Autonomous Vehicle Tracking for Drone Using Deep Learning

Project report submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Bachelor of Technology
In
Mechanical Engineering

by

Mirthipati Abhiram (BT20MEC067)

Mrunali Ukey (BT20MEC112)

under the guidance of

Dr Shital S Chiddarwar

Department of Mechanical Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

April 2024

Autonomous Vehicle Tracking for Drone Using Deep Learning

Project report submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of
the degree

Bachelor of Technology
In
Mechanical Engineering

by

Mirthipati Abhiram (BT20MEC067)

Mrunali Ukey (BT20MEC112)

under the guidance of

Dr Shital S Chiddarwar

Department of Mechanical Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

April 2024

Department of Mechanical Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

We, Mirthipati Abhiram **(BT20MEC067)** and Mrunali Ukey **(BT20MEC112)**, hereby declare that this project work titled "**Autonomous Vehicle Tracking for Drone Using Deep Learning**" is carried out by us in the Department of Mechanical Engineering of Visvesvaraya National Institute of Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution / University.

Date:

Enrollment No	Name	Signature
BT20MEC067	Mirthipati Abhiram	
BT20MEC112	Mrunali Ukey	

Certificate

This is to certify that the project titled "**Autonomous Vehicle Tracking for Drone Using Deep Learning**", submitted by **Mirthipati Abhiram (BT20MEC067) and Mrunali Ukey (BT20MEC112), of the students** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Mechanical Engineering**, VNIT Nagpur. The work is comprehensive, complete, and fit for final evaluation.

Name of the Supervisor
Designation,
Department of Mechanical Engineering,
VNIT, Nagpur

Head, Department of Mechanical Engineering
VNIT, Nagpur
Date:22-04-2024

ACKNOWLEDGEMENT

First and foremost, we would like to express our utmost gratitude to Dr. Shital S Chiddarwar, who has been an outstanding advisor and mentor for us and many other students, inspiring us to do our best and amplifying it by her skillful guidance, vast knowledge, and dedicated attitude. We thank her, for so much we have learned from her lectures and collaboration, for her caring support and understanding and for her continuous enthusiasm about research and teaching.

We would also like to thank Prof. P. M. Padole, Director V.N.I.T & Prof. Atul B. Andhare, Head of Department and Prof. S. B. Thombre, former Head of Department & Prof. Ravikumar Dumpala and Prof. Ajinkya Baxy, our FYP Panel members and for all the administrative support.

We thank Dr. Ravikumar Dumpala, Coordinator for B. Tech projects 2023-24 for following up our all-progress seminar to give his valuable suggestions and encouraging comments during various stages of this research work.

We would also like to specially thank Dr. S. S. Chiddarwar, Dr. N. K. Lautre, Dr. A. K. Srivastav, Dr. D. A. Jolhe and Dr. T. V. K. Gupta, VNIT, Nagpur, and Department of Mechanical Engineering for their guidance and useful suggestions during the course of whole research work.

Our thanks also go to the members of our research scholar group, from whom we have learned much and got a lot of help. We would like to thank Rahul M R, Sathish Sonawane, A Vinay and Utkarsh V for the scientific discussions, friendly behavior and useful timely advice on all matters.

Mirthipati Abhiram
Mrunali Ukey

ABSTRACT

This paper proposes a combination of different techniques in a chronological order to achieve proper and accurate autonomous vehicle tracking for a drone. The drone we used to deploy the final algorithm is "DJI Tello" drone. Traditional tracking methods often struggle with range of the target, range, occlusion, illumination changes. Deep Learning allows us to train the model with specific complex scenarios to overcome those struggles.

It is critical to be able to detect and acquire objects efficiently in the wide skies where drones fly, and autonomous systems operate. In addition to guaranteeing the effectiveness and safety of airborne operations, this capacity creates a plethora of opportunities for applications ranging from search and rescue to surveillance. The combination of deep learning with visual tracking, which has the potential to completely change the autonomous systems landscape, is at the core of this project.

This project time line consisted many numbers of flips and turns in terms of technologies used and methodologies involved, because we wanted to try all the possible opportunities to build an algorithm for autonomous drone tracking, starting with understanding the tracking concept we studied human face tracking, pre trained deep learning models like YOLO algorithms for object tracking.

Further we aimed to design a unique locking system utilizing the Bag-of-Features (BoF) visual algorithm for robust visual tracking and target acquisition. Traditional locking methods often struggle with range of the target, occlusion, illumination changes. We focused on trialing and testing the locking system on a sample target.

Real time live tracking often demands diverse scenarios, therefore we also trained few of the object classification, detection and tracking models with truly handpicked and customized datasets to handle peculiar scenarios while tracking the object.

There were many challenges faced by our team during this project phase some of them were eventually overcame and some of them are still needed to be developed in order to achieve a precise target tracking using drones. The future scope of the project is also clearly explained in this thesis. We hope that we have given a significant contribution in autonomous industry through this project.

Key Words:
Autonomous target tracking, Drone, DJI Tello, Deep Learning, Visual tracking, Object tracking, Bag-of-Features (BoF), Real-time tracking, Object classification, Object Detection, Dataset.

LIST OF FIGURES

NOMNECULATURE

CNN	Convolutional Neural Networks
DNN	Deep Neural Network
mPA	Mean Average Precision
RMSE	Root Mean Square Error
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
BoF	Bag of Features
SIFT	Scale Invariant Feature Transform
YOLO	You Only Look Once
SSD	Single Shot Detector
DMM	Dynamic Motion Model
UAV	Unmanned Aerial Vehicle
SURF	Speeded-Up Robust Features
RCNN	Region Based Convolution Network
DSOD	Deeply Supervised Object Detection
DSSD	Deconvolutional Single Shot Detector
FPN	Feature Pyramid Network
APYC	Adaptive Proportional Yaw Control

RANSAC	Random Sample Consensus
bbox	Bounding Box
RPN	Region Proposed Network
IoU	Intersection Over Union

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 General

Developments in vision technologies at current situation are innumerable. Autonomous tracking of a vehicle using a drone by incorporating Deep Learning techniques into it plays a crucial role in development of this autonomous vision domain. The fundamental components of autonomous aerial operations - visual tracking, target acquisition, and locking allow drones and to recognize and track things of interest in real time. However, obstacles like occlusion, varying range, and challenging ambient conditions frequently cause classic tracking algorithms to fail.

Advancements in Object recognition and targeting systems have gained a lot of attention since they have applications in a lot of different areas, like autonomous navigation, robotics, and surveillance. The capacity of these systems to precisely locate and latch onto particular targets in dynamic situations is one of their most important features. Conventional object identification techniques frequently struggle to effectively detect and identify distinguishing characteristics, particularly when targets have complicated forms or different orientations.

This project explores the field of visual tracking and target acquisition in autonomous systems, emphasizing the use of vision techniques and deep learning approaches to improve performance and overcome inherent difficulties. Important components that are essential to our investigation include target locking, range estimation, and occlusion mitigation. These components are all vital to guaranteeing tracking system’s precision and dependability.

A paradigm shift is brought about by the incorporation of deep learning into visual tracking, allowing autonomous systems to adjust and learn on their own to a variety of settings. Deep learning algorithms can identify complex patterns and characteristics through training on large datasets that contain annotated examples of target objects in a variety of scenarios. This allows for reliable target acquisition and tracking even when there is environmental fluctuation.

The combined problem of utilizing deep learning's transformative potential while addressing ethical issues and guaranteeing the safe and responsible deployment of autonomous systems faces us as we proceed through this investigation. Our study aims to pave the way for a future where autonomous airborne platforms act as intelligent partners, enhancing human skills and driving societal progress from increasing computational efficiency to protecting privacy and security.

Our target through this work is to achieve autonomous target tracking for a drone using a target locking system that will be discussed in this paper. Hence all the images and the video feeds used for this work were captured and recorded from DJI Tello mini drone's camera. The specifications of this drone are as follows.

Table 1.1 Specifications of DJI Tello mini drone

Camera resolution	5MP
Video resolution	720p
Live video feed resolution	720p
Live video feed frame rate	30fps

However, the implementation of autonomous tracking comes with its own set of challenges. The primary challenges are the target locking, primary method that we chose for locking the target is unique feature extraction machine vision algorithms (Scale Invariant Feature Transform, SIFT Algorithm & Bag of Features, BoF Algorithm) and conducted a comparison work between them. From this we have derived a valuable conclusion on these extracting the unique feature extracting algorithm. But since our purpose is to track the vehicles in real-time, we decided Deep learning could be the best method, and further the work proceeded in using deep learning algorithms

However, the implementation of autonomous tracking comes with its own set of challenges. The primary challenges are the target locking, primary method that we chose for locking the target is unique feature extraction machine vision algorithms (Scale Invariant Feature Transform, SIFT Algorithm & Bag of Features, BoF Algorithm) and conducted a comparison work between them. From this we have derived a valuable conclusion on these extracting the unique feature extracting algorithm. But since our purpose is to track the vehicles in real-time, we decided Deep learning could be the best method, and further the work proceeded in using deep learning algorithms.

Vehicle tracking, theft, search and rescue, film making, sports, races, security and surveillance are some of the novel applications of this project

1.2 Aim, Objectives and Scope of Research Work

The aim of this research work is to design and build an efficient algorithm for a Tello mini drone to automatically follow and track a vehicle.
The specific objectives to attain this aim are formulated as:

1. Understanding the basics of detection and tracking algorithms from human face tracking and pre trained YOLO models
2. Target locking using unique feature extractors
3. Preparing quality dataset that aligns with the model needs
4. Develop and train the Deep Learning models for Object classification, detection and tracking
5. Creating logic and algorithm for drones to autonomously track the vehicles in real-world scenarios

The scope of current work is limited to

1. The implementation of the algorithm was done on a sample remote control sports car, before testing the algorithm on real life vehicles
2. Conduct a comprehensive comparison between deep learning techniques and algorithms

1.3 Contributions of the Thesis

The important contribution of this work are as follows

1. A chronological order to achieve proper and accurate autonomous vehicle tracking for a drone.
2. Significance of machine vision algorithms in target acquisition.
3. A novel method to prepare datasets for object classification, detection and tracking tasks in deep learning
4. A novel approach for maintaining continuous track on locked target

1.4 Summary

In this chapter, a brief introduction to image processing, classification, detection, tracking and the need for using an autonomous-based system for the tracking or following the target. It outlines the studies key objectives and presents an overview of the thesis organization, while also highlighting the significant contributions of this study.

CHAPTER 2 REVIEW OF LITERATURE

2.1.1 General

Drone integration has altered established procedures in a number of businesses by providing previously unheard-of capabilities for airborne monitoring, reconnaissance, and surveillance. Autonomous vehicle tracking has emerged as one of the most interesting uses of drone technology in recent years. Researchers and practitioners have investigated novel approaches to tracking vehicles with high precision and efficiency by utilizing the agility and versatility of drones.

This chapter seeks to provide a thorough overview of the state-of-the-art approaches, technologies, and breakthroughs in drone-based autonomous vehicle tracking by synthesizing and analyzing pertinent studies.

Advances in computer vision and aerial robotics, along with the widespread use of deep learning techniques, have made significant progress in autonomous vehicle tracking possible. To improve the precision, resilience, and real-time performance of vehicle tracking systems, researchers have investigated a wide range of strategies, from conventional object identification techniques to complex deep learning architectures.

Important subjects like feature extraction methods, motion estimation approaches, object detection algorithms related to drone-based autonomous vehicle monitoring will be covered in detail in this chapter on the literature review. It will also look at performance benchmarks, evaluation measures, and benchmark datasets that are frequently used to evaluate how well tracking algorithms work. This chapter will also highlight significant obstacles and restrictions that current methods have to deal with, opening the door to potential directions for further study and invention. This research seeks to further contribute in autonomous vehicle tracking technology and its useful applications in a range of fields, such as urban planning, security, and transportation, by critically analyzing the advantages and disadvantages of existing approaches.

Aerial robotics and computer vision algorithms interact intricately, and this is the foundation of autonomous vehicle tracking. Deep learning architectures that are specifically designed for object detection and tracking in aerial data have become possible because to advancements in traditional methods that rely on feature extraction and motion estimates. The research is full with many ways to increase the precision and effectiveness of vehicle tracking utilizing drones, ranging from region-based convolutional neural

networks (CNNs) to single-shot detectors (SSDs) to instance segmentation models.

Furthermore, this chapter's literature review provides an extensive examination of the most recent approaches and state-of-the-art methods for tracking autonomous vehicles with drones. We have discovered the amazing progress achieved in this field—fuelled by developments in computer vision algorithms, and machine learning—through a methodical examination of the literature.

2.2 Moving Object Detection

COMBINED DMM AND SUED

Various methods have been suggested for motion-based object detection, such as long-term motion pattern analysis, contextual information, parallax filtering, and global illumination adjustment [5]. These methods seek to detect and follow moving objects in aerial photography by utilizing several facets of motion and scene context.

Motion-based object detection techniques now in use have a number of drawbacks, including a reliance on brightness variations, trouble processing scenes with significant parallax, and a lack of resilience when it comes to identifying moving objects [5]. These restrictions draw attention to the difficulties that come with object detection in dynamic aerial situations, where picture complexity and lighting can change dramatically.

A few of the shortcomings of current techniques are addressed by the suggested method in the research, which combines the dynamic motion model (DMM) and the segmentation methodology SUED. It successfully and accurately detects moving objects in UAV aerial photos with low false alarm rates [1]. The suggested approach takes advantage of complimentary characteristics to improve overall detection performance by incorporating both DMM and SUED.

Targeted segmentation of particular areas for moving object identification is made possible by the suggested DMM, which offers efficient search windows based on pixel intensity. This targeted strategy focuses on regions that are likely to contain moving objects, which helps lower computational overhead and increase the efficiency of the detection process. By taking into account the distribution of pixel intensity and minimizing false detections from background objects with similar colours, the SUED segmentation technique, which employs edge-based dilation, efficiently recovers moving objects. SUED improves

segmentation by introducing edge-based dilatation, producing detection results that are more precise and trustworthy.

The efficacy and validity of the suggested methodology in identifying moving objects in UAV aerial photos are supported by experimental results [1]. By means of thorough assessment and verification, the suggested approach demonstrates its aptitude for precisely identifying and following moving objects in actual aerial situations, confirming its usefulness and resilience.

