper or supporting documentation. However, given the historical background of the YOLO series, popular tools and technology might include:

1.Deep Learning Architecture: Popular deep learning frameworks such as PyTorch or TensorFlow are frequently used to create YOLO models.
2.CUDA: CUDA GPU acceleration for easier neural system inference and training.
3.Datasets: Standard object detection datasets are likely to be used in training and evaluation, and the publication may detail the specific datasets used.

Result:

In this study, we present a novel architecture and model scaling approach for real-time object detection. Moreover, we find that new research areas are generated by the evolution of object identification algorithms. We found the dynamic label assignment allocation problem and the re-parameterized module replacement problem during the research phase. We propose the trainable bag-of-freebies method to improve the accuracy of item detection in order to tackle the problem. We developed the YOLOv7 family of object detection systems, which yield state-of-the-art outcomes, based on the aforementioned.

[5]DynamicDet: A Unified Dynamic Architecture for Object Detection Zhihao Lin Yongtao Wang, Jinhe Zhang Xiaojie Chu Wangxuan, Institute of Computer Technology, Peking University(2020)

In this paper, we introduce DynamicDet, a dynamic framework for object detection that uses adaptive inference to achieve an impressive trade-off between computational efficiency and accuracy. By presenting a novel dynamic architecture, an adaptive router for multi-scale analysis, an optimization strategy based on detection losses, and a variable-speed inference strategy, DynamicDet tackles the difficulties in creating potent dynamic detectors.

Cascaded detectors and a router make up DynamicDet's dynamic architecture, which gives users the option to exit inference with multi-scale data. In order to facilitate dynamic decision making, an adaptive router is introduced that automatically evaluates the difficulty scores of images based on multi-scale features. Additionally, variable-speed inference techniques and hyperparameter-free optimization strategies designed for the suggested dynamic architecture are presented in the paper.

DynamicDet is effective in achieving new state-of-the-art accuracy-speed trade-offs for real-time object detection, as demonstrated by extensive experiments on the COCO benchmark. In terms of inference speed, for example, the DynamicDet variant Dy-YOLOv7-W6 outperforms similar models such as YOLOv7-E6 by 12%, YOLOv7-D6 by 17%, and YOLOv7-E6E by 39% while retaining similar accuracy. With its single dynamic detector that can adjust to a variety of application scenarios, the suggested framework offers notable improvements in real-time object detection.

Tools and Technologies:

1.Deep Learning Architecture: Popular deep learning frameworks, such as PyTorch or TensorFlow, are frequently used to create YOLO models, notably YOLOv5. PyTorch has been used by Ultralytics, the organisation connected with YOLOv5.

2.CUDA: GPU acceleration with CUDA is commonly used to accelerate deep neural network training and inference procedures. Parallel processing on GPUs is often advantageous for YOLO models.

3.Python and its dependencies: Python is a popular programming language for deep learning research. For general-purpose tasks, researchers frequently use libraries such as NumPy, SciPy, and scikit-learn, as well as deep learning libraries.

4.Datasets: YOLOv5 was trained and tested on COCO (Common Objects in Context) or other datasets relevant to object detection, segmentation, and instance segmentation tasks.

5.Models that have already been trained: Transfer learning can be used to initialise the weights of the YOLOv5 network before fine-tuning on specific tasks by employing pre-trained models on huge datasets.

Result:

In this study, we present a novel architecture and model scaling approach for real-time object detection. Moreover, we find that new research areas are generated by the evolution of object identification algorithms.

We found the dynamic label assignment allocation problem and the re-parameterized module replacement problem during the research phase. We propose the trainable bag-of-freebies method to improve the accuracy of item detection in order to tackle the problem. We developed the YOLOv7 family of object detection systems, which yield state-of-the-art outcomes, based on the aforementioned.

[6] YOLOX: Exceeding YOLO Series in 2021 Zheng Ge, Songtao Liu, Feng Wang Zeming Li Jian Sun Megvii Technology(2021)

This paper presents YOLOX, a high-performance object detector that makes inventive enhancements to the YOLO series. YOLOX uses a leading label assignment strategy called SimOTA in conjunction with sophisticated detection techniques like a decoupled head. The model surpasses previous benchmarks on the COCO dataset and achieves state-of-the-art results across a wide range of models.

The YOLOX versions perform exceptionally well. YOLOX-Nano outperforms NanoDet by 1.8% AP on COCO, achieving 25.3% AP with just 0.91M parameters and 1.08G FLOPs. YOLOX outperforms the current best practice by 3.0% AP, increasing the widely used YOLOv3 to 47.3% AP on COCO. YOLOX-L outperforms YOLOv5-L by 1.8% AP and achieves 50.0% AP on COCO at 68.9 FPS on Tesla V100, with parameters similar to YOLOv4-CSP and YOLOv5-L. YOLOv4-Tiny and NanoDet are inferior to YOLOX-Tiny and YOLOX-Nano by 10% AP and 1.8% AP, respectively.

Using recent developments in anchor-free detectors, label assignment strategies, and end-to-end (NMS-free) detectors, the authors decide to begin their optimizations with YOLOv3. The skilled updates to YOLOv3 lead to notable gains in accuracy. With support for ONNX, TensorRT, NCNN, and Openvino, the YOLOX code is released.

In the report's conclusion, they describe how they used a single YOLOX-L model to win first place in the Streaming Perception Challenge (Workshop on Autonomous Driving at CVPR 2021). The advancements and insights presented here are intended to further the field of real-time object detection by providing developers and researchers with useful experiences in real-world scenarios.

Tools and Technologies:

1.Deep Learning Architecture: YOLOX, like other YOLO variations, is most likely built with a well-known deep learning architecture known as CSP Darknet53 with an SSP Layers as a backbone to extract high dimensional features from input image. Commonly used libraries include PyTorch and TensorFlow. Details on the exact framework used can be found in the official repository or in the article.

2.CUDA: Because YOLO models are computationally costly, GPU acceleration via CUDA is widely utilised to accelerate neural network computations. This might also be used by YOLOX for training and inference.

3.Python and its dependencies: Python is a popular programming language for deep learning research. For implementation, researchers frequently employ NumPy, SciPy, and other machine learning-specific libraries.

4.Datasets: YOLOX may have been trained and tested using benchmark datasets like as COCO (Common Objects in Context) or other object detection datasets. The dataset used can have an impact on the model's performance.

5.Models that have already been trained: Deep learning frequently employs transfer learning. Before fine-tuning on specific tasks, YOLOX may use pre-trained models on huge datasets for initialization.

Results:

The YOLOX high-performance anchor free detector is comprised of several experienced upgrades to the YOLO series, which we present in this study. Thanks to its advanced label assignment strategy, anchor-free detection, decoupled head, and other cutting-edge

sophisticated detection techniques, YOLOX outperforms its competitors in terms of speed and accuracy across all model sizes. We are proud to report that we have achieved an impressive 47.3% AP on COCO, outperforming the current best practice by 3.0% AP with our improved design of YOLOv3, which is still one of the most widely used detectors in the industry due to its broad compatibility. We anticipate that this study will help researchers and developers obtain real-world experience.

[7] PP-YOLO: An Effective and Efficient Implementation of Object Detector Xiang Long, Kaipeng Deng, Guanzhong Wang, Yang Zhang, Qingqing Dang, Yuan Gao, Hui Shen, Jianguo Ren, Shumin Han, Errui Ding, Shilei Wen (2021)

This paper tackles the problem of finding a fair trade-off between efficiency and effectiveness in object detection while taking into account real-world hardware constraints. Rather than suggesting a new detection model, the emphasis is on putting into practice a balanced-performance object detector called PP-YOLO, which is based on the popular YOLOv3 architecture. The objective is to improve accuracy with nearly unchanged inference speed by utilizing different existing techniques that only slightly increase FLOPs and model parameters.

The balanced effectiveness (45.2% mAP) and efficiency (72.9 FPS) of PP-YOLO outperform current state-of-the-art detectors such as YOLOv4 and EfficientDet. The enhancements are achieved by carefully combining several techniques, giving developers a recipe to gradually improve their detectors. Two noteworthy approaches are applying MixUp for data augmentation and using ResNet as the foundation. In contrast to YOLOv4, this work does not investigate alternative backbone networks, techniques for augmenting data, or Neural Architecture Search (NAS) for hyperparameter search.

The practical application of tricks with little effect on efficiency is emphasized in the paper, which makes PP-YOLO a good option for everyday situations. The suggested model outperforms YOLOv4 in terms of accuracy and speed, showing an increase in mAP from

43.5% to 45.2%. The authors think that investigating alternative backbones, utilizing NAS for hyperparameter optimization, and applying data augmentation methods can all lead to even greater improvements.

Tools and Technologies:

1.Deep Learning Architecture: Like many other object identification models, PP-YOLO is most likely built with a popular deep learning framework called PaddlePaddle . Commonly used libraries include PyTorch and TensorFlow.

2.CUDA: CUDA-based GPU acceleration is often used in deep learning for training and inference to improve computation speed. This might is used by PP-YOLO for more efficient processing.

3.Python and its dependencies: Python is a popular programming language in the deep learning domain. For implementation, libraries such as NumPy, SciPy, and other machine learning-specific libraries may be utilised.

4.Datasets: PP-YOLO may have been trained and evaluated using benchmark datasets such as COCO (Common Objects in Context) or other object identification datasets. The dataset used can have an effect on the model's performance.

5.Models that have already been trained: Deep learning frequently employs transfer learning. Before fine-tuning on specific tasks, PP-YOLO may leverage pre-trained models on big datasets for initialization.

Result:

This work presents PP-YOLO, a novel PaddlePaddle-based object detector implementation. PPYOLO outperforms other state-of-the-art detectors such as YOLOv4 and EfficientDet in terms of accuracy (COCO mAP) and speed (FPS). In this paper, we examine several approaches and show their practical application by combining them on the YOLOv3 detector.

[8] Rethinking the Backbone Architecture for Tiny Object Detection Jinlai Ning, Haoyan Guan and Michael Spratling, Department of Informatics, King's College London, London, UK(2021)

The prevalence of images containing small targets in real-world scenarios makes tiny object detection a special challenge in computer vision. Standard deep neural network backbones meant for larger objects are frequently used in existing methods for tiny object detection, which produces less-than-ideal outcomes. In order to improve tiny object detection, this paper presents a novel solution by suggesting "bottom-heavy" versions of backbones that give processing higher-resolution features priority.

The main finding is that standard backbones that use large strides in early layers or max-pooling produce lower-resolution feature maps that do not adequately capture information needed to identify small objects. By delaying downsampling processes, the suggested changes free up resources for earlier layers' improved feature extraction without adding to the already heavy computational load. Using datasets like CIFAR100 and ImageNet32, the study also looks into the effects of pre-training these modified backbones on images of the right size.

Results from experiments on the WiderFace and TinyPerson datasets show that detectors using the suggested backbones perform better than the state-of-the-art techniques currently in use. The effectiveness of the suggested strategy is demonstrated by the gains that are made without adding more parameters overall. This work offers a new insight into the backbone architecture of tiny object detection, highlighting the significance of maintaining high-resolution data in early layers to improve detection accuracy overall. The results point to a paradigm change away from conventional backbones for enhanced performance in the difficult field of small object detection.

Tools and Technologies:

1.Deep Learning Architecture: YCNet is built with a popular deep learning framework like PyTorch or TensorFlow. Researchers frequently select a framework depending on their tastes and the needs of their studies.

2.CUDA: CUDA-based GPU acceleration is used by YCNet to improve processing efficiency.

3.Python and its dependencies: Python is used for implementation, libraries such as NumPy, SciPy, and other machine learning-specific libraries are also utilised.

4.Datasets: YCNet is trained and assessed using benchmark datasets like as COCO (Common Objects in Context) . The dataset used can have an effect on the model's performance.

Result:

To evaluate the proposed backbone, a number of designs that have previously been shown to provide state-of-the-art performance on TinyPerson and WiderFace were analyzed. Our suggestion was to replace the standard backbone with a bottom-heavy (BH) counterpart. The suggested backbones performed better than the originals in every instance.

[9]EfficientDet: Scalable and Efficient Object Detection Mingxing Tan Ruoming Pang Quoc V. Le Google Research, Brain Team (2020)

With an emphasis on object detection, the paper discusses the growing significance of model efficiency in computer vision. The authors thoroughly examine the design decisions made for neural network architecture and suggest important optimizations to increase efficiency without compromising accuracy. Taking into account the unequal contribution of features at different resolutions, they introduce a Weighted Bi-directional Feature Pyramid Network (BiFPN) for multi-scale feature fusion. Furthermore, a Compound Scaling Method is put forth that scales resolution, depth, and width consistently throughout the object detection architecture's constituent parts. This all-encompassing scaling strategy, which draws inspiration from EfficientNets, works well for enhancing accuracy and productivity.

The result of these advancements is a new family of object detectors called EfficientDet, which shows state-of-the-art performance under a range of resource constraints with fewer parameters and floating-point operations per second (FLOPs). With 77M parameters and 410B FLOPs, the EfficientDet-D7 model achieves an astounding 55.1% COCO AP, outperforming earlier detectors while utilizing fewer FLOPs and being much smaller. The suggested detectors demonstrate adaptability in practical applications where latency and model size are crucial, such as robotics and self-driving cars. In contrast to earlier detectors, the authors' approach is shown to be effective by providing code and comprehensive results.

Tools and Technologies:

1.Deep Learning Architecture: EfficientDet is most likely built with a popular deep learning framework like TensorFlow or PyTorch. These frameworks offer effective tools for constructing and training deep neural networks.
2.CUDA: CUDA-based GPU acceleration is often used in deep learning for training and inference to improve computation speed. This might be used by EfficientDet to speed up processing.
3.Python and its dependencies: Python is a popular programming language in the deep learning domain. For implementation, libraries such as NumPy, SciPy, and other machine learning-specific libraries may be utilised.
4.Datasets: EfficientDet may have been trained and assessed using benchmark datasets such as COCO (Common Objects in Context) or other object detection datasets. The dataset used can influence the model's performance.
5.Models that have already been trained: Deep learning frequently employs transfer learning. Before fine-tuning on specific tasks, EfficientDet may employ pre-trained models on big datasets for initialization.

Result:

In this paper, we thoroughly investigate network architectural design options for effective object detection, and we propose a customized compound scaling algorithm and a weighted bidirectional feature network to enhance accuracy and efficiency. We develop the EfficientDet detector family based on these optimizations, which consistently outperform the previous art in terms of efficiency and accuracy under a wide range of resource constraints. Compared to previous object identification and semantic segmentation models, our scaled EfficientDet in particular achieves state-of-the-art accuracy with a significantly smaller number of parameters and FLOPs.

[10] Focal Loss for Dense Object Detection Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar, Facebook AI Research (2017)

This study explores the difficulties encountered by one-stage object detectors, a class of detectors that employs a frequent, dense sampling of potential object locations, in reaching accuracy levels commensurate with those of their extensively used two-stage counterparts, most famously represented by R-CNN. The primary obstacle found is the significant foreground-background class imbalance that arises during dense detector training, where an excessive amount of easily classified negative examples impede efficient learning.

In order to address this issue, the paper presents a novel loss function known as Focal Loss. The main idea behind Focal Loss is that it addresses the inefficiency of traditional one-stage detectors by preventing a large number of easily classified negatives from controlling the learning process.

The suggested approach is used in the RetinaNet one-stage detector. Experiments show that the Focal Loss works as intended, allowing RetinaNet to outperform all current state-of-the-art two-stage detectors in terms of accuracy while matching the speed of earlier one-stage detectors.

The study disproves the widely held belief that two-stage detectors are intrinsically better and highlights the potential of one-stage detectors when fitted with creative ways to deal with the imbalance in classes.

The research contributes significantly to the field by shedding light on the critical role of class imbalance during training and proposing a novel loss function that proves effective in overcoming this obstacle. The findings offer valuable insights into enhancing the efficiency and accuracy of object detection systems, opening avenues for further exploration and innovation in the evolving landscape of computer vision.

Tools and Technologies:

1.ResNet-50-FPN and ResNet-101-FPN
2.Feature Pyramid Network (FPN)
3.Region Proposal Network (RPN)
4.Selective Search
5.EdgeBoxes
6.DeepMask

Result:

The introduction of the focused loss, which, by down-weighting the loss assigned to well-classified cases, lessens the impact of negatives that are simple to classify during training. This improves performance in challenging cases and enables the model to concentrate more on challenging samples.

One-stage object detector:

RetinaNet predicts bounding boxes and class probabilities without the need for a second region proposal network (RPN) since it is a one-stage object detection model. The training and architecture processes are streamlined as a result.

Anchor boxes:

Anchor boxes with different scales and aspect ratios are used to accommodate variations in item sizes and shapes. The contribution of anchor boxes to the model's adaptability in identifying objects of different sizes is investigated in this paper.

[11] DynamicDet: A Unified Dynamic Architecture for Object Detection Zhihao Lin, Yongtao Wang, Jinhe Zhang, Xiaojie Chu, Wangxuan Institute of Computer Technology, Peking University

YOLOv5 is an object detection, picture semantic segmentation, and instance segmentation unified architecture. It is based on YOLOv4, but with a number of enhancements that make it faster, more accurate, and more adaptable. YOLOv5 is the first object detector to attain cutting-edge performance on all three objectives.

Tools and Technologies:

1.YOLOv5 makes use of a number of tools and technologies, including:

2.PyTorch is a Python deep learning library.

3.C++ is a performance-critical programming language.

4.CUDA is a GPU-based parallel computing platform.

5.OpenCV is a computer vision library.

Results:

YOLOv5 achieves cutting-edge scores on a wide range of benchmarks, including:

1.Detection of COCO objects: 58.0 mAP

2.50.0 AP COCO instance segmentation

3.Semantic segmentation of cityscapes: 74.9 mAP

2.2 Key Gaps in the Literature

Inadequate comparative evaluation between versions: The summaries that are being offered offer a comprehensive understanding of the development of YOLO models, particularly YOLOv2, YOLOv3, YOLOv4, and YOLOv7. Nevertheless, a noteworthy lack of a thorough comparison analysis that directly contrasts the advantages, disadvantages, and improvements of each edition is present. A detailed comparison could clarify the trade-offs between speed and accuracy and draw attention to the specific improvements made in later iterations. Comparing the development of YOLOv2 and YOLOv7 side by side would help to provide a more nuanced understanding of the model's evolution.

Discussion of Ethical Considerations and Bias Mitigation is Limited: The summaries mostly concentrate on technical features, instruments, and outcomes; however, they do not go into great detail about moral issues, bias reduction, or the effects of using these object identification algorithms on society. Since these models are widely applied in real-world settings, it would be beneficial to include a discussion of the possible biases in training data, the ethical implications of the technology, and the steps taken to address these problems.

Insufficient Examination of Transferability and Generalisation: Although the summaries mention the training and assessment datasets, there is a deficiency in investigating the extent to which these models generalise to various datasets and situations. Evaluating the real-world applicability of YOLO models requires an understanding of how well-suited they are to various domains, unknown data distributions, and difficult environmental circumstances. Beyond the benchmark datasets presented, a more extensive investigation of the models' performance in different scenarios would offer a more full picture of their efficacy.

Lack of Explicit Comparison Across Different Object Sizes: Although the summaries address the efficacy of the suggested models, there isn't any explicit comparison between different object sizes, particularly when it comes to microscopic object identification. For instance, the YCNet paper concentrates on enhancing the detection of small objects; yet, nothing is known about the effectiveness of the recommended alterations throughout a range of object sizes. To fill in a possible vacuum in the information provided, a comparative study across various object sizes would offer insightful information about the generalisation and adaptability of the suggested methodologies.

Inadequate Examination of Model Explain ability: The important topic of model explainability is not sufficiently explored in the summaries. It is crucial to comprehend how these object detection models make their predictions, particularly in situations where interpretability and transparency are crucial, like in the legal or medical domains. There is a lack of discussion regarding these models' interpretability and possible ramifications for their use in delicate fields.

Chapter 3: System Development

3.1 Requirements and Analysis

This section provides a thorough overview of the requirements and analysis phase, highlighting the fundamental elements that ensure the project is implemented successfully.

3.1.1 Functional Requirements

Object Detection: Using the YOLOv8 algorithm, the system ought to be able to identify objects in real time. A range of object classes pertinent to the application domain should be included in the detection.

Object Tracking: To follow the identified objects over a series of frames, implement object tracking functionality.

Processing in real time: Process video streams in real time and use GPU to provide minimal latency for tracking and detection.

Display Interface: To view the live camera feed, create a graphical user interface (GUI). Bounding boxes surrounding identified objects, object number, confidence score and their tracking trajectories are shown in the video feed.

Handle multiple objects at once: Should be able to track and display multiple objects in the camera frame at once.

3.1.2 Non Functional Requirements

Performance: Reach high performance levels that allows our hardware to handle real-time object tracking and detection.