YOLO

The use of YOLOv4 and DeepSORT algorithms for vehicle recognition and tracking in traffic management, exhibits efficacy and efficiency. In particular, the YOLOv4 model is very accurate at identifying cars, but the YOLOv4-tiny model is faster and more accurate than the others, making it ideal for real-time surveillance on highways or in traffic management situations. Additionally, the system's performance can be further enhanced by utilizing cloud computing for quicker processing and Raspberry Pi integration for smaller-scale deployments. These improvements improve the system's scalability and adaptability in a variety of operating settings.

Using a heterogeneous dataset of 70,000 photos, the system is trained and assessed using the Darknet neural network architecture. By means of rigorous assessment criteria such as recall, mean

Average Precision (mAP), average Intersection over Union (IoU), and precision, the research shows significant improvements in detection and classification performance. In particular, the system works well for reliably recognizing and classifying different road users in difficult situations, which protects drivers and passengers alike.

2.3 Feature extraction

SIFT

Because it can identify and describe unique local characteristics in images, the Scale Invariant Feature Transform (SIFT) algorithm has attracted a lot of attention and is a good fit for tasks involving object recognition and tracking.

Camera-equipped drones are useful instruments for monitoring, surveillance, and reconnaissance tasks when the capacity to track targets in real-time independently is crucial. By enabling drones to recognize and track things across a variety of weather

conditions and viewing angles, SIFT provides a viable solution for these tasks. Numerous experiments have shown that SIFT is effective when used with drones to track targets.

Scale Invariant Feature Transform (SIFT) method, which is well-known for its capacity to recognize and describe distinctive local features in images. The unique quality of SIFT is that it remains highly adaptable to variations in illumination, scale, and rotation, which makes it ideal for a broad range of applications [1]. Its capacity to extract SIFT descriptors—which are necessary for tasks like object detection, matching, and image stitching—is its main feature.

The Python library of OpenCV is used in practice to construct the SIFT method, which makes feature matching between a reference image of the sample target and frames taken from a video feed easier. Setting up the SIFT algorithm and loading the reference image is the first step in the execution procedure.

In summary, the Scale Invariant Feature Transform (SIFT) method has potential for drone-based autonomous target tracking, but its computational complexity might make it unsuitable for real-time applications on platforms with limited resources. To overcome these obstacles, future research endeavors ought to concentrate on refining SIFT-based algorithms or investigating substitute feature extraction methods.

BAG OF FEATURES (BoF)

For feature extraction and object recognition applications, the Bag of Features (BoF) framework has become well-known. BoF is a powerful method for compactly and discriminatively encoding the visual content of images, which makes it ideal for use in applications like target tracking in drone-captured aerial imagery.

Among the many benefits that BoF provides is its capacity to record the distribution of local image descriptors, which makes target recognition reliable and effective under a variety of environmental circumstances.

Numerous research works have exhibited the suitability of BoF for drone-based target tracking. For instance, in tracking cars in aerial photos using BoF representations, allowing for autonomous traffic flow and congestion monitoring. Similar to this, BoF-based feature matching was used to track people in outdoor settings, aiding in search and rescue efforts.

BoF offers a strong framework for feature extraction and representation, but other considerations like the size of the codebook and the selection of local descriptors may have an impact on how well it performs. Furthermore, BoF might not be able to handle changes in viewpoint, rotation, or scale, all of which are significant problems in target tracking circumstances.

To improve the discriminative capacity of BoF representations, for example, more discriminative local descriptors like Scale Invariant Feature Transform (SIFT) or Speeded-Up Robust Features (SURF) might be used. Furthermore, methods like spatial pyramid matching and spatial verification can enhance BoF-based object recognition's spatial consistency.

In summary, the Bag of Features (BoF) framework presents a strong option for feature extraction and object recognition tasks, and it shows potential for autonomous target tracking with drones. Subsequent investigations ought to concentrate on refining BoF-based algorithms and investigating sophisticated methods to tackle the difficulties linked with target tracking in aerial photography.

In summary, the Bag of Features (BoF) framework presents a strong option for feature extraction and object recognition tasks, and it shows potential for autonomous target tracking with drones. Subsequent investigations ought to concentrate on refining BoF-based algorithms and investigating sophisticated methods to tackle the difficulties linked with target tracking in aerial photography.

Multi-scale Training and Testing

The study emphasizes the value of multi-scale training and testing approaches and shows how well they work to raise performance levels overall. Models are better able to catch differences in object sizes and looks when they take into account objects at different scales. This improves detection accuracy in a variety of settings.

2.4 Innovative Method in Object Detection

Methods Based on Region Proposals

The study emphasizes the advantages of region proposal-based techniques over regression-based ones, including Faster R-CNN and R-FCN. Compared to straight regression techniques, these approaches use region proposal networks to produce

candidate object regions, allowing for more accurate localization and detection.

Complementary Information Integration

Accurate object localization is found to be significantly dependent on the integration of supplementary information from related tasks. Object detection models can achieve higher localization accuracy by utilizing data from tasks like semantic segmentation and instance segmentation, which provide a deeper understanding of object boundaries and contexts.

Deep Neural Networks (DNNs)

The study highlights the notable improvements in object detection accuracy that may be obtained by implementing DNNs. Because DNN-based models can learn complicated hierarchical representations straight from raw data, they perform better than previous methods and allow for more reliable and precise item detection.

Multi-feature Boosting Forest and Multi-scale Adaptation

In conclusion, the study outlines the object detection application domains, including multi-feature boosting forest and multi-scale adaptation. By utilizing ensemble learning techniques and adaptivity to various scales, respectively, these approaches improve object identification systems' accuracy and resilience in practical situations.

RCNN

The region-based convolutional neural network, or R-CNN, is a two-stage early object detection model. It creates region proposals first, then takes each one, extracts its properties, and uses a CNN to categorize them. Although it was a breakthrough in object detection, the two-stage procedure makes it slow.

Spatial Pyramid Pooling Network, or SPP-net:

By adding spatial pyramid pooling, SPP-net increases the speed of R-CNN and enables it to process images of different sizes without scaling. It can now process photos more quickly thanks to this, which is especially helpful for object detection in settings where objects have varying scales.

Fast R-CNN

By minimizing repetitive calculations and sharing feature extraction computation across all region suggestions, Fast R-CNN considerably expedites object detection. Additionally, a Region of Interest (RoI) pooling layer is introduced, from which fixed-size feature maps are extracted for every proposal.

By including region proposal creation straight into the network design, Faster R-CNN increases speed and accuracy. In order to facilitate end-to-end training and speedier inference, it presents a Region Proposal Network (RPN) that shares convolutional features with the detection network.

Region-based Fully Convolutional Networks (R-FCN)

R-FCN is a region-based object detection model that makes use of position-sensitive score maps to predict object bounding boxes. It increases efficiency by sharing computation across spatial locations within each region proposal.

Feature Pyramid Network (FPN)

FPN builds a feature pyramid from a single convolutional network to address the problem of detecting objects at various scales and resolutions. This improves performance. particularly for small objects.

Mask R-CNN

Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks alongside bounding box detection. This enables it to perform instance segmentation, delineating individual object instances within a scene.

DSSD (Deconvolutional Single Shot Detector)

Using deconvolutional layers, this single-shot object recognition model improves feature maps and produces more precise object detections. It seeks to strike a balance between precision and speed.

DSOD

Deeply Supervised Object Detection (DSOD) is a model for deeply supervised object detection that includes supervision throughout the feature extraction process. This improves object detection performance by facilitating more efficient learning of hierarchical characteristics

2.5 Findings from Literature Review

After performing extensive literature review on various aspects of object detection, classification, tracking and associated techniques following observations are made:

•Multi-scale training and testing improves object detection performance.

•Region proposal-based techniques, such as Faster R-CNN and R-FCN, outperform regression-based methods.

•Integration of complementary information from related tasks enhances object localization accuracy.

•Deep Neural Networks (DNNs) show significant improvements in object detection accuracy compared to traditional approaches.

•Multi-feature boosting forest and multi-scale adaptation enhance object detection systems' accuracy and resilience.

2.6 Summary

The literature review encompasses various methodologies and techniques in object detection and tracking. It reveals the effectiveness of combining the dynamic motion model (DMM) and SUED for motion-based object detection, addressing limitations in existing methods. YOLOv4 and DeepSORT algorithms show promise in vehicle detection and tracking for traffic management, with YOLOv4-tiny offering fast processing suitable for real-time surveillance. Additionally, the Scale Invariant Feature Transform (SIFT) algorithm demonstrates potential for autonomous target tracking with drones, while Bag of Features (BoF) framework proves effective for feature extraction and recognition tasks. The Global Context Convolutional Network

(GCCN) enhances traditional CNNs by incorporating global context information, and other findings highlight the benefits of multi-scale training, region proposal-based techniques, and deep neural networks in improving object detection accuracy. Overall, the literature review provides valuable insights for further research and development in the field of object detection and tracking.

CHAPTER 3

PROBLEM OVERVIEW AND PROPOSED METHODOLOGY

3.1 Problem Identification

The comprehensive literature review provided in the previous sheds light on the complex challenges in tracking an object especially vehicles while they are travelling in a high traffic region or in inaccessible areas. Traditional methods have their own cons of occlusion handling and multiple targets detection etc.

In high-density urban environments or areas with intricate landscapes, the ability to maintain continuous and accurate tracking of a target becomes even more critical. Vehicles can change speed, direction, and even disappear from view due to factors like buildings, foliage, or other obstructions. These factors create a demanding environment where standard tracking algorithms struggle to maintain reliable tracking over extended periods.

The need for a robust system capable of handling these challenges is evident. The system should not only detect and track the target accurately but also be resilient to occlusion and other adverse conditions commonly encountered in real-world scenarios. Additionally, the system should be efficient in terms of computational resources, allowing for real-time performance without compromising accuracy.

Given the challenges mentioned above in autonomous tracking systems, the problem statement for the current research work is presented in the subsequent section.

3.2 Problem Formulation

The problem at hand involves addressing multiple challenges in autonomous tracking systems.

Firstly, developing a deep learning model capable of classifying, accurately detecting and properly locating vehicles in the video frames of the drone, specially focusing on peculiar scenarios. Developing an algorithm which inputs the training weights of the above deep learning algorithms to detect and keep a track on the target and helps the drone to continuously follow the vehicle through its live feed.

3.3 Proposed Methodology

The proposed methodology to address the outlined problem involves a multi-stage process that integrates machine vision algorithms and deep learning techniques. The key steps are as follows:

1. Study and observe pre-trained model and their results
2. Target locking using unique features machine vision techniques
3. Design and train Object classification model (CNN)
4. Prepare datasets then design and train object detection model
5. Design and train a feature extraction model
6. Insert the training weights in YOLOv8 tracker algorithm to track the vehicle
7. Deploy the software into drone's live feed
8. Test various trackers using the trained weights
9. Self-tracking logic and algorithm

Above steps are the designed methodology for achieving the final tracking algorithm of autonomous target tracking, the work is done in two phases. Phase one consists of literature survey, study and observations and complete machine vision techniques used for locking the target. The phase 2 consists of Deep Learning part of classification, detection and tracking the target.

3.4 Summary

This chapter delineates the identification of the problem for the research work and briefly describes the problem statements. The implementation of the first step.

CHAPTER 4
HUMAN FACE DETECTION & TRACKING USING OpenCV AND PYTHON

4.1 Introduction

For the basic understanding of image processing, machine vision and tracking algorithms we considered solving a simple problem which is Human face tracking using the drone with the help of Python and OpenCV framework which is used vastly in image and video processing field by experts.

For the face detection task "Haar cascades" from OpenCV was used, The most efficient human face detection technique in image processing. Haar cascades are a machine learning-based approach used for object detection. The concept is based on the Haar wavelet technique, which analyzes regions of an image and detects changes in intensity patterns. These changes can represent edges, corners, or other features that define objects. Haar cascades works in the following manner

1. Haar Features: Haar features are simple rectangular filters that detect changes in pixel values in different regions of an image. These features are similar to convolutional kernels and are capable of capturing different aspects of an object's appearance, such as edges and textures.

2. Integral Image: To speed up computation, Haar features are computed using an integral image. The integral image allows for instant calculation of the sum of pixel values in any rectangular area of an image.

3. Training: Haar cascades are trained using a large dataset of positive and negative examples. Positive examples are images containing the object of interest (e.g., faces), while negative examples are images without the object. The cascade training algorithm then learns to distinguish between positive and negative examples by selecting the most discriminative Haar features.

4. Cascade Classifier: The trained Haar cascade classifier consists of multiple stages, each containing a subset of Haar features. At each stage, the classifier evaluates whether the region of interest contains the object based on the presence or absence of certain features. If the region passes all stages, it is classified as containing the object.

Sliding Window Approach: During the detection phase, the Haar cascade classifier is applied using a sliding window technique. A window of fixed size is moved across

the image at different scales, and at each position, the classifier determines whether the region within the window contains the object.

After detecting the face and drawing the bounding box using OpenCV's Haar Cascades, further step is to achieve continuous tracking of the drone along with the target. For this we designed an arithmetic algorithm and named it "Adaptive Proportional – Yaw Control (APYC)".

4.2 Adaptive Proportional – Yaw Control (APYC)

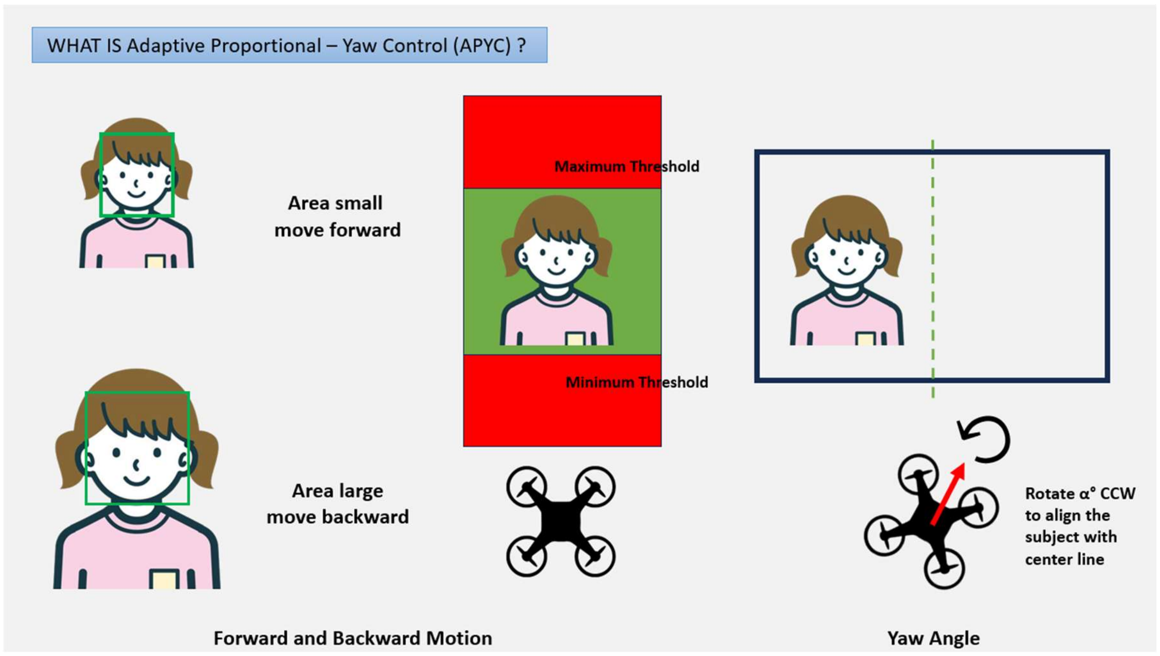

WHAT IS Adaptive Proportional – Yaw Control (APYC) ?
Area small move forward
Maximum Threshold
Minimum Threshold
Area large move backward
Rotate α° CCW to align the subject with center line
Forward and Backward Motion
Yaw Angle

Figure 4. 1 Overview of APYC algorithm

The APYC algorithm follows the following logic, for forward and backward motion there are two parameters named maximum threshold value and minimum threshold value, which are simply the maximum and minimum areas of square shaped bounding boxes that are generated using Haar cascades.