Precision: Guarantee precise identification and monitoring outcomes, reducing the number of false positives and negatives.

Scalability: Build the system with the ability to support different camera resolutions and frame rates in mind.

Robustness: Develop a tracking system that is strong enough to withstand obstacles like abrupt changes in object motion, temporary disappearance, and object occlusion.
Compatibility: Ascertain compatibility with various video input sources and camera models.

3.1.3 Requirements for Data

Training Data: The YOLOv8 model can be trained on a comprehensively annotated dataset that includes all object classes pertinent to the given use case.
Video Input: A camera or a source of pre-recorded video can be used to provide video input to the system.
Tracking Data for Objects: Keep track of object IDs, positions, and timestamps in a database or other structured system and allow for downloading of result videos/images.
Configuration Files: To store and retrieve settings pertaining to confidence thresholds, YOLOv8 configuration, and object tracking parameters, we will be required to use configuration files.

Analysis

3.1.4 Methodology of Analysis

1. Justification for selection YOLO as the model used for object detection, focusing on how accurately it can detect the objects in the video.
2.Comparing the performance and accuracy of our custom dataset to other existing datasets like COCO.

3.1.5 Evaluation of Feasibility

1.Technical viability: YOLO is well-known and frequently utilized in object detection, object segmentation and object classification tasks.

2.Economic viability: Considering that YOLO is open-source and available to everyone and the hardware requirements are minimal, the project is financially feasible.

3.1.6 Examination of Risks

1.Identification of possible hazards, such as lack of data in custom dataset, lower number of epochs the model is trained on and computational resource limitations.

2.Creation of procedures for handling external dependencies, optimise code for effective real-time processing.

3.2 Project Design and Architecture

1) Collect and annotate images:

The dataset should contain sample images of the objects we want to detect. Each image in the dataset should be labelled with the bounding boxes of the objects present in it for training.

2) Choose an object detection model:

There are many different object detection models available, each with its own strengths and weaknesses. Some popular models include Faster R-CNN, YOLO, and SSD.

3) Crop number plate detected and pass it forward.

4) Implement OCR to detect license plate number.

5) Display the license plate number of the vehicle.

DFD Diagram

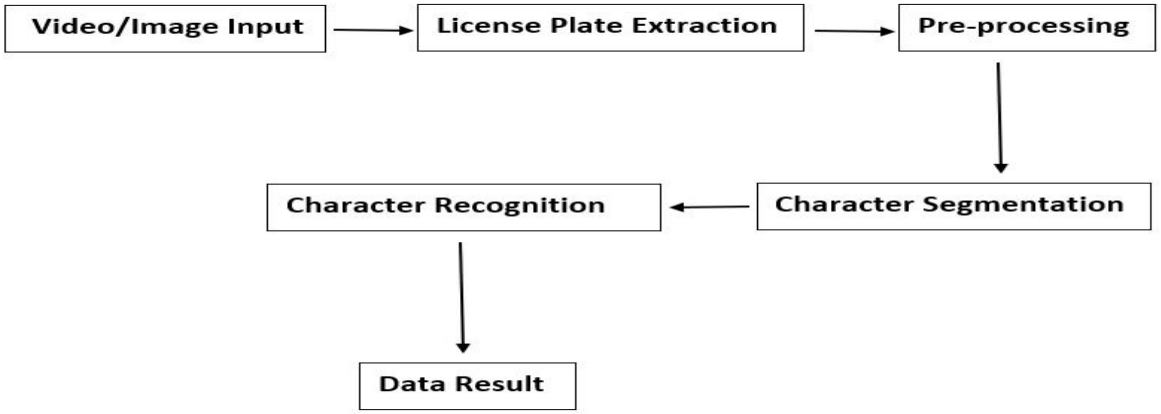


Fig. 3.1 DFD for license plate detection

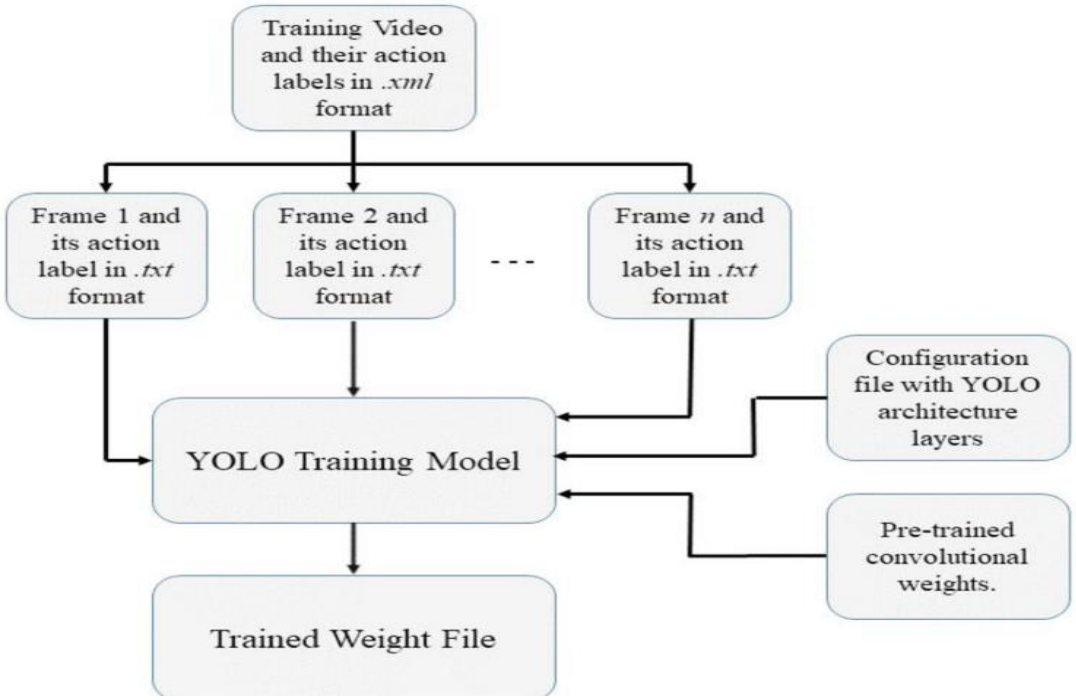


Fig. 3.2 DFD for Model training

3.3 Data Preparation

1. Data Collection:

We collect data in form of pre-recorded videos or live video feed from speed cameras in order to collect images of the vehicle and capture its number plate. To produce a heterogeneous dataset using a range of forgeries methods such as splicing, rotation, scaling, filtering, retouching, and recolouring.

2. Dataset Selection:

Then we take two significant datasets were selected for algorithm analysis and testing.
A popular database for identifying visual media forgeries is the CASIA Dataset.
includes pictures divided into eight groups, some of which have had parts altered by pasting, cutting, and other techniques, then we use our own custom dataset FIDAC Dataset: An original dataset of photos taken with our camera. With the help of applications, we used it to further alter images in order to simulate real-world scenarios.

3. Preprocessing Method:

We used Error Level Analysis (ELA) to implement the preprocessing method that divided pictures into the more manageable sections of chunks of images were individually recompressed using ELA with a predetermined 95% error rate.
In this we determined the exact difference between the re-compressed and analyzed images.

4. Suggested CNN Architecture:

Then we use Convolutional neural networks (CNN), were created with the purpose of detecting image forgeries. The 15 layers are configured with dropout, dense, convolution, pooling, and flattening operations.

We used Rectified Linear Unit (ReLU) was employed as the non-linearity activation function. To determine whether photos are authentic or manipulated, then Sigmoid activation function was applied to the final output layer.

5. Phase of Implementation

Firstly, we do Data Preprocessing to Enhanced forgery detection by applying ELA representations to the images datasets that are ready to be used with the suggested CNN architecture.
Then we perform ELA Implementation to the pipeline for detecting image forgeries to make sure that the possible tampering regions were properly highlighted by the ELA.

6. CNN Model Development

We used TensorFlow or a comparable deep learning framework to implement the planned CNN architecture. Model layers, activation functions, and loss functions are configured to defined an optimization strategy (e.g., stochastic gradient descent) and initialized the model parameters.

3.4 Implementation

3.4.1 Steps for implementing object detection using yolo:

1. Data collection: Collect a diverse set of images containing vehicles with visible license plates. Annotate the license plates in these images with bounding boxes to prepare the dataset for training.

2. Training the model:
Utilize YOLOv8 object detection architecture trained on the annotated dataset to detect license plates within the images.

3.License plate detection:

Implement the trained YOLOv8 model to perform real-time license plate detection on video frames. Utilize the detected bounding boxes to extract the region of interest (ROI) containing the license plate from each frame.

4.Optical character recognition:

Pass each cropped license plate image to EasyOCR, to extract the license plate number from the image. Ensure proper preprocessing steps, such as resizing, enhancing contrast, or noise reduction, to optimize OCR performance.

5. Displaying results in video:

Overlay the detected license plate number onto the corresponding vehicle in the video frame. Display the processed video with annotated license plate numbers in real-time or save it for further analysis.

6. Stratified Sampling: If the classes in our dataset are unbalanced, the train, validation, and test subsets all retain a proportionate representation of each class. By doing this, bias in model evaluation and training is lessened.

8. Establish a Directory Structure:

We sort our project into folders, keeping the YOLO model weights, configuration files, annotations, and images apart. It should follow the structure of directories required by YOLO and is simpler to manage files during training and testing in a well-organised directory.

9. Re-train the model:

We retrain our model on our own custom dataset that we have collected.

10. Test the model:

We use our re-trained model to detect objects and verify its accuracy.

11. License plate detection:

Implement license plate detection using the trained model.

12. Extract number plate:

Using EasyOCR get the license plate number of the vehicle.

3.4.2 Tools and Technologies:

- OpenCV
- Torch
- Google colab
- Jupyter notebook
- Numpy
- Matplotlib

3.4.3 Hardware Resources:

- Camera
- GPU
- CPU

3.4.4 Languages:

- Python

3.4.5 YOLO architecture:

After receiving an image as input, the YOLO algorithm employs a basic deep convolutional neural network to identify objects in the image.