In the code it is given that if the area of the bounding box is more than some specific number (unit2) it will be concluded by the algorithm that the targeted object is very near to the camera and the drone should move backwards to prevent collision with the person and

this area is saved as maximum threshold value, vice versa for minimum threshold value case the drone moves forward to keep the bounding box in the mid region of maximum and minimum threshold. For a better understanding please look at the figure 4.1

For maintaining the yaw angle the APYC algorithm follows the following principle, to always keep the target at the middle of the frame throughout the drone's journey of tracking the person. For this, a center line was taken which exactly bisects the drone's video feed, now the centroid of the bounding box obtained using Haar cascades should lay on this center line which ultimately turns the drone (changes its yaw angle) in order to bring the target to the middle of the frame.

4.3 Summary

To summarize this chapter, for object detection Haar cascades from OpenCV were used in human face detection and for tracking a logical algorithm which named by us as Alternate Proportional – Yaw Control (APYC) was used. Also, the results were clear and the DJI Tello drone was able to track the targeted person throughout his/her movement.

CHAPTER 5

OBJECT CLASSIFICATION, DETECTION AND SEGMENTATION USING YOLOV8

5.1 Introduction

The fields of autonomous cars have come to rely heavily on object recognition, classification, and segmentation capabilities in the constantly changing field of computer vision. The creation of reliable algorithms that can distinguish and identify things in photographs is essential to allowing machines to understand and communicate with their environment. You Only Look Once (YOLO) is an approach that stands out among the many others due to its efficiency and real-time performance.

Two-stage algorithms and one-stage algorithms are the two main groups into which object detection algorithms fall. Two-stage techniques usually involve a region proposal network followed by object identification; models such as Fast R-CNN, Faster R-CNN, and Mask R-CNN [17-19] are examples of such models. However, object detection is done in a single step using one-stage algorithms, which include well-known frameworks like the You Only Look Once (YOLO) family [13,20–24] and single shot multi-box detector (SSD) methods [25,26].

The YOLO series is one of these algorithms that has had substantial progress and is considered one of the best methods available. In instance, the 2023-introduced YOLOv8 algorithm [27] has proven to be remarkably accurate, outperforming its predecessors.

Building on the ideas of its predecessors, the YOLOv8 architecture seeks to increase object detection tasks' accuracy and speed. It uses a single neural network to predict bounding boxes and class probabilities for several objects in an image at the same time. This means that YOLOv8 processes the entire image in a single pass, which makes it faster and more efficient than running numerous models or algorithms one after the other.

YOLOv8 integrates a number of deep learning innovations, including improved object detection methods, training methodologies, and more complex backbone networks (like Darknet). Better detection performance is a result of these enhancements for a variety of object classes and contexts.

In this chapter, the use of YOLOv8, a cutting-edge version of the YOLO architecture, to improve object recognition, classification, and segmentation is explored. Even while YOLO has performed remarkably well on common datasets, adapting it to new datasets brings with it special potential and challenges. Custom datasets frequently include features unique to their domain, hence customized methods are required to achieve the best results.

Examining YOLOv8's effectiveness in handling the complexity of unique datasets for tasks like object identification, classification, and segmentation is the main goal of this study. In order to push the limits of accuracy and efficiency in practical circumstances, this study makes use of YOLOv8's architecture and training approaches. In order to optimize performance on unique datasets.

All things considered, this chapter provides a thorough investigation of object recognition, classification, and segmentation on customized datasets using YOLOv8.This chapter intends to add to the wider discussion on the practical deployment of deep learning models in computer vision applications by assessing the trade-offs between accuracy, speed, and processing resources.

5.2 Overview of the Methodology

1.Training YOLOv8 Model:

Data Collection: Assemble a varied dataset of pictures or videos pertinent to the drone autonomous navigation challenge, with an emphasis on situations with numerous cars, occlusion, and different distances from the camera.

Data Annotation: Mark objects of interest (like cars) with bounding boxes and provide associated class labels (like car).

Preprocessing: Incorporate methods like as rotation, scaling, and flipping to strengthen and diversify the dataset.

Model Configuration: Configure the YOLOv8 architecture with the right values for input size, class count, and anchor boxes.

Training: With a large-scale GPU infrastructure, train the YOLOv8 model on the annotated dataset, optimizing for both speed and accuracy.

Evaluation: Validate the trained model on a separate validation set to assess its performance metrics, such as precision, recall, and mean average precision (mAP).

2.Using an Algorithm for Detection, Segmentation, and Classification:

Object Detection: In real-time photos or videos that the drone takes, use the trained YOLOv8 model to identify cars and other pertinent things.

Semantic Segmentation: Create pixel-level masks for each object by using segmentation algorithms to separate the identified objects from the background.

Classification: Using deep learning classifiers or conventional machine learning techniques, classify the segmented objects into predetermined categories (e.g., car, lorry) based on their visual attributes.

3.Occlusion Handling

Implement techniques to mitigate occlusion effects and improve object detection accuracy

4.Distance from the Camera

Depth Estimation: Integrate depth estimation algorithms to estimate the distance of detected objects from the drone's camera.

5.Multiple Cars

Multi-Object Tracking: Implement multi-object tracking algorithms to track multiple cars simultaneously across frames, maintaining identity information for each car.

Collision Avoidance: Incorporate collision avoidance strategies based on the positions and trajectories of multiple detected cars, ensuring safe navigation of the drone in dynamic environments.

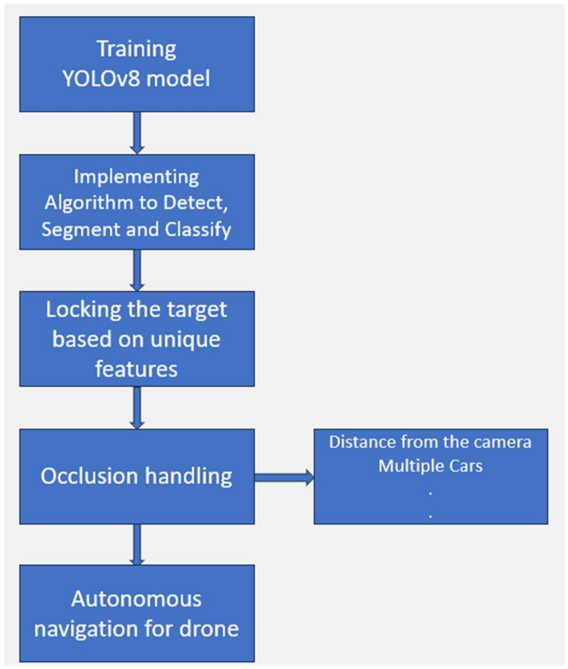

Training YOLOv8 model
Implementing Algorithm to Detect, Segment and Classify
Locking the target based on unique features
Occlusion handling
Distance from the camera
Multiple Cars
.
.
Autonomous navigation for drone

Figure 5. 1 Overview of the proposed approach

5.3 Methodology of code

A central entry point orchestrates the workflow at the start of the object detection, segmentation, and classification pipeline's code structure. In order to facilitate a seamless execution, dependencies are installed beforehand, and requirements.txt is used to maintain versioning. The selection of datasets is crucial, and the codebase has preprocessing stages to prepare the data for training and assessment, as well as systems for standard and bespoke datasets.

One of the main components is training the YOLOv8 model using deep learning frameworks like TensorFlow or PyTorch. Training protocols are meticulously monitored, with checkpoints recorded for later usage and metrics tracked. Incoming data is handled by input processing modules, which format, resize, and normalize it in accordance with model specifications. Various application scenarios are accommodated with options for batch processing or real-time streaming.

The output module creates annotated photos with labels, bounding boxes, and masks to visually represent the results. Real-time monitoring and analysis are facilitated by options for streaming or storing output. The codebase places a strong emphasis on modularity and clarity. To facilitate experimentation and expansion, components are arranged into distinct modules or classes for reusability and maintainability.

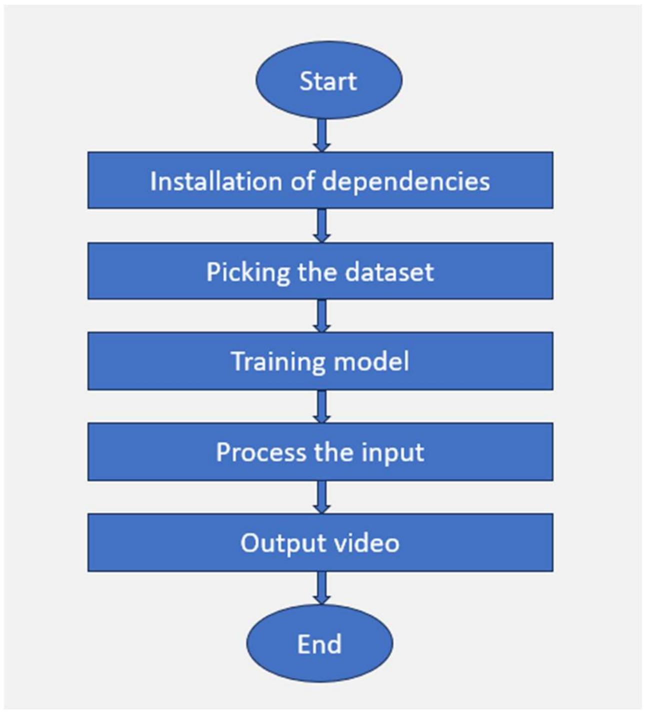


Figure 5.2 Structure of the code

5.4 Results

5.4.1 Processed video using pre-trained YOLOv8 model

The codebase contains functionality to perform detection, segmentation, and classification tasks on pictures using a pre-trained YOLOv8 model. The code first preprocesses each frame in the input video to make sure it complies with the model's input specifications. Next, each frame is subjected to the pre-trained YOLOv8 model, which identifies objects of interest, separates them from the background, and groups them into predetermined categories. To illustrate the findings, bounding boxes, segmentation masks, and class labels are superimposed onto the video frames. Post-processing methods like non-maximum suppression are used to improve the output quality and fine-tune the detections. The annotated frames are then combined to create a processed video that displays the items that were identified, divided into segments, and categorized during the input video's duration.

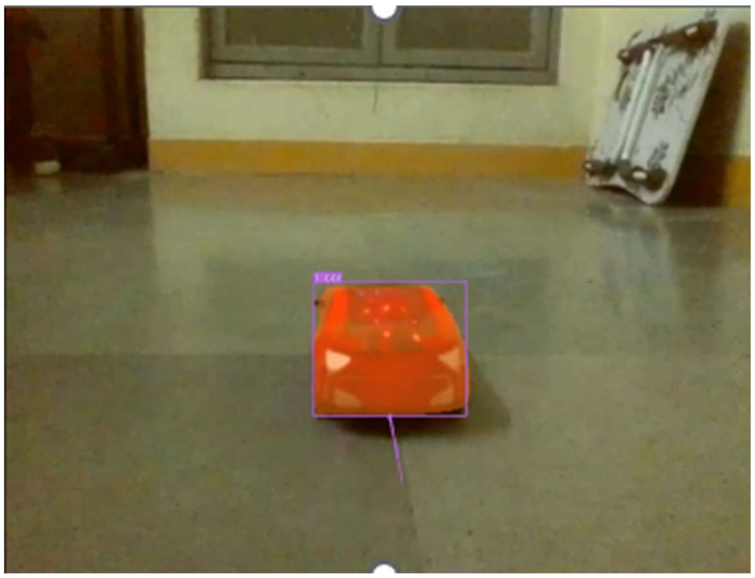
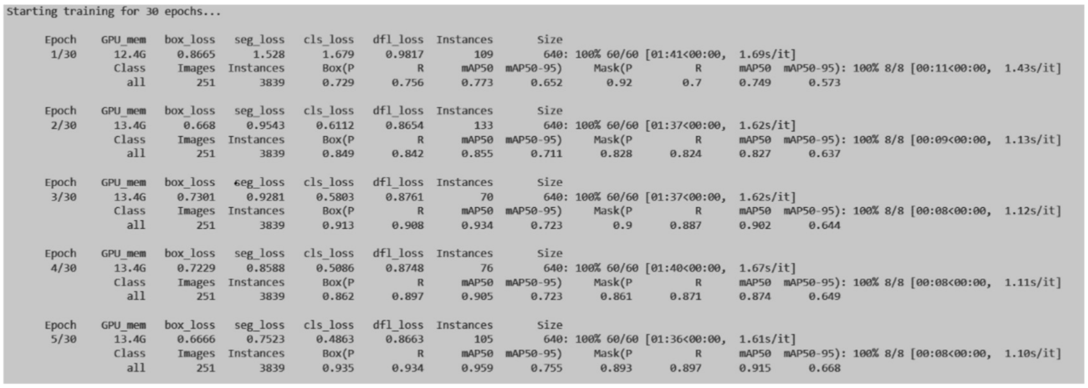


Figure 5.3 Processed output video obtained using pretrained yolov8 model

5.4.2 Model training using custom dataset

The number of epochs, which determines how many times the whole dataset is run through the neural network for training, was a critical parameter taken into account for the dataset's training. It is commonly known that additional epochs enable the model to learn more intricate patterns and subtleties within the data, which improves training and increases the model's accuracy or precision. The number of epochs was limited to 30 in this particular scenario, though, because of limitations imposed by computational resources, specifically the availability of GPU resources. Deep learning model training may be computationally demanding, needing a significant amount of GPU compute power and memory, especially for models as complicated as YOLOv8.