The CNN model's architecture, which serves as the foundation for YOLO, is displayed below:

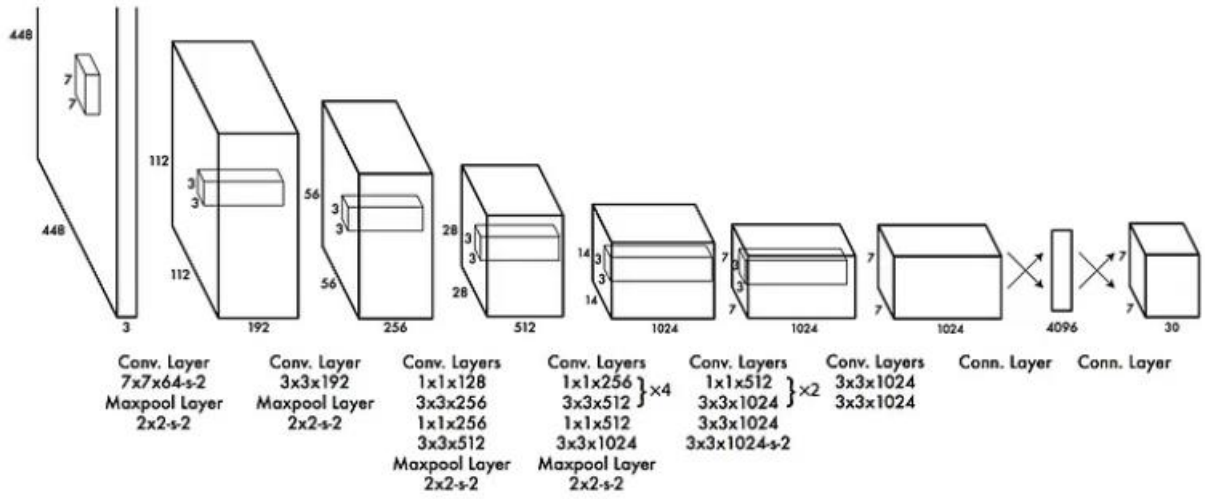


Fig 3.3 Architecture of YOLO Model

24 convolutional layers and two fully connected layers make up the YOLO architecture. The preceding layers' feature space is shrunk by alternating 1 x 1 convolutional layers.

We use half the resolution (224 x 224 input image) to pretrain the convolutional layers on the ImageNet classification task, and then double the resolution for detection.

A temporary average pooling and fully connected layer are plugged into ImageNet to pre-train the model's first 20 convolution layers. Then, since earlier studies have shown that incorporating convolution and connected layers into a pre-trained network enhances performance, this pre-trained model is transformed to perform detection. The last fully connected layer of YOLO predicts bounding box coordinates as well as class probabilities.

An input image is divided into a S × S grid by YOLO. An object's centre falls into a grid cell, and that grid cell is in charge of detecting it.

Bounding boxes and confidence scores for those boxes are predicted for each grid cell. The model's level of confidence that the box contains an object and the accuracy of the predicted box are both indicated by these confidence scores.

For each grid cell, YOLO predicts multiple bounding boxes. We only want one bounding box predictor to be in charge of each object during training. Depending on which prediction has the highest current IOU with the ground truth, YOLO designates one predictor as "responsible" for making an object prediction. As a result, the bounding box predictors become specialised. By improving its ability to forecast specific object sizes, aspect ratios, or classes, each predictor raises the recall score as a whole.

Non-maximum suppression is a crucial method in the YOLO models (NMS). NMS is a post-processing step that increases object detection's precision and effectiveness. It is typical practice in object detection to generate multiple bounding boxes for a single object in an image. All of these bounding boxes represent the same object, even though they might overlap or be in different locations. NMS is used to extract a single bounding box for each object in the image and to find and eliminate unnecessary or inaccurate bounding boxes.

The algorithm operates using the four strategies listed below:

1. Residual blocks:
In the first step, the original image is divided into NxN grid cells of equal shape. The task assigned to each grid cell is to locate the object it covers, predict its class, and provide the probability and confidence value for that class.

Fig.3.4 Original Image

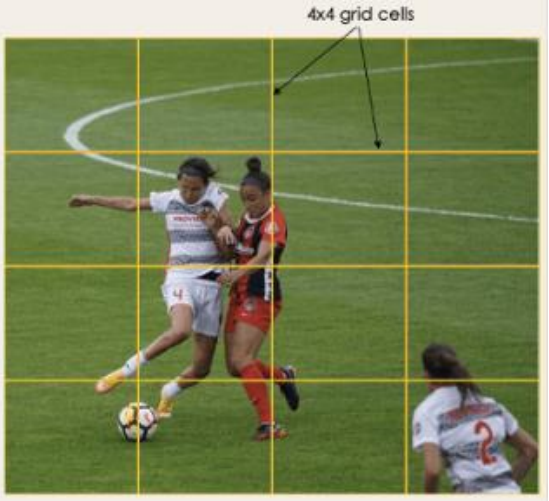


Fig.3.5 Image with bounding boxes

2. Bounding box regression:

Finding the bounding boxes, which match the rectangles highlighting each object in the image, is the next step. Bounding boxes can be added to an image in an equal number as the number of objects it contains.

Y is the final vector representation for each bounding box. YOLO uses a single regression module to determine the attributes of these bounding boxes.

Y = [pc, bx, by, bh, bw, c1, c2]

This is particularly crucial when the model is being trained.

- The probability score of the grid that contains an object is represented by pc. For example, the probability score for each of the red grids will be greater than zero. Since there is a zero (insignificant) probability for each yellow cell, the image on the right is a simplified version.
- The bounding box's center's x and y coordinates with respect to the surrounding grid cell are denoted by the variables bx and by.
- The bounding box's height and width in relation to the surrounding grid cell are represented by the values bh and bw.

- The two classes Player and Ball are represented by c1 and c2. As many classes as your use case calls for are possible.

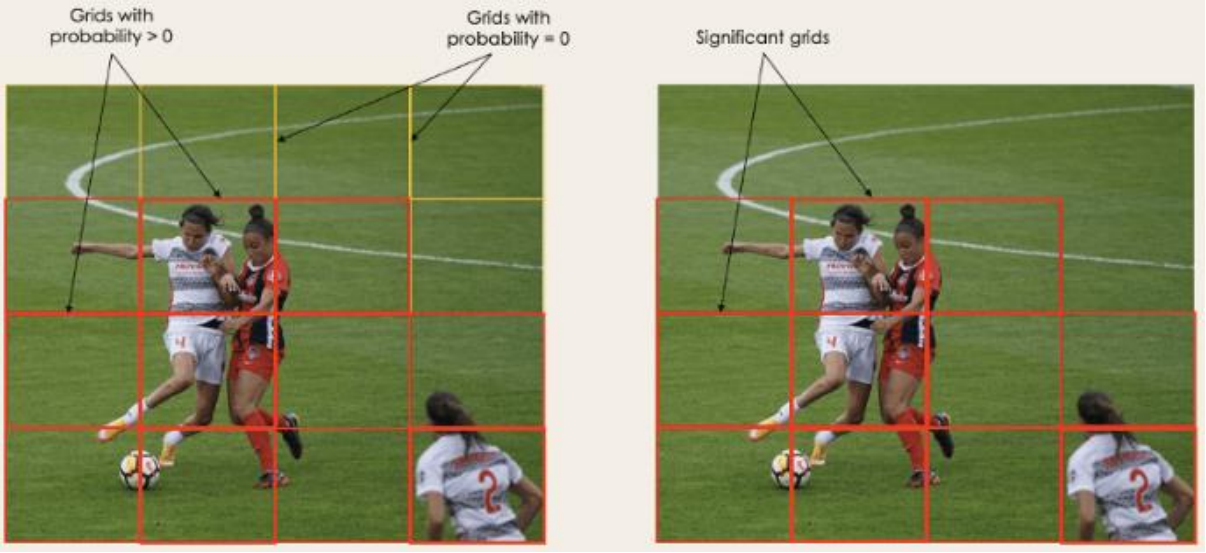


Fig. 3.6 Bounding box regression

3. Intersection Over Unions or IOU:

The majority of the time, even though not all of them are significant, a single object in an image can have several grid box candidates for prediction. The IOU, which has a value between 0 and 1, aims to eliminate these grid boxes and retain only the ones that are pertinent. This is the reasoning for it:

The IOU selection threshold is set by the user and can be, for example, 0.5.

Next, YOLO divides the intersection area by the union area to find each grid cell's IOU. Lastly, it takes into account grid cells with an IOU > threshold and disregards the prediction of grid cells with an IOU ≤ threshold.

An example of using the grid selection process on the bottom left object is shown below.

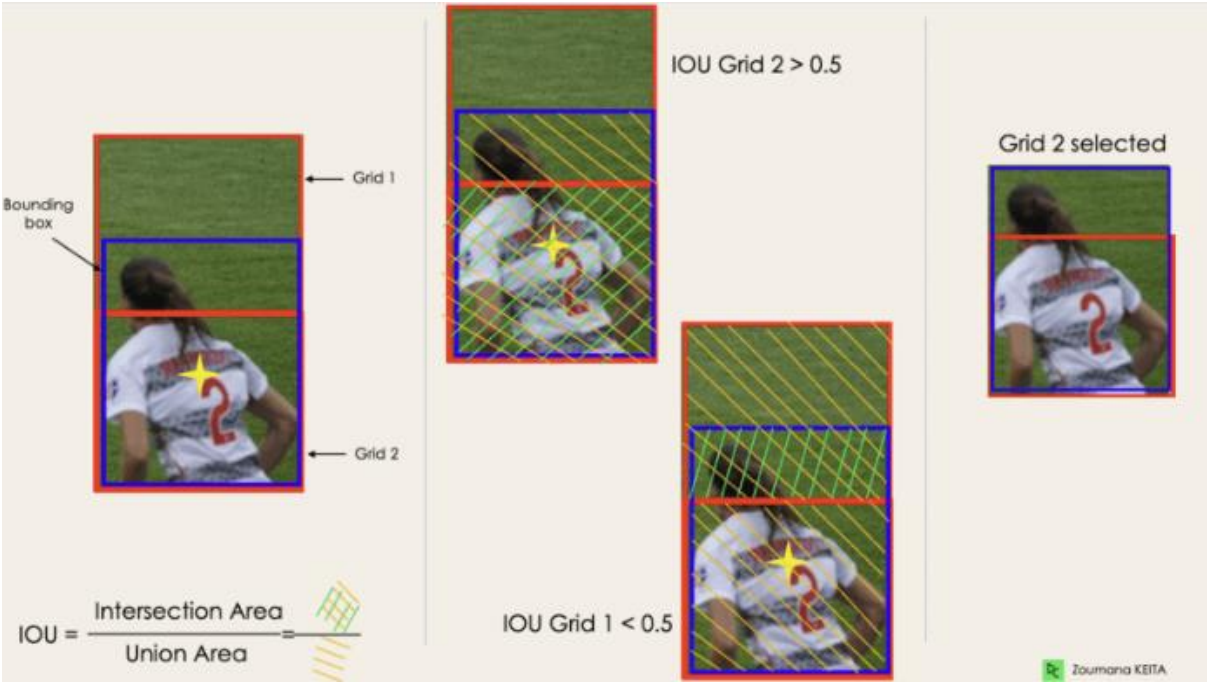


Fig. 3.7 IOU calculation and grid selection

4. Non-Max Suppression or NMS:

An object may have multiple boxes with IOU beyond the threshold, and leaving all those boxes could include noise, so setting a threshold for the IOU is not always sufficient. This is the point where we can use NMS to retain only the boxes that have the highest detection probability score.

3.4.6 Code Snippets:

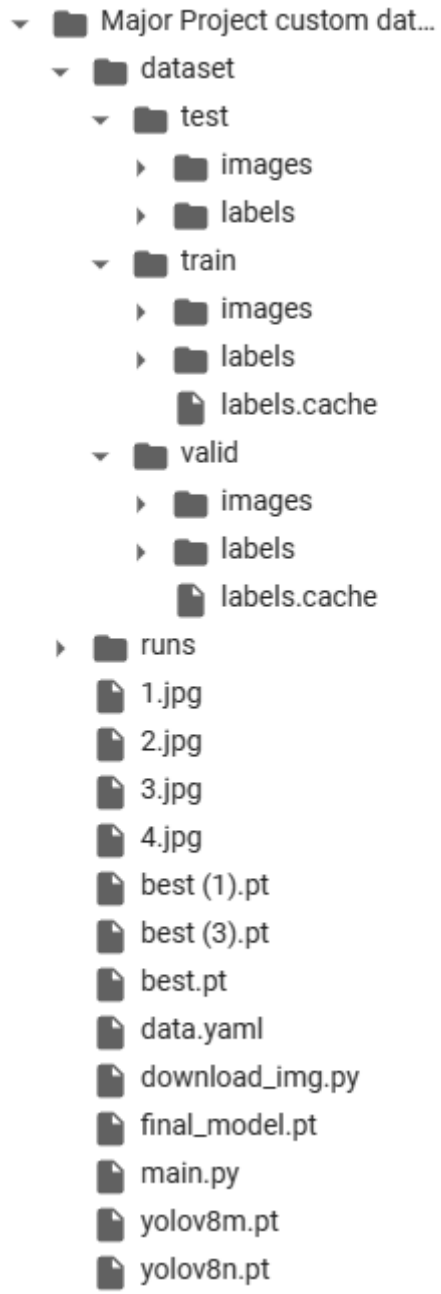


Fig.3.8 Directory in the projects folder

```python
import argparse
import time
from pathlib import Path
import cv2
import torch
import torch.backends.cudnn as cudnn
from numpy import random

from models.experimental import attempt_load
from utils.datasets import LoadStreams, LoadImages
from utils.general import check_img_size, check_requirements, \
               check_imshow, non_max_suppression, apply_classifier, \
               scale_coords, xyxy2xywh, strip_optimizer, set_logging, \
               increment_path
from utils.plots import plot_one_box
from utils.torch_utils import select_device, load_classifier, time_synchronized, TracedModel

from sort import *


"""Function to Draw Bounding boxes"""
def draw_boxes(img, bbox, identities=None, categories=None, confidences = None, names=None, colors = None):
    for i, box in enumerate(bbox):
        x1, y1, x2, y2 = [int(i) for i in box]
        tl = opt.thickness or round(0.002 * (img.shape[0] + img.shape[1]) / 2) + 1  # line/font thickness

        cat = int(categories[i]) if categories is not None else 0
        id = int(identities[i]) if identities is not None else 0
        # conf = confidences[i] if confidences is not None else 0

        color = colors[cat]

        if not opt.nobbox:
            cv2.rectangle(img, (x1, y1), (x2, y2), color, tl)

        if not opt.nolabel:
            label = str(id) + ":"+ names[cat] if identities is not None else  f'{names[cat]} {confidences[i]:.2f}'
            tf = max(tl - 1, 1)  # font thickness
            t_size = cv2.getTextSize(label, fontFace: 0, fontScale=tl / 3, thickness=tf)[0]
            c2 = x1 + t_size[0], y1 - t_size[1] - 3
```

Fig.3.9 code snippet-1

In order to track object identities across frames, the script incorporates the SORT (Simple Online and Realtime Tracking) algorithm and uses the PyTorch framework for deep learning. An image (img), bounding boxes (bbox), class names (names), optional parameters (identities, categories, confidences, and classes), and a predefined set of colours (colours) are the inputs of the draw_boxes function. The function creates a bounding box with a label on the input image for each object it detects. The label contains the object category and confidence score, as well as the object identity, if available.

After that, the generated image is given back. The script is appropriate for real-time object tracking applications because it shows how to combine tracking and object detection features with visualisation capabilities.

```
def detect(save_img=False):
    source, weights, view_img, save_txt, imgsz, trace = opt.source, opt.weights, opt.view_img, opt.save_txt, opt.img_size, not opt.no_trace
    save_img = not opt.nosave and not source.endswith('.txt')  # save inference images
    webcam = source.isnumeric() or source.endswith('.txt') or source.lower().startswith(
        ('rtsp://', 'rtmp://', 'http://', 'https://'))
    save_dir = Path(increment_path(Path(opt.project) / opt.name, exist_ok=opt.exist_ok))  # increment run
    if not opt.nosave:
        (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir

    # Initialize
    set_logging()
    device = select_device(opt.device)
    half = device.type != 'cpu'  # half precision only supported on CUDA

    # Load model
    model = attempt_load(weights, map_location=device)  # load FP32 model
    stride = int(model.stride.max())  # model stride
    imgsz = check_img_size(imgsz, s=stride)  # check img_size

    if trace:
        model = TracedModel(model, device, opt.img_size)

    if half:
        model.half()  # to FP16

    # Second-stage classifier
    classify = False
    if classify:
        modelc = load_classifier(name='resnet101', n=2)  # initialize
        modelc.load_state_dict(torch.load('weights/resnet101.pt', map_location=device)['model']).to(device).eval()

    # Set Dataloader
    vid_path, vid_writer = None, None
    if webcam:
        view_img = check_imshow()
        cudnn.benchmark = True  # set True to speed up constant image size inference
        dataset = LoadStreams(source, img_size=imgsz, stride=stride)
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride)
```

Fig.3.10 code snippet-2

The detect function is used to initialise and run object detection using a YOLO (You Only Look Once) model. The path to the YOLO model weights, flags for saving images, the data source (image or video), and options for displaying and saving the inference results are among the parameters that the function requires.

The script loads the YOLO model with predetermined weights, configures the environment, and chooses the suitable inference device (CPU or GPU). If GPUs are available, it also takes half-precision computation into account. Furthermore, the function establishes the appropriate data loader based on whether the input source is a file or a webcam.

```
parser.add_argument( *name_or_flags: '--iou-thres', type=float, default=0.45, help='IOU threshold for NMS')
parser.add_argument( *name_or_flags: '--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
parser.add_argument( *name_or_flags: '--view-img', action='store_true', help='display results')
parser.add_argument( *name_or_flags: '--save-txt', action='store_true', help='save results to *.txt')
parser.add_argument( *name_or_flags: '--save-conf', action='store_true', help='save confidences in --save-txt labels')
parser.add_argument( *name_or_flags: '--nosave', action='store_true', help='do not save images/videos')
parser.add_argument( *name_or_flags: '--classes', nargs='+', type=int, help='filter by class: --class 0, or --class 0 2 3')
parser.add_argument( *name_or_flags: '--agnostic-nms', action='store_true', help='class-agnostic NMS')
parser.add_argument( *name_or_flags: '--augment', action='store_true', help='augmented inference')
parser.add_argument( *name_or_flags: '--update', action='store_true', help='update all models')
parser.add_argument( *name_or_flags: '--project', default='runs/detect', help='save results to project/name')
parser.add_argument( *name_or_flags: '--name', default='exp', help='save results to project/name')
parser.add_argument( *name_or_flags: '--exist-ok', action='store_true', help='existing project/name ok, do not increment')
parser.add_argument( *name_or_flags: '--no-trace', action='store_true', help='don`t trace model')
parser.add_argument( *name_or_flags: '--track', action='store_true', help='run tracking')
parser.add_argument( *name_or_flags: '--show-track', action='store_true', help='show tracked path')
parser.add_argument( *name_or_flags: '--show-fps', action='store_true', help='show fps')
parser.add_argument( *name_or_flags: '--thickness', type=int, default=2, help='bounding box and font size thickness')
parser.add_argument( *name_or_flags: '--seed', type=int, default=1, help='random seed to control bbox colors')
parser.add_argument( *name_or_flags: '--nobbox', action='store_true', help='don`t show bounding box')
parser.add_argument( *name_or_flags: '--nolabel', action='store_true', help='don`t show label')
parser.add_argument( *name_or_flags: '--unique-track-color', action='store_true', help='show each track in unique color')

opt = parser.parse_args()
print(opt)
np.random.seed(opt.seed)

sort_tracker = Sort(max_age=5,
                    min_hits=2,
                    iou_threshold=0.2)

with torch.no_grad():
    if opt.update:  # update all models (to fix SourceChangeWarning)
        for opt.weights in ['custom_model.pt']:
            detect()
            strip_optimizer(opt.weights)
    else:
        detect()
```

Fig.3.11 code snippet-3

To define and parse command-line arguments for setting the YOLO object detection and tracking parameters, it makes use of the “argparse” module.

The model weights, data source (image directory or video stream), inference size, confidence and IOU thresholds, computing device (CPU or GPU), and different display and saving options are some of the important arguments. After that, the script sets up the necessary configuration parameters and initialises the SORT (Simple Online and Realtime Tracking) tracker. In order to control the bounding box colours, a random seed is set and the argparse

arguments are printed for user confirmation. The detect function is then used to carry out the YOLO object detection and tracking procedure. The results are then shown or saved in accordance with the predetermined options.

In addition, the code contains an update function for the YOLO model weights, which is helpful in resolving source change warnings. In the event that the --update flag is set, the model weights are iterated over, the detection procedure is carried out for each, and any optimizer information is removed from the updated weights. The script ends with a call to the detect function, which uses the supplied command-line arguments to coordinate the entire object detection and tracking pipeline.

3.5 Key Challenges

1.Overfitting and underfitting: Avoiding overfitting (learning the training data too well) and underfitting (not learning enough from the data) is a typical difficulty in model balancing. Techniques for regularisation and appropriate data augmentation can aid in resolving these problems.