Achieving appropriate training outcomes while optimizing the consumption of available CPU resources is balanced by setting the number of epochs to thirty. Even if adding more epochs might enhance the model's performance even more, practical considerations and resource restrictions must be taken into account to guarantee effective training and experimental procedures.



```
Starting training for 30 epochs...

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       1/30      12.4G     0.8665      1.528      1.679     0.9817        109        640: 100% 60/60 [01:41<00:00,  1.69s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 8/8 [00:11<00:00,  1.43s/it]
                   all        251       3839      0.729      0.756      0.773      0.652       0.92        0.7      0.749      0.573

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       2/30      13.4G      0.668     0.9543     0.6112     0.8654        133        640: 100% 60/60 [01:37<00:00,  1.62s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 8/8 [00:09<00:00,  1.13s/it]
                   all        251       3839      0.849      0.842      0.855      0.711      0.828      0.824      0.827      0.637

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       3/30      13.4G     0.7301     0.9281     0.5803     0.8761         70        640: 100% 60/60 [01:37<00:00,  1.62s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 8/8 [00:08<00:00,  1.12s/it]
                   all        251       3839      0.913      0.908      0.934      0.723        0.9      0.887      0.902      0.644

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       4/30      13.4G     0.7229     0.8588     0.5086     0.8748         76        640: 100% 60/60 [01:40<00:00,  1.67s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 8/8 [00:08<00:00,  1.11s/it]
                   all        251       3839      0.862      0.897      0.905      0.723      0.861      0.871      0.874      0.649

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       5/30      13.4G     0.6666     0.7523     0.4863     0.8663        105        640: 100% 60/60 [01:36<00:00,  1.61s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 8/8 [00:08<00:00,  1.10s/it]
                   all        251       3839      0.935      0.934      0.959      0.755      0.893      0.897      0.915      0.668
```

Figure 5.4 Output of model training using custom dataset

5.4.3 Curves depicting losses

Making curves that show class loss, segmentation loss, and box loss in relation to the number of epochs provide important information about the YOLOv8 model's performance and training state. These curves are used as diagnostic tools to track the training process's convergence and spot possible problems like overfitting or underfitting. All three losses are usually large in the early epochs as the model gains the ability to identify and locate objects, accurately segment them, and categorize them. Ideally, as training goes on, the losses go down, showing that the model is getting better at certain tasks. On the other hand, variations or plateaus in the loss curves could point to problems with model convergence or difficulties understanding specific parts of the dataset.

In order to maximize model performance, these curves can be analysed and changes made to the training procedure, such as changing learning rates, adding regularization strategies, or expanding the dataset. For better object recognition, segmentation, and classification accuracy, the YOLOv8 model is refined by graphing box loss, segmentation loss, and class loss versus the number of epochs. This visualization offers important insights into the training dynamics.

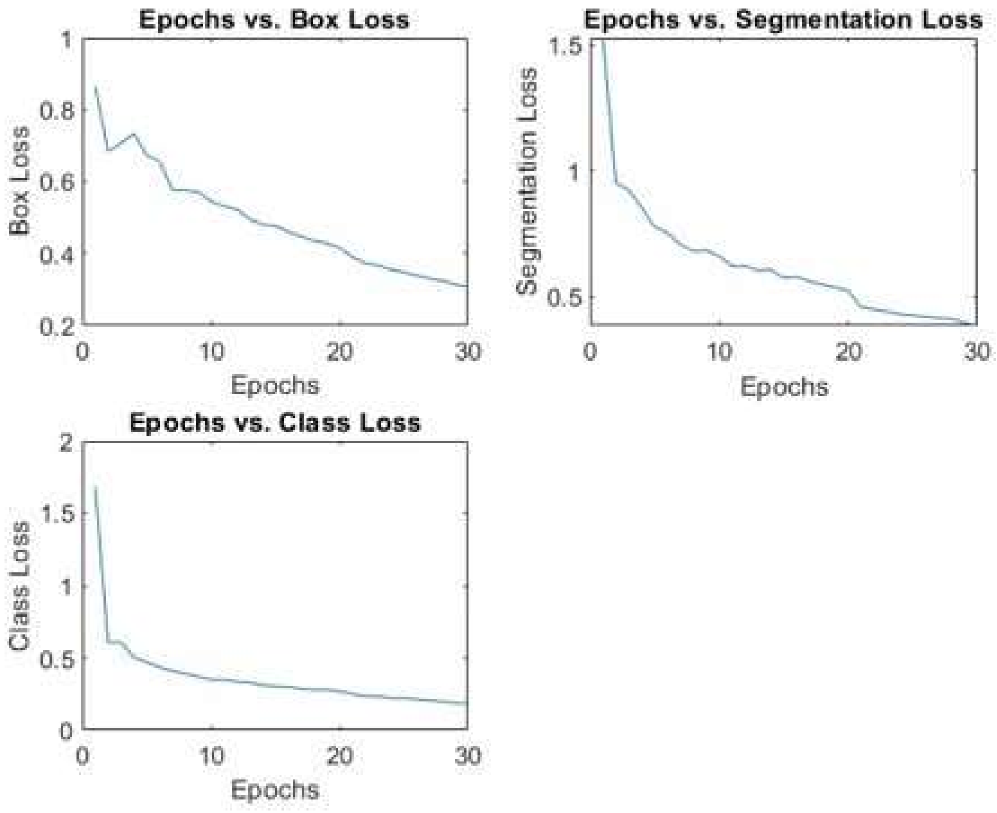


Figure 5.5 Curves depicting Box, Segmentation and Class Loss vs Number of Epochs

When the number of epochs is set to 30, the loss function converges at 0.8, indicating that the model has stabilized and is successfully minimizing its total loss on the training dataset. A loss value of 0.8 indicates that, on average, the model's predictions differ from the ground truth by a specific amount, which is deemed tolerable considering the task's complexity, the dataset's limitations, and the computational resources available.

Indeed, as the model has more chances to hone its parameters and absorb dataset knowledge, expanding the number of epochs usually results in an even greater reduction of the loss function. This finding is consistent with the way loss curves often behave during training: as the number of epochs rises, the model keeps modifying its weights to reduce the loss and enhance performance.

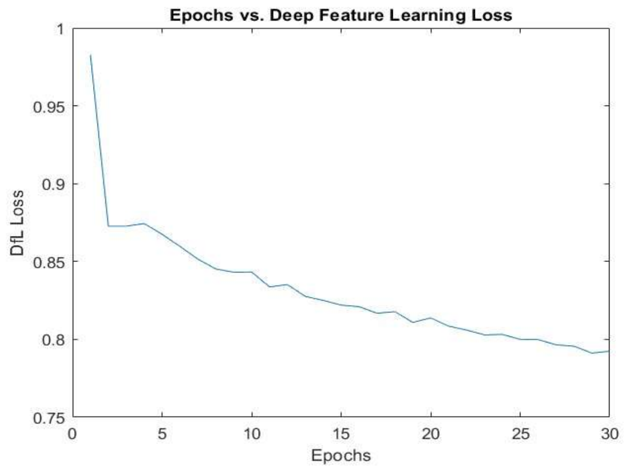


Figure 5.6 Curve depicting DFL Loss vs Number of Epochs

5.5 Summary

In summary, the detection of an RC model car has been successfully achieved with the use of a DJI Tello drone that is outfitted with a 5 MP camera that operates at 30 frames per second. Through the use of segmentation and labelling methods in conjunction with the YOLOv8 algorithm for training, the system was able to precisely identify and distinguish the remote-control car from the acquired photos or video frames. This achievement highlights how advanced computer vision techniques work well in dynamic settings like aerial surveillance.

The incorporation of YOLOv8, a recognized real-time performance and efficiency tool, allowed the model to learn and identify the remote-control car (RC car) in a variety of backgrounds and environmental circumstances that are experienced in drone operations. Furthermore, segmentation and labeling techniques enhanced the interpretability and utility of the detection results, facilitating precise tracking and analysis of the detected object.

Overall, the DJI Tello drone's ability to detect the RC model vehicle successfully shows the potential benefits of fusing advanced deep learning algorithms with cutting-edge technologies like drones. This synergy creates opportunities for improved capabilities in a variety of domains, including infrastructure inspection, agriculture, security, and defense. Future developments in drone technology and computer vision algorithms have the potential to completely transform remote sensing and aerial surveillance applications, spurring creativity and advancement across a wide range of industries.

CHAPTER 6

UNIQUE FEATURE EXTRACTION

6.1 Introduction

In Machine Vision, a unique feature refers to a piece of information extracted from an image that is particularly informative for a specific task, in our case target locking. It's like a characteristic detail that helps a machine vision system understand what it's seeing. There are several methods to extract unique features from images, two methods we chose for this task were Scale Invariant Feature Transform (SIFT), Bag of Features (BoF). Both the algorithms have their own benefits and works in different methods. These algorithms analyze local neighborhoods of pixels to find regions with unique characteristics, such as corners, edges, or blobs (also called as interest points).

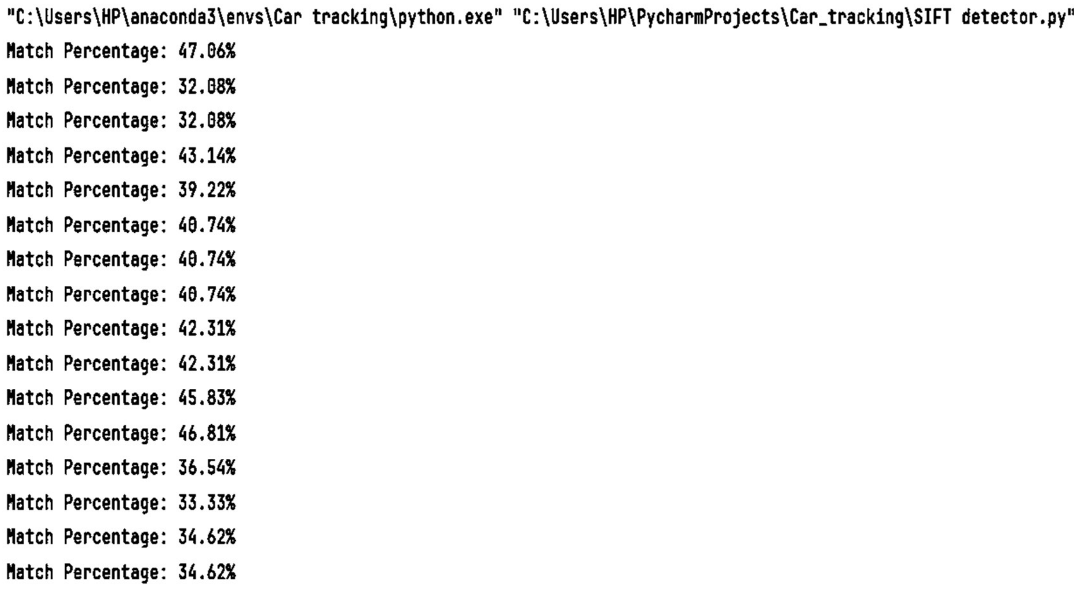
We also conducted a study that shows the comparison of the match percentage of the target with the reference image throughout the video feed while the target moves away from the camera between matching using Scale-Invariant Feature Transform (SIFT) feature points and matching using Bag of Features (BoF) histograms.

In our code we used RANSAC algorithm to increase the efficiency of the SIFT feature detector. The RANSAC (Random Sample Consensus) algorithm has been used to improve the efficiency of SIFT (Scale Invariant Feature Transform) by removing mismatches in feature point matching. The RANSAC algorithm is a method of robust estimation that can tolerate a large number of outliers in the data space and effectively deal with multiple structure data. By using an improved RANSAC algorithm, parts of the error feature points can be removed before estimating the model, leading to an increase in the proportion of correct matching feature points and improving the efficiency of the SIFT algorithm. Experiments have shown that the improved RANSAC algorithm can find the model more accurately, improve efficiency, and make the feature point matching more accurate.

6.2 Scale Invariant Feature Transform (SIFT) Algorithm

The SIFT algorithm is an essential tool in computer vision, is excellent for identifying and characterizing unique local features in images that are unaffected by changes in illumination, scale or rotation. The strength of SIFT comes from its capacity to extract SIFT descriptors that are needed for a variety of applications, such as object detection,

matching, and image stitching Algorithm is executed using OpenCV's python library by feature matching between a reference image of the sample target and frames extracted from a video feed. Initially, the SIFT algorithm is set up, and the reference image is loaded, followed by its conversion to grayscale. Key points and descriptors are then extracted from the reference image using the SIFT algorithm. Later the code proceeds by opening the video file and extracting its properties like frame count, FPS, width, and height. Within the primary loop iterating through the video frames, each frame is converted to grayscale, and SIFT key points and descriptors are extracted. These descriptors are then matched with those from the reference image using a Brute-Force Matcher. The matches are filtered through RANSAC (Random Sample Consensus) to find geometrically consistent matches and the match percentage is calculated based on the number of good matches found.



```
"C:\Users\HP\anaconda3\envs\Car tracking\python.exe" "C:\Users\HP\PycharmProjects\Car_tracking\SIFT detector.py"
Match Percentage: 47.06%
Match Percentage: 32.08%
Match Percentage: 32.08%
Match Percentage: 43.14%
Match Percentage: 39.22%
Match Percentage: 40.74%
Match Percentage: 40.74%
Match Percentage: 40.74%
Match Percentage: 42.31%
Match Percentage: 42.31%
Match Percentage: 45.83%
Match Percentage: 46.81%
Match Percentage: 36.54%
Match Percentage: 33.33%
Match Percentage: 34.62%
Match Percentage: 34.62%
```

Figure 6.1 Output terminal displaying the match percentage of SIFT feature points between the reference image and the video feed of the sample target i.e. remote-control car.

Discussing about the output, the code visually displays the matching points between the reference image and the video feed in a window or a terminal as shown in 6.2. The printed match percentage in Figure 6.1 represents the proportion of successful matches between the reference image and the video frames. The code also includes a check of minimum number of matches required for RANSAC to proceed. If the minimum match count is not met, a message stating insufficient matches is printed.