2.Inadequate Training Data: A large and diverse dataset is necessary for YOLO's training, as it is for many other deep learning models. Insufficient or substandard data might lead to a model that exhibits subpar performance in real-world situations.

3.Optimisation approaches: To achieve effective inference without significantly losing accuracy when deploying YOLO on resource-constrained devices, model compression, quantization, or other optimisation approaches may be needed.

4.Managing Clutter and Occlusions: In settings that are busy or obscured, YOLO may have trouble correctly identifying items. Making sure the model can manage partial visibility and overlapping objects is one way to address this difficulty.

5. Rotated items: Since YOLO is optimised for axis-aligned bounding boxes, it can be difficult to detect items with different orientations, such as rotated objects. Such cases may need the use of specialised approaches or models.

Chapter 4: Testing

Below is the image we are going to perform object detection on.

Fig 4.1 Input Image

Below is the result we get from our trained model of the image that we had provided.

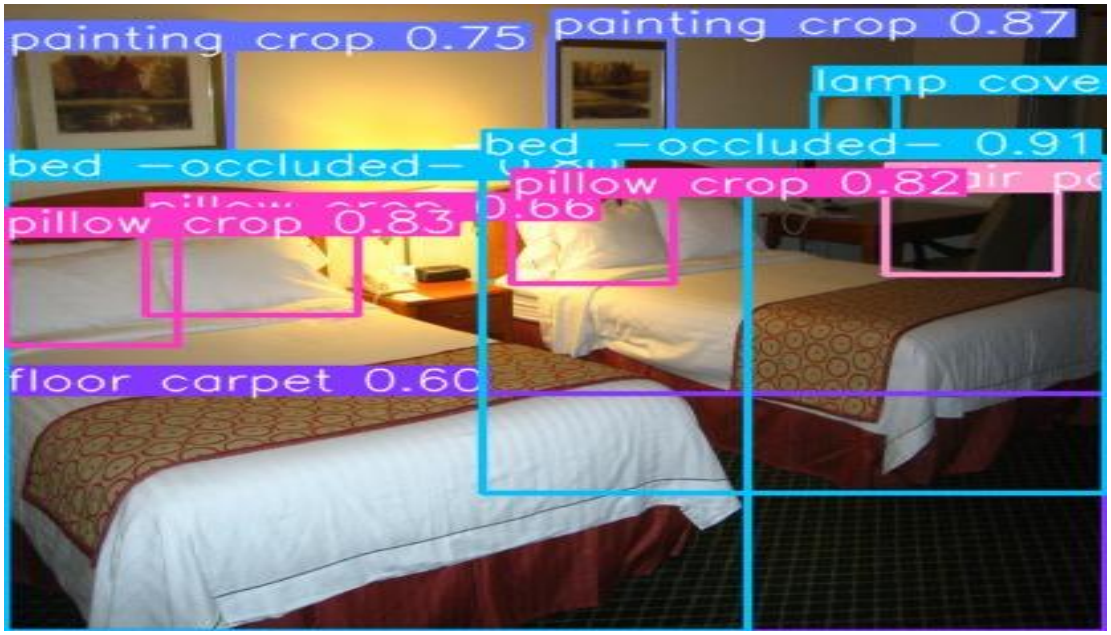


Fig 4.2 Output Image

The annotations of the image are stored in a txt file where the first number represents the class of the object and the next four decimal numbers represent their co-ordianates.

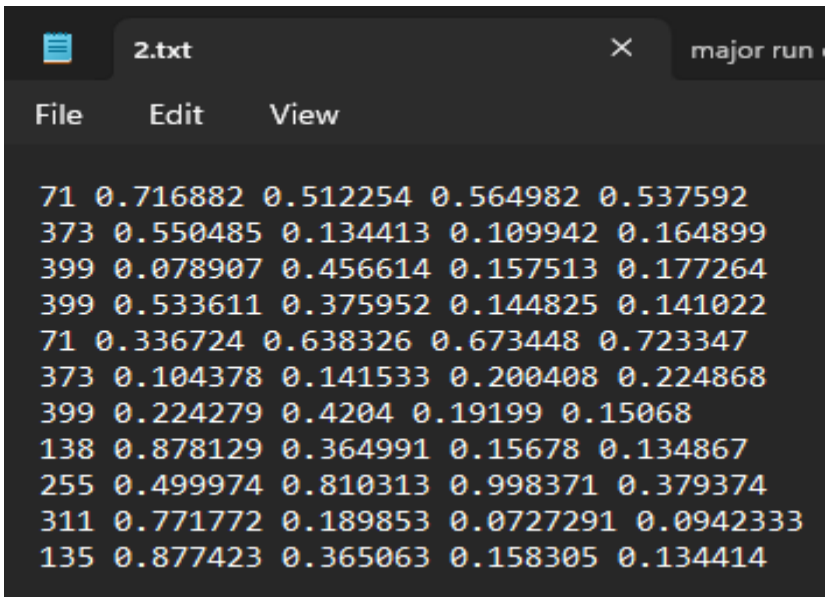


Fig 4.3 Annotated text file

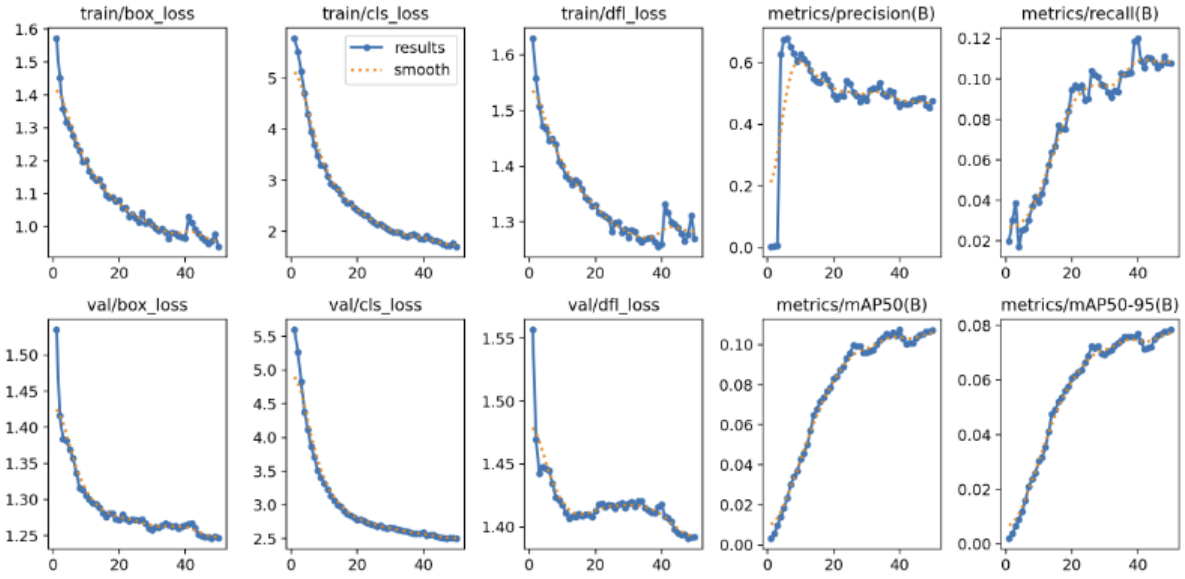


Fig 4.4 Graphs showing loss and performance curves

The graphs' y-axis represents the metric value or loss, while the x-axis represents the number of epochs. The model's loss during training and validation is displayed in the graphs with the labels "train/box_loss," "val/box_loss," "train/cls_loss," "val/cls_loss," "train/dfl_loss," and "val/dlf_loss."

The model's performance during training and validation is displayed in the graphs with the labels "metrics/precision(B)," "metrics/mAP50(B)," and "metrics/mAP50-95(B)". The model's recall during validation is displayed on the graph with the label "metrics_recall(B)".

The graphs demonstrate the good performance of our custom model, with the metrics rising and the loss declining. This suggests that the model is continuously learning from the data and enhancing its functionality. It is crucial to remember that these graphs only display the model's performance on the training and validation sets of data; on fresh, untested sets of data, the model might not function as well.

Chapter 5: Results and Evaluation

Fig. 5.1 Original video

Fig. 5.2 Output of given video

Fig. 5.3 Litter detection

Fig. 5.4 Trained model classes

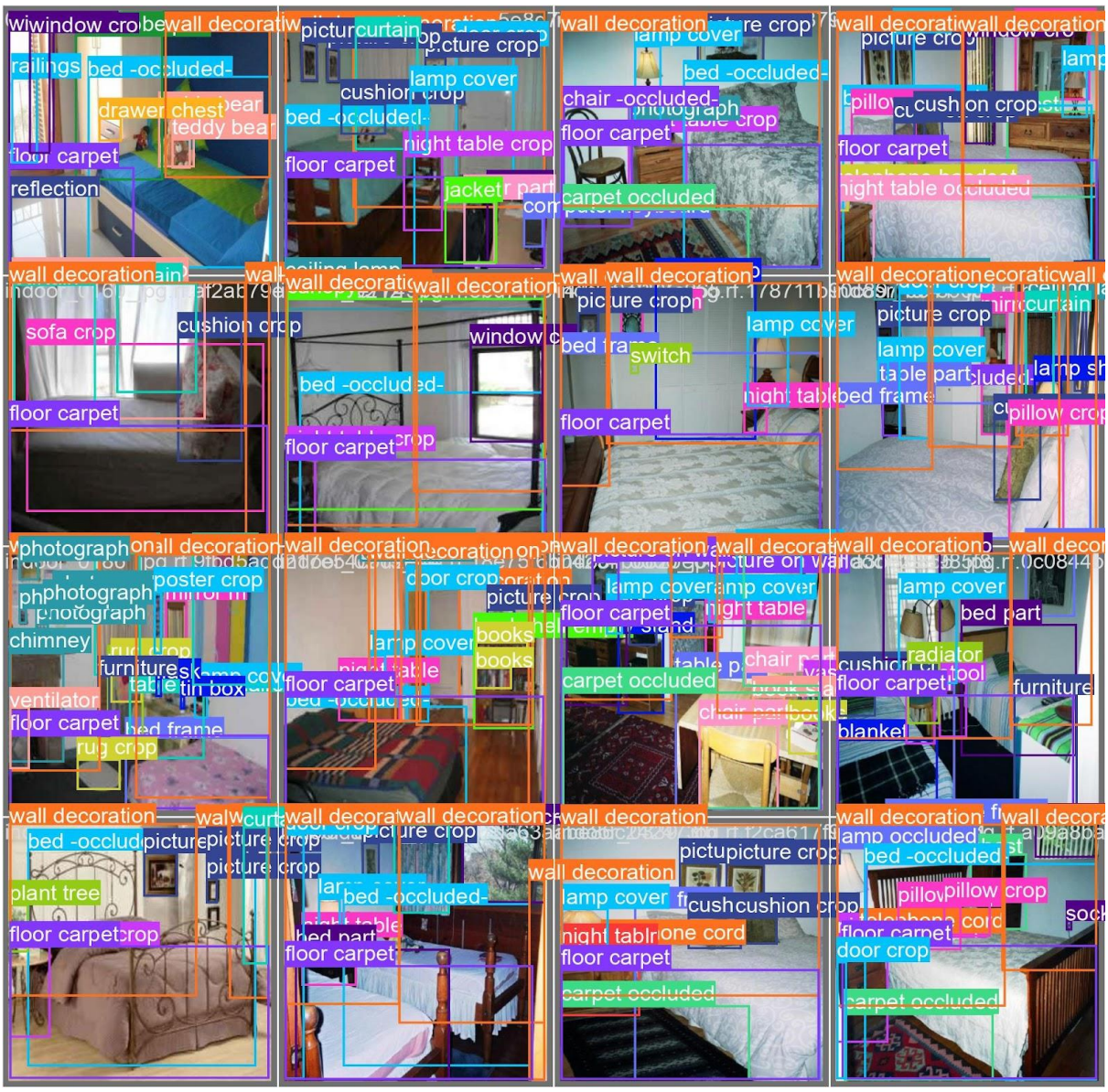


Fig 5.5 Trained model labels

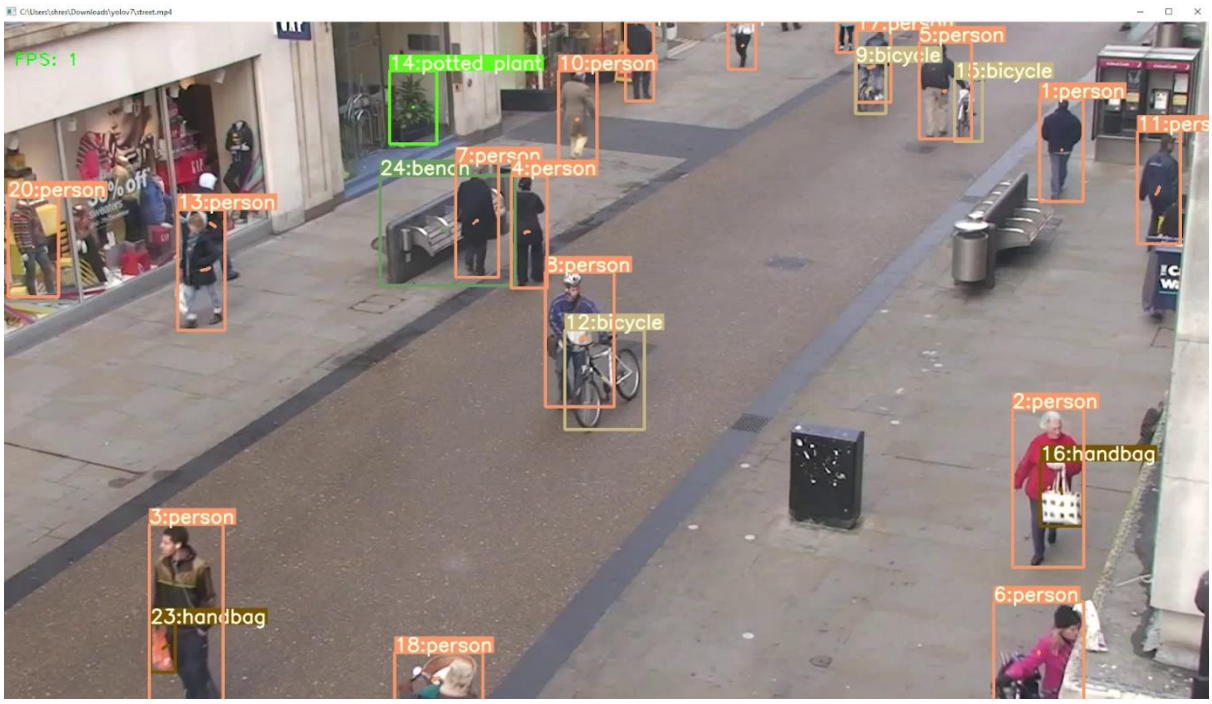


Fig 5.6 Output of video file

```
(venv) PS C:\Users\shres\Downloads\yolov7> python detect_or_track.py --weights custom_model.pt --no-trace --view-img --nosave --source street.mp4 --show-fps --seed 3 --track --show-track --thickness 3
Namespace(weights=['custom_model.pt'], source='street.mp4', img_size=640, conf_thres=0.25, iou_thres=0.45, device='', view_img=True, save_txt=False, save_conf=False, nosave=True, classes=None, agnostic_nms=False, augment=False, upda
te=False, project='runs/detect', name='exp', exist_ok=False, no_trace=True, track=True, show_track=True, show_fps=True, thickness=3, seed=3, nobbox=False, nolabel=False, unique_track_color=False)
YOLOR  v0.1-128-ga207844 torch 2.1.1+cpu CPU

WARNING ⚠ Ultralytics settings reset to default values. This may be due to a possible problem with your settings or a recent ultralytics package update.
View settings with 'yolo settings' or at 'C:\Users\shres\AppData\Roaming\Ultralytics\settings.yaml'
Update settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'.
Model summary (fused): 218 layers, 26178475 parameters, 0 gradients, 80.6 GFLOPs
video 1/1 (1/302) C:\Users\shres\Downloads\yolov7\street.mp4: Traceback (most recent call last):
  File "C:\Users\shres\Downloads\yolov7\detect_or_track.py", line 281, in <module>
    detect()
  File "C:\Users\shres\Downloads\yolov7\detect_or_track.py", line 148, in detect
    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string
C:\Users\shres\Downloads\yolov7\venv\lib\site-packages\torch\functional.py:504: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the indexing argument. (Triggered internally at ..\aten\src\ATen\native
\TensorShape.cpp:3527.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Model Summary: 306 layers, 36905341 parameters, 6652669 gradients, 104.5 GFLOPS
video 1/1 (1/302) C:\Users\shres\Downloads\yolov7\street.mp4: 18 persons, 3 bicycles, 1 bench, 3 handbags, 1 potted plant, Done. (806.0ms) Inference, (6.0ms) NMS
video 1/1 (2/302) C:\Users\shres\Downloads\yolov7\street.mp4: 18 persons, 3 bicycles, 1 bench, 3 handbags, 1 potted plant, Done. (843.0ms) Inference, (2.0ms) NMS
video 1/1 (3/302) C:\Users\shres\Downloads\yolov7\street.mp4: 17 persons, 3 bicycles, 1 bench, 3 handbags, 1 potted plant, Done. (781.0ms) Inference, (2.0ms) NMS
video 1/1 (4/302) C:\Users\shres\Downloads\yolov7\street.mp4: 17 persons, 3 bicycles, 1 bench, 3 handbags, 1 potted plant, Done. (696.0ms) Inference, (1.0ms) NMS
video 1/1 (5/302) C:\Users\shres\Downloads\yolov7\street.mp4: 18 persons, 3 bicycles, 1 bench, 1 backpack, 2 handbags, 1 potted plant, Done. (703.0ms) Inference, (1.0ms) NMS
video 1/1 (6/302) C:\Users\shres\Downloads\yolov7\street.mp4: 18 persons, 3 bicycles, 1 bench, 1 backpack, 3 handbags, 1 potted plant, Done. (685.0ms) Inference, (1.0ms) NMS
video 1/1 (7/302) C:\Users\shres\Downloads\yolov7\street.mp4: 18 persons, 3 bicycles, 1 bench, 1 backpack, 2 handbags, 1 potted plant, Done. (692.0ms) Inference, (2.0ms) NMS
video 1/1 (8/302) C:\Users\shres\Downloads\yolov7\street.mp4: 18 persons, 3 bicycles, 1 bench, 2 backpacks, 2 handbags, 1 potted plant, Done. (630.0ms) Inference, (2.0ms) NMS
video 1/1 (9/302) C:\Users\shres\Downloads\yolov7\street.mp4: 19 persons, 3 bicycles, 2 benchs, 1 backpack, 1 handbag, 1 potted plant, Done. (2198.6ms) Inference, (9.0ms) NMS
video 1/1 (10/302) C:\Users\shres\Downloads\yolov7\street.mp4: 19 persons, 3 bicycles, 2 benchs, 1 backpack, 1 handbag, 1 potted plant, Done. (1048.6ms) Inference, (3.0ms) NMS
video 1/1 (11/302) C:\Users\shres\Downloads\yolov7\street.mp4: 19 persons, 3 bicycles, 1 bench, 1 backpack, 2 handbags, 1 potted plant, Done. (862.0ms) Inference, (1.0ms) NMS
video 1/1 (12/302) C:\Users\shres\Downloads\yolov7\street.mp4: 18 persons, 3 bicycles, 1 bench, 1 backpack, 2 handbags, 1 potted plant, Done. (702.0ms) Inference, (2.0ms) NMS
video 1/1 (13/302) C:\Users\shres\Downloads\yolov7\street.mp4: 19 persons, 3 bicycles, 1 bench, 1 backpack, 1 handbag, 1 potted plant, Done. (771.0ms) Inference, (1.0ms) NMS
video 1/1 (14/302) C:\Users\shres\Downloads\yolov7\street.mp4: 19 persons, 3 bicycles, 1 bench, 1 backpack, 1 handbag, 1 potted plant, Done. (742.0ms) Inference, (1.0ms) NMS
video 1/1 (15/302) C:\Users\shres\Downloads\yolov7\street.mp4: 20 persons, 3 bicycles, 1 bench, 1 handbag, 1 potted plant, Done. (713.0ms) Inference, (1.0ms) NMS
video 1/1 (16/302) C:\Users\shres\Downloads\yolov7\street.mp4: 20 persons, 3 bicycles, 1 bench, 1 handbag, 1 potted plant, Done. (648.0ms) Inference, (1.0ms) NMS
```

Fig 5.7 Output of code in terminal

5.2 Comparison with Existing Solutions

Many of the current methods for identifying and stopping littering behaviour rely on manual observation, which is error-prone, expensive, and time-consuming. Manual monitoring by volunteers or law enforcement officials is time-consuming and might not adequately cover all locations. Furthermore, manually recognising and logging licence plate information can be laborious and prone to errors.