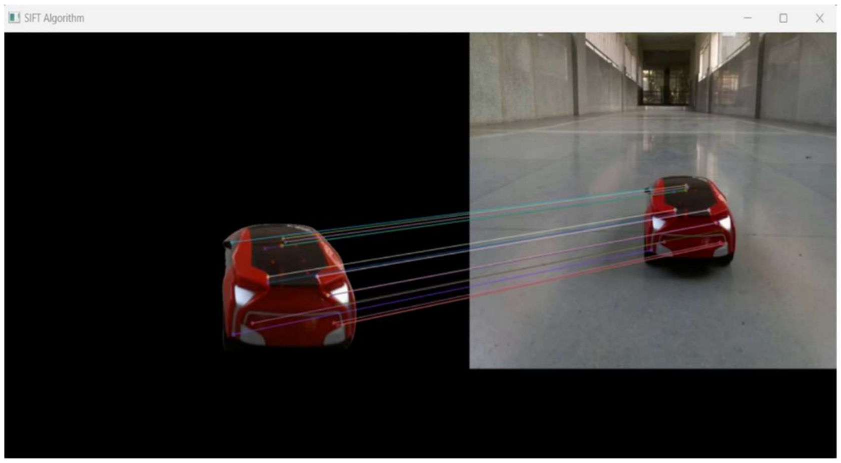


Figure 6.2 Output terminal displaying how SIFT feature points being matched between reference image and frames of the video feed

The drawback of using SIFT unique feature points alone to match the reference image and the video feed is that the match percentage will be reduced as the target stays farer from the camera, fig. 3 can be helpful in understanding this.

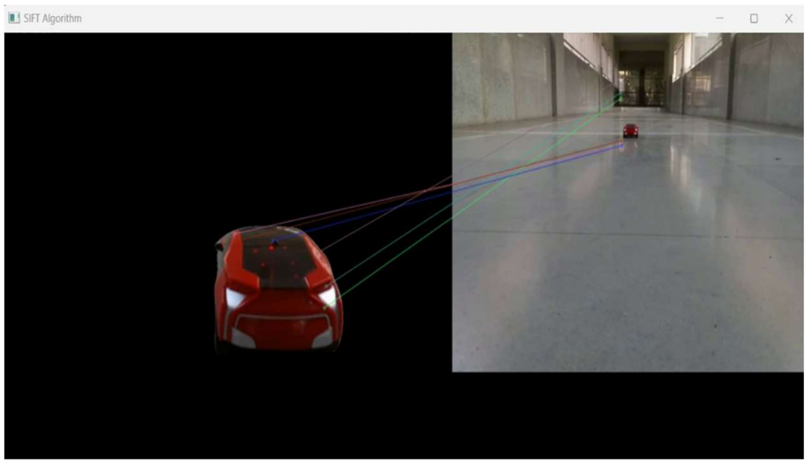


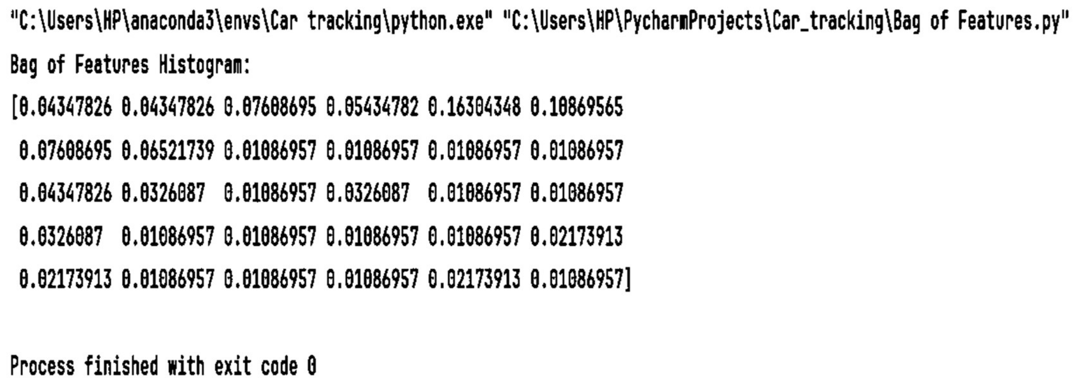
Figure 6.3 Mismatch between the SIFT feature points as the target stays farer from the camera.

As a result, the match percentage of this algorithm drastically falls down as the distance of the target kept increased.

6.3 Bag of Features (BoF) Algorithm

The Bag of Features (BoF) algorithm is a technique commonly used in computer vision and image processing for feature representation and analysis. It is driven by text document processing, which transforms words into a numerical representation (word bag). Similar to this, local features in images are retrieved from regions of interest, such as descriptors and key points of the Scale-Invariant Feature Transform (SIFT). BoF uses several steps to produce a condensed representation of these local traits.

The code we implemented operates on a reference image using the Bag of Features (BoF) technique to generate histograms. Initially, the reference image is loaded and converted into grayscale. The SIFT algorithm is employed to detect key points and compute descriptors for this grayscale image. A FLANN-based matcher is set up for future use in matching descriptors efficiently. Subsequently, k-means clustering is performed on the descriptors obtained from the reference image, where 'k' is chosen as the minimum value between 30 and the number of available descriptors (since in some cases the available descriptors may be less than 30). The code then assigns each descriptor to a cluster center and constructs a histogram by counting the frequency of descriptors associated with each cluster. Afterward, the histogram is normalized to ensure that its values lie within the range of 0 to 1. Finally, the code prints the Bag of Features histogram computed from the reference image descriptors, showcasing the distribution of features across the clusters.



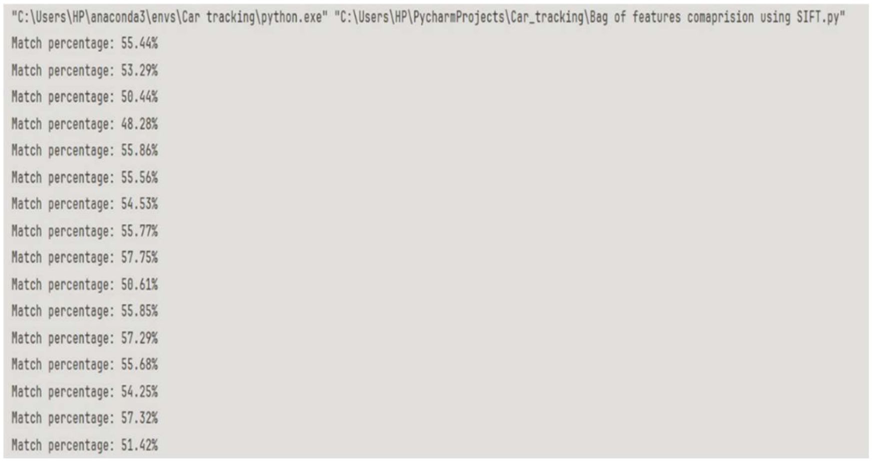
```
"C:\Users\HP\anaconda3\envs\Car tracking\python.exe" "C:\Users\HP\PycharmProjects\Car_tracking\Bag of Features.py"
Bag of Features Histogram:
[0.04347826 0.04347826 0.07608695 0.05434782 0.16304348 0.10869565
 0.07608695 0.06521739 0.01086957 0.01086957 0.01086957 0.01086957
 0.04347826 0.0326087  0.01086957 0.0326087  0.01086957 0.01086957
 0.0326087  0.01086957 0.01086957 0.01086957 0.01086957 0.02173913
 0.02173913 0.01086957 0.01086957 0.01086957 0.02173913 0.01086957]

Process finished with exit code 0
```

Figure 6.4 Bag of Features Histogram

Figure 6.4 represents the output of the code which contains a histogram containing 30 entries, which represent the features of our sample target. Each value in the histogram signifies the relative frequency or importance of descriptors associated with a particular

cluster center. This histogram quantitatively represents the visual characteristics and distribution of features present in the reference image based on the BoF algorithm, providing a condensed representation useful for various computer vision tasks such as image classification, object recognition, and content-based image retrieval.

Later part of our code implements a comparison between the Bag of Features (BoF) histogram of a reference image and the histograms computed from frames extracted from a video feed. Initially, the reference image is loaded, converted to grayscale and SIFT key points and descriptors are computed then the histogram is generated. Subsequently, the code opens a video file, reads its frames, and for each frame, converts it to grayscale, detects key points, computes descriptors using SIFT, and performs k-means clustering to generate a histogram for the frame similar to the reference image. The Bhattacharyya distance, via cv2.compareHist(), measures the similarity between the histograms of the reference image and each video frame, yielding a match score. This match score is then converted to a match percentage, which represents the similarity between the reference image and each frame. The code displays the match percentage for each frame in the terminal and shows the video frames in a window.



```
"C:\Users\HP\anaconda3\envs\Car tracking\python.exe" "C:\Users\HP\PycharmProjects\Car_tracking\Bag of features comaprision using SIFT.py"
Match percentage: 55.44%
Match percentage: 53.29%
Match percentage: 50.44%
Match percentage: 48.28%
Match percentage: 55.86%
Match percentage: 55.56%
Match percentage: 54.53%
Match percentage: 55.77%
Match percentage: 57.75%
Match percentage: 50.61%
Match percentage: 55.85%
Match percentage: 57.29%
Match percentage: 55.68%
Match percentage: 54.25%
Match percentage: 57.32%
Match percentage: 51.42%
```

Figure 6.5 Output terminal displaying the match percentage of BoF histograms between the reference image and the video feed of the sample target

Above figure 6.5 helps us to understand that the match percentage of BoF algorithm is a little bit higher and consistent throughout the video frames. This concludes that BoF algorithm is having a track to the target even at larger distances compared to the SIFT algorithm alone.

6.4 Scale – Invariant Feature Transform (SIFT) vs Bag of Features (BoF)

The video feed that is used for this purpose contains 789 frames in total, match percentage in each frame is taken and visually represented as a plot below.

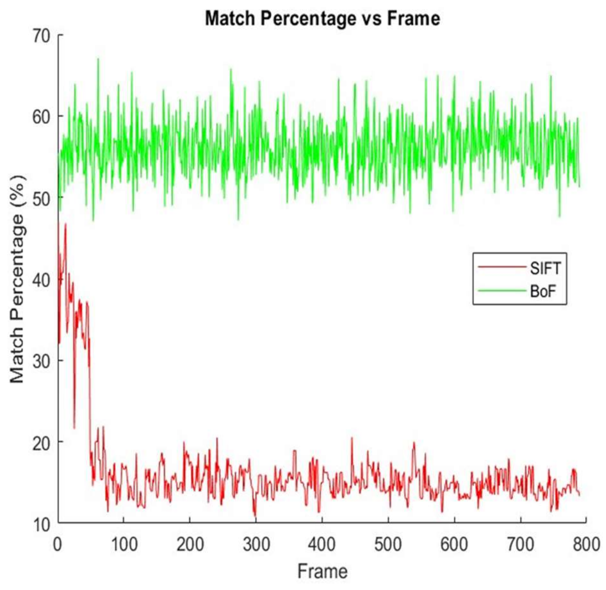


Figure 6.6 Match percentage (%) vs Frame Number

As we can clearly observe in fig. 6 that the match percentage in SIFT algorithm drastically fell down as the target moves far away in each frame of the video whereas while using the BoF algorithm the match percentage is almost in a constant range throughout all the frames.

This study finally concludes that BoF algorithm is highly suitable for our work, and hence we decided to proceed with BoF histograms for our deep learning model to lock the desired target.

Since, our project needs live time tracking unfortunately any of the above unique feature extraction method will be helpful while implementing our final algorithm. But this comparison work done by us surely will contribute to the field of machine vision and its applications.

6.5 Summary

This chapter explores two feature extraction methods for machine vision, particularly targeting a remote-controlled car in a video feed. The first method, Scale Invariant Feature Transform (SIFT), excels at identifying unique image features but suffers from reduced accuracy as the target moves away. The chapter details how SIFT works and its limitations.

The second method, Bag of Features (BoF), addresses this limitation. BoF builds a histogram representing the feature distribution in an image. By comparing histograms from the reference image and video frames, BoF achieves more consistent matching even with target distance variations. The chapter explains BoF's implementation and highlights its advantage over SIFT for this specific task. Ultimately, BoF's robustness to distance variations makes it the preferred choice for building a deep learning model for target locking.

CHAPTER 7

CNN FOR OBJECT CLASSIFICATION AND DETECTION

7.1 Introduction

Convolutional neural networks, or CNNs, have become a key component of computer vision technology in recent years, transforming applications like object identification and categorization. CNNs are now essential tools for deciphering complicated images and extracting relevant information because of their capacity to automatically learn hierarchical representations of visual data.

It is crucial to comprehend the foundational ideas and developments in this field in light of this chapter, which focuses on object recognition and classification using CNNs. This introduction provides a quick summary of CNNs and how they are used for tasks involving object recognition and categorization.

A class of deep learning models known as CNNs is motivated by the structure of the visual cortex in the human brain. They are made up of convolutional, pooling, and fully connected layers, among other layers of interconnected neurons. CNNs can automatically learn the spatial hierarchies of features from raw pixel data by means of convolution, pooling, and non-linear activation. This allows CNNs to identify objects and patterns within images.

CNNs' capacity to handle massive datasets and pick up intricate patterns without the need for manual feature extraction is one of their main advantages. Because of this, they work especially well at tasks like object recognition and classification, where the objective is to precisely locate and identify items in a picture.

CNN-based techniques have outperformed conventional methods in terms of accuracy and efficiency in object detection and classification tasks in recent years. The capabilities of CNNs are always being pushed to the limit, from cutting-edge models like Faster R-CNN, SSD, and YOLO to ground-breaking architectures like AlexNet and VGGNet.

In this chapter, we investigate several topologies, training approaches, and evaluation criteria for CNNs used in object recognition and categorization. Our goal is to create reliable and accurate systems that can identify and categorize things in various real-world

situations by utilizing CNNs. This will have consequences for autonomous driving, and surveillance. We aim to make a positive impact on the current state of CNN-based object detection and classification research, which will eventually lead to the development of more sophisticated and self-governing systems.

7.2 Overview of the Methodology

Build the CNN Model: Convolutional Neural Network (CNN) model architecture design and construction is the first phase in our process. The layers, such as convolutional, pooling, and fully connected layers, must be defined. Activation functions and input/output dimensions must also be specified. The architecture is customized to meet the unique needs of our object identification and classification task, accounting for elements like the target accuracy level and dataset complexity.

Compile the Model: Before training can start, the CNN model architecture must be built after it has been defined. During the model's compilation, the evaluation metrics, optimizer, and loss function that will be employed for training must be specified. The type of job (binary classification, multi-class classification, object detection, etc.) and the properties of the dataset determine which loss function and optimizer to use. Furthermore, assessment metrics including recall, accuracy, precision, and F1-score are chosen to evaluate the model's performance during training and assessment.