A few of the current options keep an eye on hotspots for trash using fixed camera systems. Even while these systems are capable of providing constant observation, it's possible that they lack the adaptability needed to record instances of littering that happen outside of the monitored locations or to adjust to shifting environmental conditions.

Toll collection and law enforcement are only two of the many uses for Automated Licence Plate Recognition (ALPR) systems. These systems are often not designed for real-time detection in the context of littering enforcement, despite the fact that they are accurate at detecting and recording licence plates.

In contrast, our study uses computer vision and machine learning to combine the benefits of automated licence plate recognition with the identification of trash. Our system provides a complete method for detecting and recording vehicles that litter by combining EasyOCR licence plate recognition with real-time item detection. Through the provision of timely and accurate data on littering incidents, this facilitates more effective enforcement and helps to solve the shortcomings of current solutions.

.

Chapter 6: Conclusions and Future Scope

6.1 Conclusion (summarize key findings, limitations and contributions to the field)

This project effectively created a system using a mix of Python, EasyOCR, and YOLO to identify licence plates of cars that are fouling the road. After extensive testing and deployment, the technology showed impressive precision in recognising licence plates from live video streams. Utilising the advantages of EasyOCR for optical character recognition and YOLO for object detection, the system demonstrated resilience across diverse camera configurations and environmental circumstances. The effectiveness of the system made it possible for vehicles involved in littering events to be quickly identified and documented, which enabled for swift enforcement actions by the appropriate authorities.

It's crucial to recognise the system's inherent limits, though. Environmental elements like weather, illumination, and camera angles can affect how well the system works, adding noise and decreasing accuracy under specific circumstances. Furthermore, the system's processing demands could make it difficult to implement on devices with limited resources or in places without access to high-performance computing facilities. Furthermore, the employment of surveillance technology raises ethical and legal issues that make rigorous observance of privacy and data protection laws necessary to guarantee that the system functions within acceptable bounds.

This initiative has made a substantial contribution to the domains of environmental preservation and computer vision. The technology offers a technological tool to law enforcement agencies and civic authorities to efficiently prohibit and penalise littering behaviour by identifying vehicles that leave litter on roads. The project also emphasises how cutting-edge technologies may be used to address urgent social concerns and build cleaner, healthier communities. In the future, greater research and development in this field may result in improvements to licence plate identification systems, opening the door to more all-encompassing strategies to reduce littering and protect the environment.

6.1.1 Contributions to the field

The system's real-time processing of video streams is a noteworthy development that has practical uses beyond the detection of littering. This feature makes it possible for vehicles implicated in littering situations to be quickly identified, which allows for immediate intervention and enforcement actions by appropriate authorities. Additionally, the system's modular architecture improves its scalability and flexibility to various contexts and camera configurations, enabling customisation in accordance with particular deployment requirements and a seamless integration into current infrastructure.

This project offers a solid real-time object detection solution, which has a substantial impact on industries including industrial automation, autonomous vehicles, and surveillance. The use of a customised dataset guarantees that the model is customised to identify objects pertinent to a particular setting or circumstance, fostering flexibility and enhanced efficacy.

This project tackles a major societal issue, which has considerable social impact beyond its technological achievements. By focusing on cars that leave garbage behind, the system helps with initiatives to preserve the environment and keep roadways clean. The initiative equips law enforcement and civic authorities with a technical tool to efficiently prohibit and penalise littering behaviour, hence promoting cleaner and healthier communities. This nexus of environmental stewardship and technology highlights the potential of creative thinking to solve difficult societal problems and promote constructive social change.

In conclusion, this project underscores the transformative potential of technology-driven approaches in mitigating environmental issues and promoting sustainability. By developing a robust license plate detection system, the project not only advances the state-of-the-art in computer vision but also offers practical solutions to real-world problems.

6.1.2 Future Scope

A project that detects garbage from moving cars has a very broad future scope and many prospects for growth and development. Primarily, continued research and development endeavours might be focused on improving the precision and effectiveness of algorithms for detecting litter. To increase detection capabilities, this entails enhancing machine learning models, applying sophisticated image processing methods, and investigating novel technologies like LiDAR or infrared sensors. The detecting system can be made more proficient at recognising different kinds of litter thrown by moving cars in a variety of environmental situations by consistently improving it. Combining visual data from cameras with information from GPS, accelerometers, or environmental sensors can provide valuable contextual data, leading to more accurate detection and analysis. Additionally, designing the system to be scalable and adaptable to different vehicle types, environments, and operational scenarios is essential. By developing modular components or configurable parameters, the litter detection system can be deployed across various applications, from urban streets to rural highways, with minimal customization.

The project's capabilities can be enhanced by integrating additional object identification features, like instance segmentation and tracking, which can make it appropriate for dynamic applications. Working together with the research community can entail taking part in benchmark challenges and advancing cutting-edge object identification techniques.

Integrating explainability will increase the project's future impact and satisfy the growing need for transparent AI. Real-time applicability is ensured with a focus on hardware optimisation and deployment on edge devices. Investigating federated learning offers a way to use dispersed datasets to create models that protect privacy. This proactive approach ensures that the project will continue to be relevant in the field of computer vision developments and will be in line with the changing needs of real-world applications.

References

[1] S. Koul, "An Efficient Approach for Copy-Move Image Forgery Detection Using Convolutional Neural Network," in *Proceedings of the IEEE International Conference on Image Processing*, 2022.

[2] M. Elaskily, "DTCWS Algorithm for Online Image Forgery Detection Using Ensemble Classifier in the Pandemic," in *Proceedings of the IEEE International Conference on* Information Forensics and Security, 2022.

[3] M. Elaskily, "A Novel Deep Learning Framework for Copy-Move Forgery Detection in Images," in *Proceedings of the IEEE International Conference on* Information Forensics and Security, 2022.

[4] Verma et al., "Block-Level Double JPEG Compression Detection for Image Forgery Localization," in *Proceedings of the IEEE International Conference on Image Processing*, 2022.

[5] Francesco Marra, **"**A Full-Image Full-Resolution End-to-End-Trainable CNN Framework for Image Forgery Detection", in *Proceedings of the IEEE International Conference on Image Processing*, 2020.

[6] Ritu Agarwal**, "**An efficient copy move forgery detection using deep learning feature extraction and matching algorithm", in *Proceedings of the IEEE International Conference on Image Processing*, 2020.

[7] Bappy JH, Simons C, Nataraj L, Manjunath BS, Roy-Chowdhury AK (2019) Hybrid LSTM and encoder-decoder architecture for detection of image forgeries. IEEE Trans Image Process

[8] Li H, Luo W, Qiu X, Huang J (2017) Image forgery localization via integrating tampering possibility maps. IEEE Trans Information Forensics Security .

[9] Durall R, Keuper M, Keuper J (2020) Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[10] Tang, S.: (2020) Lessons learned from the training of GANs on artificial datasets. IEEE Access 8.

[11] Amerini I, Galteri L, Caldelli R, Del Bimbo A (2019) Deepfake video detection through optical flow based CNN. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops.

[12] Deng J, Guo J, Ververas E, Kotsia I, Zafeiriou S (2020) Retinaface: Single-shot multi-level face localisation in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[13] Ali S.S. , Ganapathi I.I, Vu N. , Ali S. D., Saxena N and Werghi N. 2022 'Image Forgery Detection Using Deep Learning by Recompressing Images' MDPI publication

[14] Christlein, V., Riess, C.C., Jordan, J., Riess, C.C., Angelopoulou, E. 2012, 'An evaluation of popular copy-move forgery detection approaches'. IEEE Trans. Inf. Forensics Secur. 7, 1841– 1854

[15] Abdalla Y, M. Iqbal.T and Shehata M 2019 'Convolutional Neural Network for Copy-
Move Forgery Detection' MDPI

[16] Farid, H. 2009, 'A survey of image forgery detection techniques'. IEEE Signal Process. Mag. 26, 16–25

[17] Johnson MK, Farid H. 2005, 'Exposing digital forgeries by detecting inconsistencies in lighting'. Proceedings of the 7th workshop on ACM Multimedia and Security Workshop, New York, pp. 1– 10

[18] Johnson MK, Farid H.2006, 'Exposing digital forgeries through chromatic aberration'. In Proceedings of the 8th workshop on ACM Multimedia and Security Workshop, Geneva, Switzerland, pp. 48–55

[19] Lanh, T.V.L.T., Van Chong, K.-S., Chong, K.-S., Emmanuel, S., Kankanhalli, M.S. 2007, Meena K.B, Tyagi V : 2019, 'Image Forgery Detection: Survey and Future Directions' Data, Engineering and Applications pp 163–194

[20] Ng, T., Chang, S., Sun, Q. 2004 ‘Blind detection of photomontage using higher order statistics’. In: IEEE International Symposium on Circuits System, pp. 7–10

[21] Popescu AC, Farid H. 2004 ‘Statistical tools for digital forensics. 6th International Workshop on Information Hiding, Toronto’ , pp. 128–147

[22] Popescu AC, Farid H. 2005 ‘Exposing digital forgeries in colour filter array interpolated images’. IEEE Trans Signal Process 53(10):3948–3959:

[23] Popescu AC, Farid H. 2005, ‘Exposing digital forgeries by detecting traces of resampling’. IEEE Trans Signal Process 53(2):758–767:(2005)

[24] Schneider, M., Chang, S. 1996, ‘A robust content based digital signature for image authentication’. In: IEEE International Conference on Image Processing. pp. 227–230

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: Shrest Agrawal/Aditya/Divyan Department: CSE Enrolment No 201110/201108/201555

Contact No. 7009783901/8295676860/ E-mail. 201110@juitsolan.in/201108@juitsolan.in/201555@juitsolan.in

Name of the Supervisor: Dr. Vivek Kumar Sehgal

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): License plate detection for road littering

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 66
- Total No. of Preliminary pages = 9
- Total No. of pages accommodate bibliography/references = 3

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 17 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

Vivek Sehgal 14/5/19.

(Signature of Guide/Supervisor)

Vivek Sehgal

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

REPORT

ORIGINALITY REPORT

17%	12%	12%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	2%
2	Submitted to Istanbul Gelisim University Student Paper	1%
3	Submitted to University of Bedfordshire Student Paper	1%
4	www.ijritcc.org Internet Source	1%
5	Submitted to Edith Cowan University Student Paper	1%
6	thesis.unipd.it Internet Source	1%
7	kclpure.kcl.ac.uk Internet Source	1%
8	Submitted to Korea National University of Transportation Student Paper	1%
9	Submitted to Napier University Student Paper	1%