Train the Model: After the CNN model has been assembled, it must be trained using the training dataset. Through a process of forward and backward propagation, the model learns to identify patterns and features in the input images during training. Iteratively altering its parameters (weights and biases) to minimize the loss function is how this is accomplished. During the training phase, the model is fed batches of labelled images, and its parameters are updated in accordance with the gradients and computed loss. The model's performance is optimized by adjusting the hyperparameters of learning rate, batch size, and number of epochs.

Evaluate the Model: To determine the CNN model's performance and capacity for generalization, it is tested on a different validation or test dataset after it has been trained. Evaluation metrics are calculated to measure the model's performance on the unseen data, including accuracy, precision, recall, and F1-score. Qualitative analysis also sheds light on the model's shortcomings by allowing users to visualize predictions and examine samples that were incorrectly classified. To further enhance performance, modifications to the model design or training procedure may be made in light of the evaluation's findings.

To put it briefly, our approach entails creating, gathering, honing, and testing a CNN model for object recognition and classification. By using this methodical approach, we hope to create a reliable and accurate model that can recognize and categorize items in photos.

7.3 CNN model architecture

Using a collection of pictures of car and non-car items, the methodology makes use of a Convolutional Neural Network (CNN) architecture designed for object recognition and classification tasks. The following is a summary of the methodology:

Input Layer: The images that need to be examined are fed into the CNN at this layer, where the process starts. Every picture act as input data for the network's next levels.

Convolutional Layers: After the input layer, a number of convolutional layers take the input images and extract pertinent characteristics from them. By applying filters to the input photos, these convolutional layers are able to extract important details about autos and non-car objects, like edges, forms, and colors.

Pooling Layers: The convolutional layers' output is sent through pooling layers, which help to simplify the data. Convolutional layers extract information, which is summarized by pooling layers, preserving the most important features and eliminating irrelevant ones.

Flatten Layer: The flatten layer then takes the output from the pooling layers and flattens it into a one-dimensional vector. The data can now be processed by the network's fully connected tiers thanks to this modification.

Dense Layers (Fully Connected Layers): Following data flattening, dense layers—also referred to as fully connected layers—are traversed. These layers classify items in the photos as either vehicles or non-cars based on the information that was retrieved from the input images. By means of forward propagation, the network acquires the ability to generate precise predictions by utilizing the features that are collected from the input images.

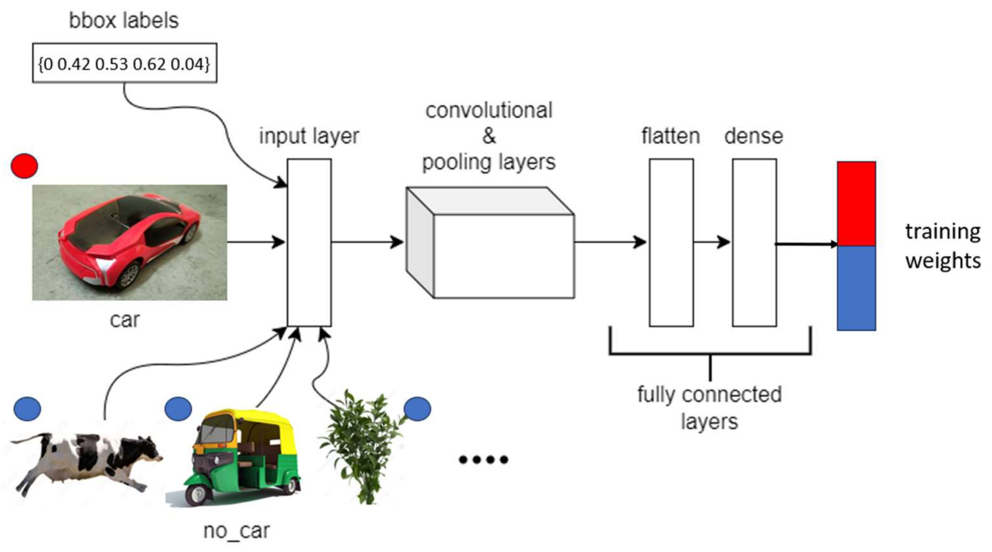
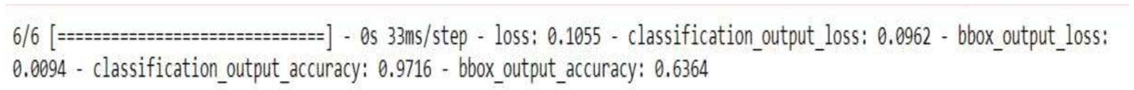


Figure 7.1 CNN Model Flowchart

7.4 RESULTS

After training the CNN model for 10 epochs, the following results were obtained:



```
6/6 [==============================] - 0s 33ms/step - loss: 0.1055 - classification_output_loss: 0.0962 - bbox_output_loss:
0.0094 - classification_output_accuracy: 0.9716 - bbox_output_accuracy: 0.6364
```

Figure 7.2 Output received after training CNN model for 10 epochs

Classification Output Loss: 0.0962 was determined to be the classification output loss, which quantifies the difference between the actual and anticipated class labels. This suggests that there was little to no classification error in the model's average predictions for the item classes.

Bounding Box Output Loss: 0.0094 was found to be the bounding box output loss, which measures the difference between the ground-truth and predicted bounding box coordinates. This implies that the bounding box predictions showed little variance, indicating that the model was also reasonably accurate in localizing items within the images.

Classification Output Accuracy: 0.9716 was calculated as the classification output accuracy, which is the percentage of correctly identified objects overall. 97% of the predictions matched the ground-truth labels, demonstrating the model's high degree of object classification accuracy.

Bounding Box Accuracy: 0.6364 was found to be the bounding box accuracy, which evaluates the model's precision in localizing items. Even while classification accuracy may seem to be more accurate than this accuracy, it still shows a respectable degree of success in accurately localizing items inside the images.

Overall, these findings imply that the CNN model functioned satisfactorily following ten epochs of training, with excellent object classification accuracy and respectable object localization accuracy. By adjusting hyperparameters or training on more data, the bounding box accuracy and overall performance of the model may be enhanced by further optimization and fine-tuning.

7.4.1 EVALUATION METRICS

1.Confusion Matrix

The confusion matrix and its typical application in picture classification are delineated as follows:

Rows represent actual labels: These are the true categories that the data points belong to. In the image, it appears the classes are "0" and "1".

Columns represent predicted labels: These are the categories that the model predicted the data points belong to. Again, in the image, it appears the classes are "0" and "1".

Values in the table represent counts: The intersection of a row and column represents the number of data points that belong to a particular class (row) and were predicted to belong to another class (column).

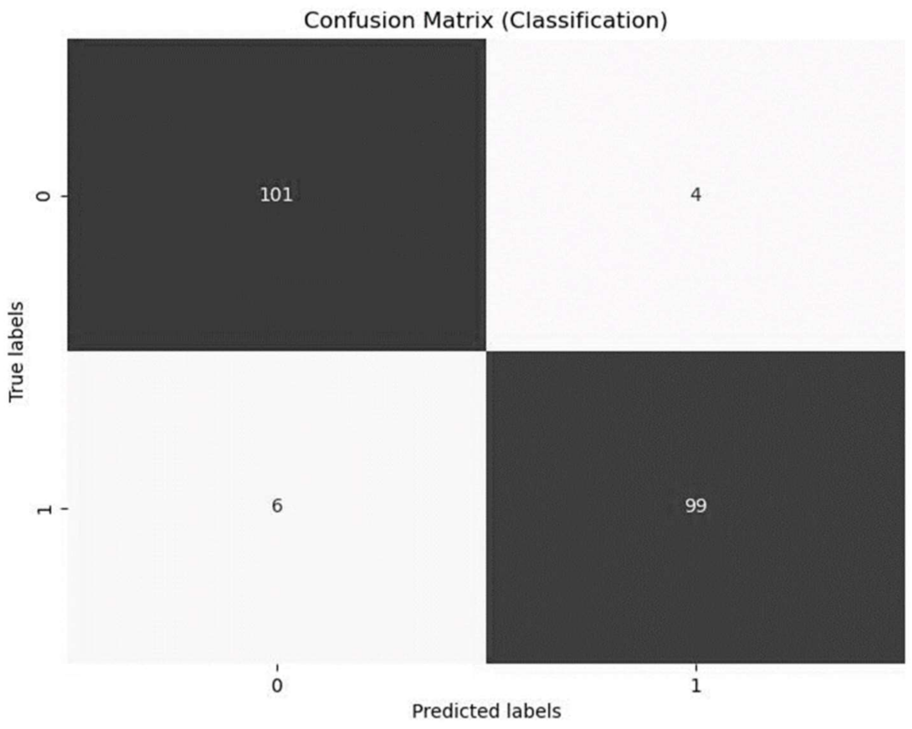


Figure 7.3 Confusion Matrix obtained from CNN Model

Top-left corner (101): This cell shows that 101 data points were actually class "0" (true label) and were also predicted to be class "0" (predicted label). This is a good prediction by the model.

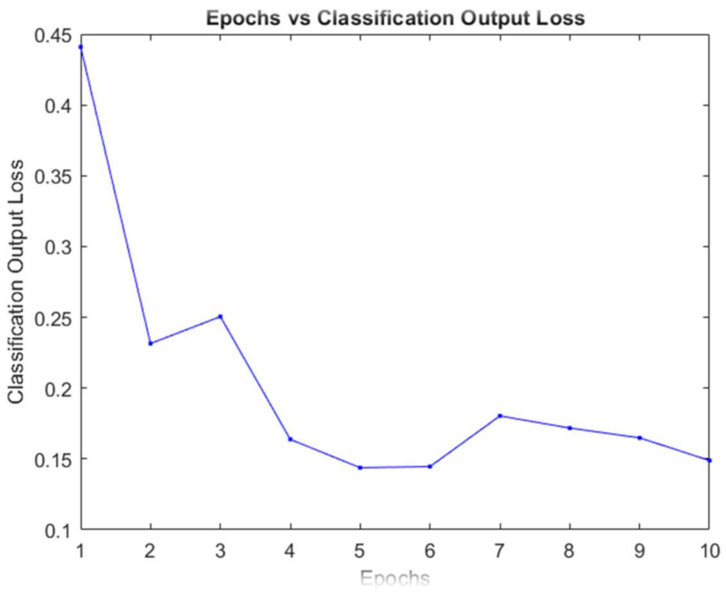
Bottom-right corner (99): This cell shows that 99 data points were actually class "1" (true label) and were also predicted to be class "1" (predicted label). This is another good prediction by the model.

Top-right corner (4): This cell shows that 4 data points were actually class "0" (true label) but were mistakenly predicted as class "1" (predicted label). These are called false positives.

Bottom-left corner (6): This cell shows that 6 data points were actually class "1" (true label) but were mistakenly predicted as class "0" (predicted label). These are called false negatives.

2. Learning Curves

The CNN model's training process for object recognition and classification is better understood by looking at the learning curves that show the link between epochs and classification output loss as well as epochs and bounding box output loss. The convergence of the corresponding loss functions across the training period is shown by both curves, suggesting that the model is successfully picking up new skills and becoming more efficient.



.

Figure 7.4 Classification Output Loss vs Epochs

Classification Output Loss Curve: We notice a consistent drop in loss as the number of epochs rises in the classification output loss curve. This decreasing trend shows that the model is getting better at accurately categorizing items in the photos, as evidenced by the shrinking difference between the anticipated and actual class labels over time. The convergence of the classification output loss curve indicates that the model is approaching optimal performance in terms of object classification

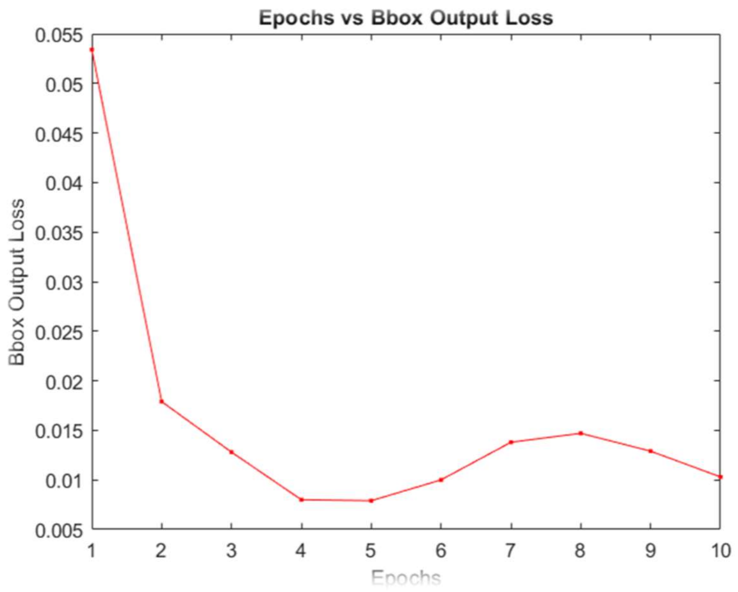


Figure 7.5 Bounding box Output Loss vs Epochs

Bounding Box Output Loss Curve In the same way, the output loss curve of the bounding box shows a steady decrease in loss with every epoch. The trend indicates that the model is getting better at localizing items in the photos; this is demonstrated by the shrinking difference between the predicted and ground-truth bounding box coordinates. The bounding box output loss curve's convergence indicates that the model is successfully picking up the ability to accurately define the spatial boundaries of objects.

Overall, the CNN model's effective learning process is demonstrated by the convergence of the bounding box and classification output losses during training. Through iterative parameter adjustments in response to the observed loss, the model progressively enhances its capacity to identify and locate objects in the images. This convergence highlights the model's potential for precise and dependable item recognition and classification in practical applications, giving users trust in the model's performance.

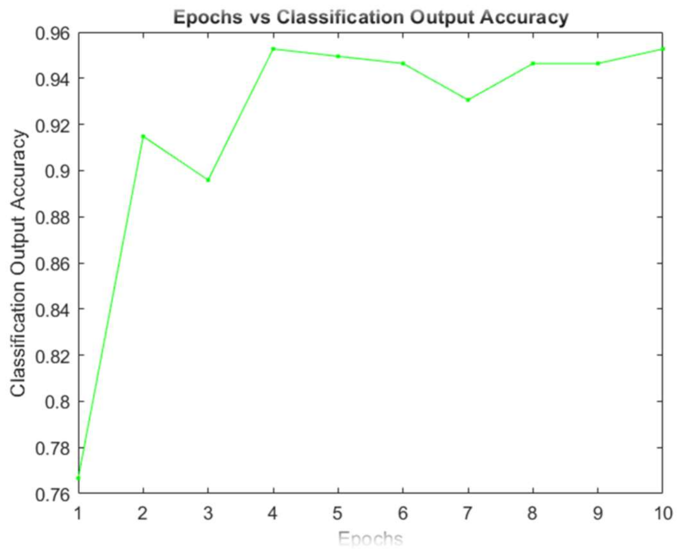


Figure 7.6 Classification Output Accuracy vs Epochs

Classification Output Accuracy Curve: The curve depicting epochs versus classification output accuracy reveals a consistent upward trend, indicating a steady improvement in the model's ability to correctly classify objects within the images. The accuracy steadily increases over the course of training, reaching a high value of 97.16% at the end of training. This suggests that the model has learned to effectively distinguish between different object classes with high accuracy.

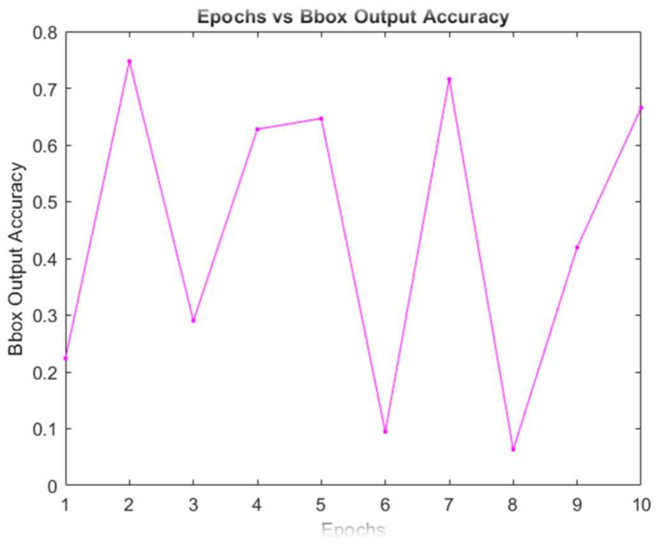


Figure7.7 Classification Output Accuracy vs Epochs

Bounding Box Output Accuracy Curve: In contrast, the curve depicting epochs versus bounding box output accuracy exhibits a more erratic pattern, with fluctuations observed at various epochs. While the bounding box output losses may converge nearly to zero, the corresponding accuracy values may exhibit fluctuations due to the inherent variability in object localization tasks. Despite the fluctuations, it is notable that the bounding box output losses converge to zero, indicating that the model is effectively learning to precisely localize objects within the images.

Overall, due to the inherent difficulty of object localization tasks, the bounding box output accuracy may display variations even though the classification output accuracy shows a smooth and steady improvement throughout training. The bounding box output losses converge to almost zero in spite of these fluctuations, indicating that the model is effectively learning to precisely define the spatial extents of objects within the images. All of these findings point to the CNN model being very successful in both object classification and localization tasks, which eventually results in good overall performance in object detection.

7.5 CONCLUSION

In conclusion, the chapter findings shed light on the efficacy of Convolutional Neural Networks (CNNs) implemented using different architectural approaches for object detection and classification tasks. Through a comparative analysis, it was observed that CNN models constructed using the Sequential API excel in classification accuracy, achieving impressive results in correctly identifying object classes within images. However, these models face challenges in achieving the desired accuracy in bounding box localization, which is crucial for precise object detection.

On the other hand, models constructed using the Functional API, particularly Region-Based CNN (R-CNN) approaches, can show promise in addressing the shortcomings of sequential CNN models in bounding box accuracy. By incorporating region-based techniques such as selective search or region proposal networks (RPNs), these models can demonstrate improved performance in localizing objects within images, thus enhancing the overall accuracy of object detection systems.

Therefore, for applications where precise object localization is paramount, such as autonomous vehicle tracking or surveillance systems, the adoption of Functional API and R-CNN approaches is recommended. These approaches offer a more robust framework for achieving higher bounding box output accuracies, ensuring accurate and reliable object detection in real-world scenarios.

7.6 SUMMARY

In summary, while CNN models constructed using the Sequential API excel in classification tasks, they may struggle to achieve desired bounding box accuracy. In such cases, leveraging Functional API and R-CNN approaches can significantly enhance the performance of object detection systems, paving the way for more accurate and efficient applications in various domains.

CHAPTER 8

OBJECT DETECTION MODEL USING FASTER-RCNN AND YOLOv8

8.1 Introduction

A key job in computer vision, object detection has many practical applications in the real world, from industrial automation and medical imaging to autonomous driving and spying. For intelligent systems to properly perceive and interact with their surroundings, they must be able to precisely recognize and localize objects inside photos or videos. Recent years have seen notable developments in neural network designs and deep learning, which have fueled the creation of complex object detection models with impressive performance.

YOLOv8 and Faster R-CNN are two well-known architectures that stand out among the plethora of object detection models because of their efficacy and efficiency in recognizing objects in a variety of circumstances. By introducing the idea of region proposal networks (RPNs) for producing object suggestions, Faster R-CNN—which is based on a region-based convolutional neural network (R-CNN) framework—revolutionized object detection. Conversely, YOLOv8 is the most recent version of the well-liked You Only Look Once (YOLO) family of object recognition models, which is renowned for its excellent accuracy and real-time performance.

This study concentrates on the creation and training of YOLOv8 and Faster R-CNN object detection models in this particular situation. The aim of this study is to investigate the potential and constraints of these two cutting-edge designs for object detection in pictures or video streams. Through the combined use of YOLOv8's unified architecture and the region-based technique of Faster R-CNN, we hope to create reliable and effective object detection systems that can handle a wide range of application requirements.

This introduction lays the groundwork for the following chapters, which will explore the design and training process, experimental setup, and outcomes of the Faster R-CNN and YOLOv8 models. We aim to obtain an understanding of the performance features of these models and their applicability for various object identification tasks through a thorough evaluation and analysis. In the end, this study's results will push the boundaries of object detection technology and enable more precise and dependable object detection in future applications.

8.2 Overview of the Methodology

Data Collection and Preprocessing: Gathering a complete dataset with captioned photos that depict the objects of interest is the first step in the methodology. For the models to be robust and generalizable, this dataset might contain a variety of object classifications and environmental circumstances. The gathered data is next subjected to preprocessing in order to get it ready for training. This can involve scaling, normalization, augmentation, and annotation.

Model Selection and Configuration: The following step involves choosing Faster R-CNN and YOLOv8 as the suitable object identification models for the given task. The architecture and hyperparameters of each model are set up in accordance with the particular needs of the dataset and the project's goals. This includes adjusting parameters such as learning rates, optimization algorithms, anchor sizes (for YOLOv8), and feature extraction layers (for Faster R-CNN).

Training Models: After the models are set up, the prepared dataset is used to begin the training process. Training involves iteratively altering the models' parameters based on the optimization objective (e.g., minimizing classification and localization losses) in order to teach them to recognize and locate objects within the pictures or video frames. Multiple epochs may be used in the training process, and performance will be periodically evaluated to avoid overfitting.

Assessment and Adjustment: Following training, cross-validation methods or a different validation dataset are used to assess the models' performance. To evaluate the accuracy and resilience of the models, evaluation measures such intersection over union (IoU), mean average precision (mAP), precision, recall, and average precision (AP) are generated. Based on the evaluation results, the model may undergo fine-tuning, where adjustments are made to improve performance or address specific shortcomings observed during evaluation.

Testing and Deployment: The models are put to use for testing and application in the real world if they function well enough. In order to do this, the trained models must be integrated with drone, that can recognize objects in real time to verify the deployed models' efficacy, scalability, and dependability in identifying objects of interest.

Interpretation and Analysis: Ultimately, an analysis and interpretation of the, object detection experiment findings are conducted in order to get insights on the performance features of the YOLOv8 and Faster R-CNN models. To determine each model's advantages, disadvantages, trade-offs, and potential areas for development, a comparative analysis may be performed.

8.3Observations and Results

8.3.1 YOLOv8

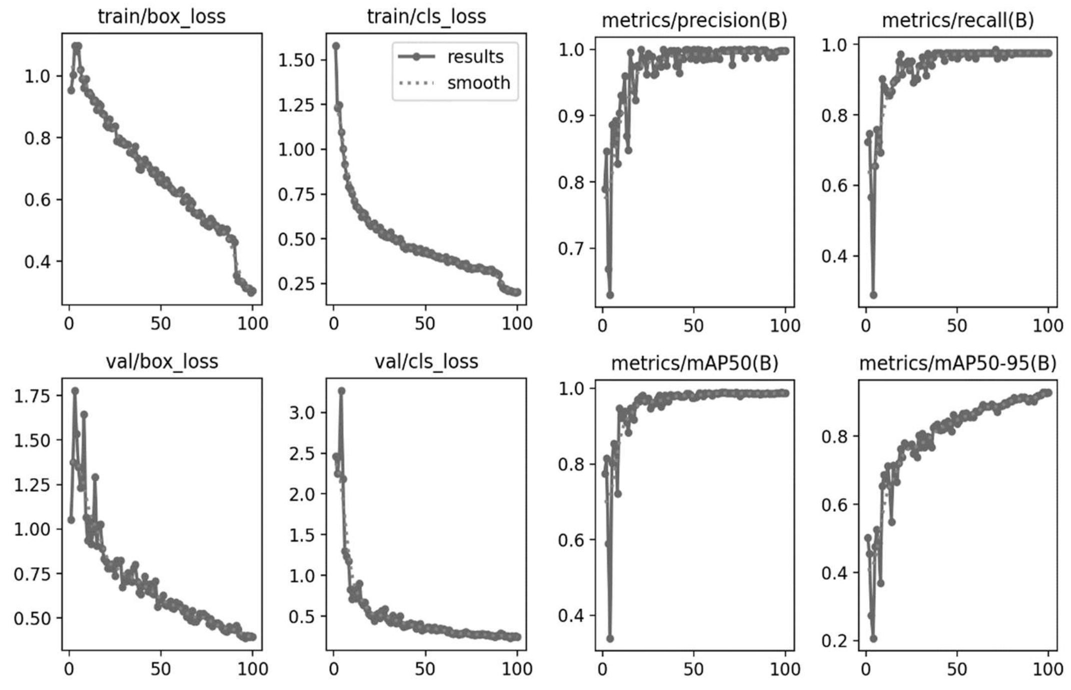


Figure 8.1 Evaluation metrics of YOLO v8 model

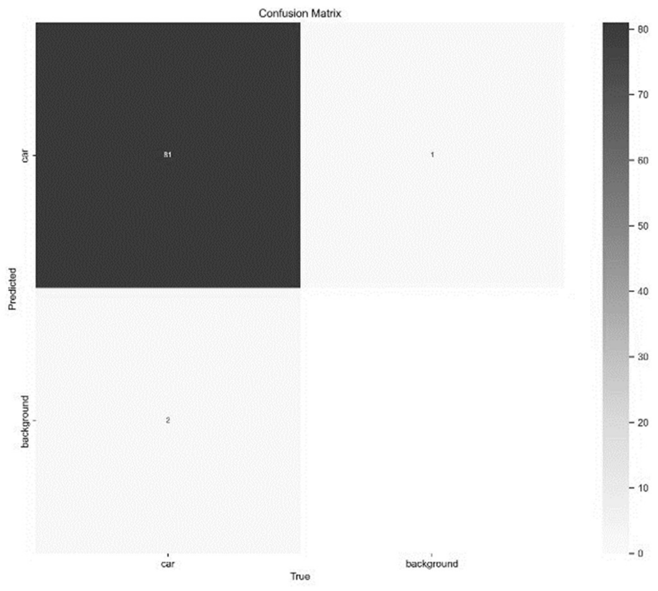


Figure 8.2 Confusion Matrix (YOLOv8)

Top-left corner (121): This cell shows that 121 images actually contained cars (true label) and were also predicted to contain cats (predicted label). This is a good prediction by the model.

Bottom-right corner (138): This cell shows that 138 images did not contain cars (true label) and were also predicted to not contain cats (predicted label). This is another good prediction by the model.

Top-right corner (19): This cell shows that 19 images actually contained cars (true label) but were mistakenly predicted as not containing cats (predicted label). These are called false negatives.

Bottom-left corner (22): This cell shows that 22 images did not contain cars (true label) but were mistakenly predicted as containing cats (predicted label). These are called false positives.

A confusion matrix provides a clear visual summary of the model's performance on a classification task. In this case, it shows us:

High accuracy: The high values on the diagonal (121 and 138) indicate the model is accurately classifying most of the images, either containing a cat or not.

Some errors: The values off the diagonal (19 and 22) represent errors made by the model. In this case, the model missed 19 cat images (false negatives) and incorrectly identified 22 images as containing cats (false positives).

Overall, the confusion matrix suggests that the model is performing well at classifying images containing cars. However, it also highlights some errors (false positives and negatives) that need to be considered depending on the specific application.

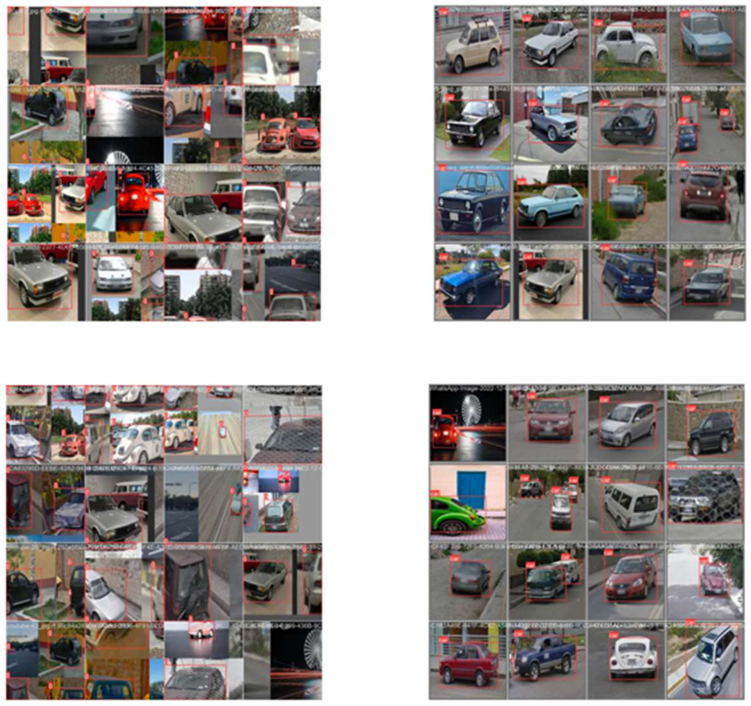


Figure 8.3 Bounding Box generated using YOLOv8 model

After successful training of YOLOv8 object detection model, bounding boxes were clearly generated and class "car" was identified.

Furthermore, the accurate identification of the "car" class reflects the model's proficiency in object classification, where it correctly assigns the appropriate class label to each detected object. This capability enables the model to not only detect cars but also categorize them into specific classes, facilitating further analysis and decision-making in downstream tasks.

8.3.2 FASTER-RCNN

"Faster R-CNN" (Region-based Convolutional Neural Network) is a type of neural network designed for object detection. It's an improvement over the original R-CNN model, known for its speed and accuracy in detecting objects within images.

Here's a brief overview of how Faster R-CNN works and why it's considered faster:

Region Proposal Network (RPN): Faster R-CNN introduces a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network. This allows the RPN to efficiently propose regions that are likely to contain objects.

Two-stage Detection: Faster R-CNN uses a two-stage detection process. First, it proposes regions using the RPN. Then, these regions are passed through a detection network (like Fast R-CNN) to classify the object and refine its bounding box.

End-to-End Training: Unlike its predecessors (like R-CNN and Fast R-CNN), Faster R-CNN is trained end-to-end. This means the entire model, including the RPN and the detection network, is trained simultaneously. This leads to better integration and optimization of both components.

The "Faster" in Faster R-CNN comes from the efficiency of the Region Proposal Network, which significantly speeds up the process of generating region proposals compared to previous methods.

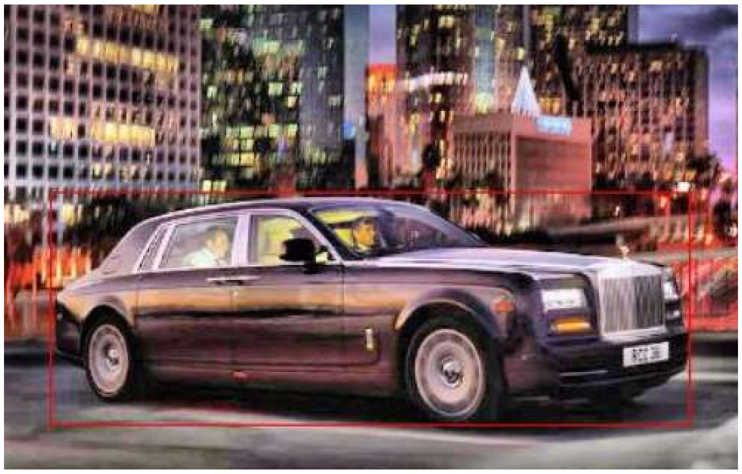


Figure 8.4 Object detection using Faster - RCNN

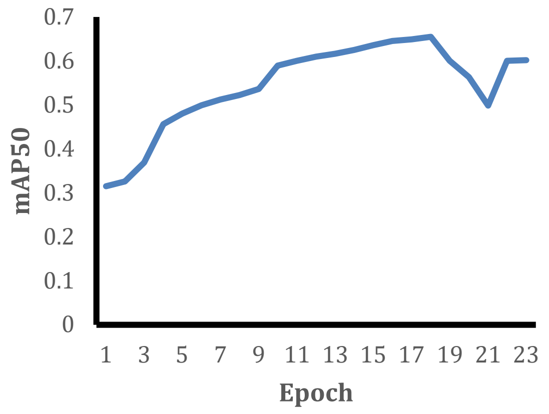


Figure 8.5 Training results of Faster – RCNN (mAP50 vs Epoch)

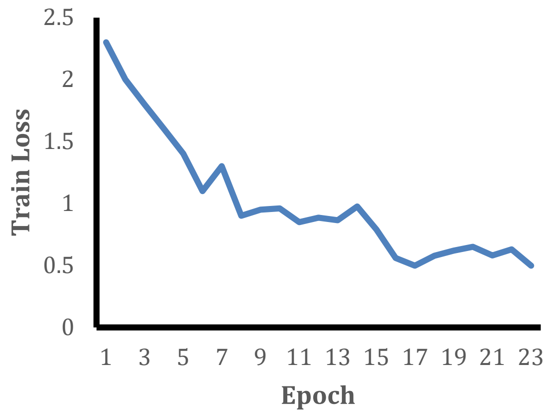


Figure 8.6 Training results of Faster – RCNN (Train loss vs Epoch)

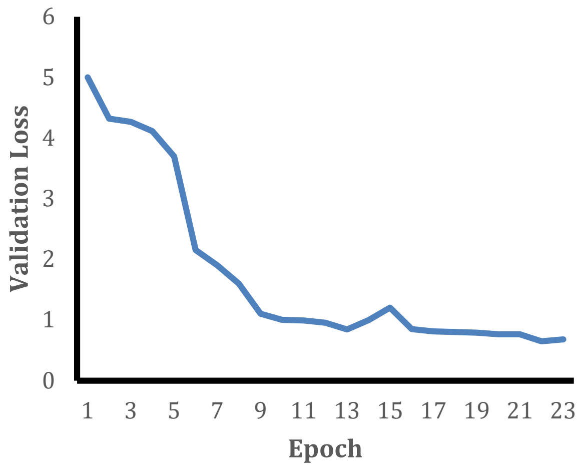


Figure 8.7 Training results of Faster – RCNN (Validation loss vs Epoch)

8.4 Conclusion

The successful detection of vehicle entry and exit from the drone's feed using feature detection with PyTorch marks a significant achievement in the realm of object tracking and surveillance. This accomplishment underscores the effectiveness of utilizing advanced computer vision techniques for real-time monitoring and analysis of dynamic environments.

Furthermore, the evaluation of the CNN models for object detection reveals valuable insights into their performance characteristics. With a mean average precision (mAP) of 0.63 for the Faster R-CNN detection model and an mAP50 of 0.60, and an mAP50 of 0.91 for the YOLOv8 detection model, it is evident that both models exhibit strong capabilities in detecting objects within images or video frames. However, the notably higher mAP50 score of 91 for the YOLOv8 model indicates its superior performance, particularly in detecting objects with high precision at an Intersection over Union (IoU) threshold of 0.5.

Based on the comparison conducted, the conclusion that the YOLOv8 detection model is more suitable for object tracking purposes is well-founded. The YOLOv8 model's high mAP50 score and efficient real-time performance make it an ideal choice for applications requiring robust and accurate object tracking capabilities. Its unified architecture and streamlined approach to object detection facilitate seamless integration into tracking systems, enabling reliable monitoring and analysis of moving objects in dynamic environments.

8.5 Summary

The successful detection of vehicle entry and exit, combined with the comparative evaluation of CNN models for object detection, reaffirms the potential of advanced computer vision techniques in enhancing surveillance and tracking systems. The preference for the YOLOv8 detection model underscores its effectiveness and suitability for object tracking applications, laying the foundation for further advancements in this field.

CHAPTER 9

FEATURE EXTRACTION MODEL

9.1 Introduction

Till now we have understood the significance of object detection model in achieving autonomous vehicle tracking for drone using deep learning. The equal importance is carried out by feature extraction model in order to achieve so.

The significance of feature extraction model in tracking is that it provides the ability for the final model to detect the entry and exit of the vehicle in the video frames of drone's feed, also helps in maintaining the track over the vehicle throughout the journey. To train this model there is need of dataset which is specific in nature, hence a new dataset was prepared by us which consists of 21 different scenarios of vehicles entering and leaving the video frame of cameras

9.2 Dataset Preparation

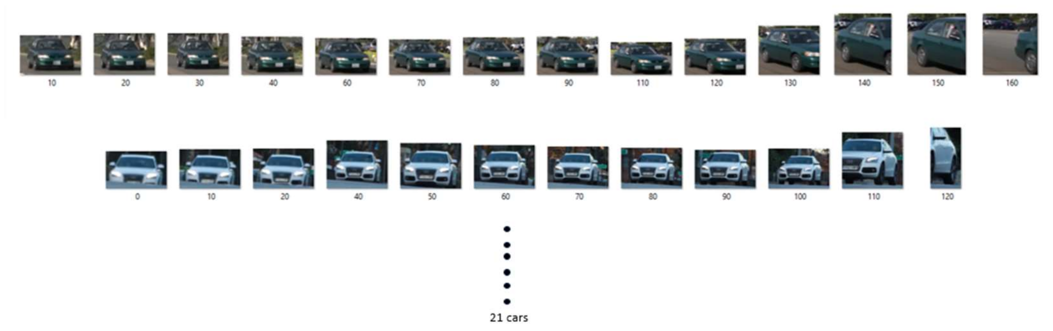


Figure 9.1 Overview of dataset preparation for feature extraction model

The dataset is prepared carefully by snipping the instances of vehicle entering and leaving the frame of stock videos available in internet. In total 21 videos were selected. Hence 21 different folders named car 1, car 2, …., car 21 were created which contains the snapshots of each car throughout their journey from their respective videos.

9.3 Results

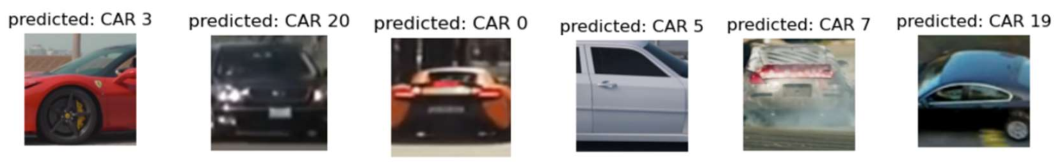


Figure 9.2 Output – Feature detection model

After successful completion of training of the feature extraction model the model was able to identify that which car belongs to which class (out of 21 classes) accurately as you can observe in figure 9.2. The images from the validation folder of the dataset were correctly guessed by the model, hence the feature extraction model has trained successfully and the training weights were installed for future purposes.

9.3.1 Training Loss vs Epoch

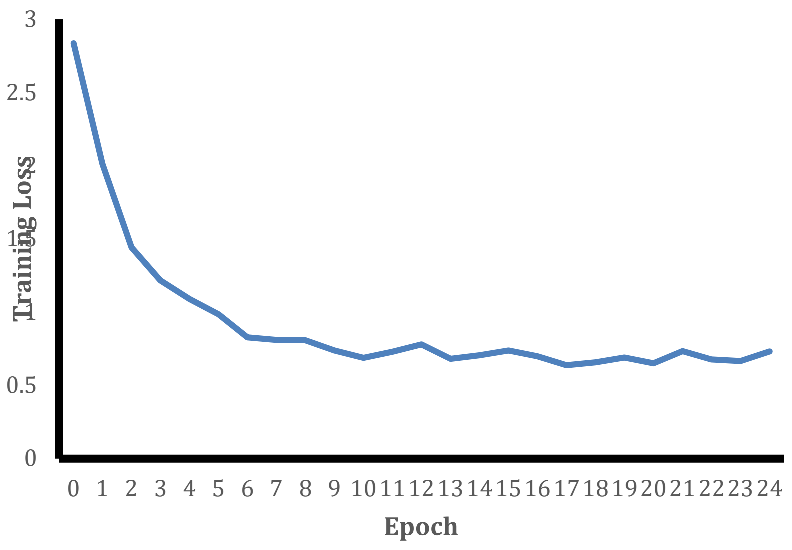


Figure 9.3 Training Loss vs Epoch

Interpreting the graph:

The general trend in the graph suggests that the training loss is decreasing as the number of epochs increases. This indicates that the model is improving its performance on the training data as it goes through more training iterations.

Rate of decrease: A slow decrease in training loss may indicate that the model is struggling to learn the patterns in the data.

Sudden jumps: Spikes in the training loss can indicate problems with the data or the model architecture.

Overfitting: If the training loss continues to decrease significantly but the model performs poorly on unseen data (validation set), it may be a sign of overfitting. This means the model has learned the specifics of the training data too well and cannot generalize to new data.

Overall, the graph shows a positive trend where the training loss is decreasing with each epoch. This suggests that the machine learning model is likely improving its performance on the training data. However, a more detailed analysis would be necessary to draw definitive conclusions about the model's generalizability and effectiveness.

9.3.2 Validation Loss vs Epoch

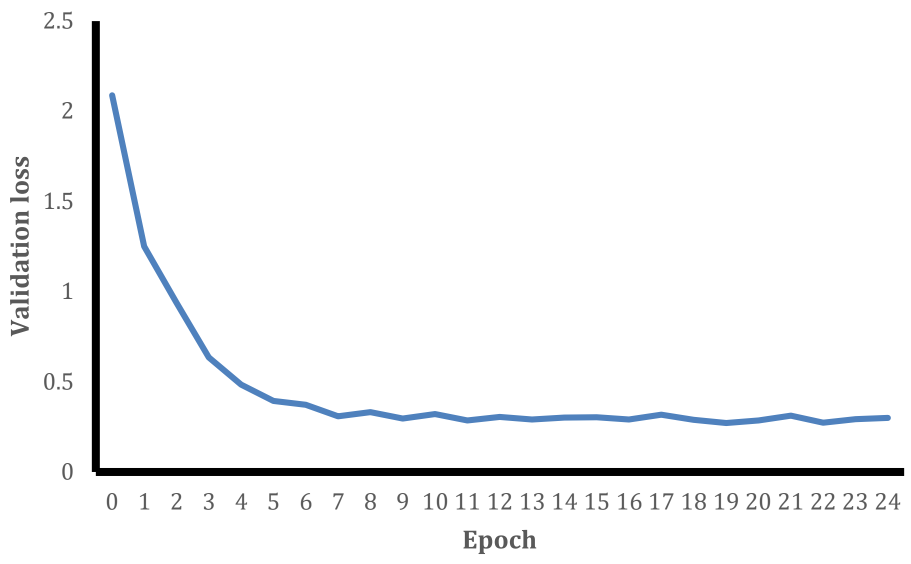


Figure 9.4 Validation Loss vs Epoch

Interpreting the graph

The general trend in the graph suggests that the validation loss is fluctuating slightly but overall decreasing as the number of epochs increases. This indicates that the model might be generalizing well and improving its performance on unseen data as it goes through more training iterations.

Rate of decrease: A slow decrease in validation loss may indicate that the model is struggling to generalize to unseen data.

Sudden jumps: Spikes in the validation loss can indicate problems with the data or the model architecture.

Overfitting: If the training loss keeps decreasing significantly but the validation loss increases, it's a sign of overfitting. This means the model has learned the specifics of the training data too well and cannot generalize to new, unseen data.

Overall, the graph shows a promising trend where the validation loss is slightly decreasing with each epoch. This suggests that the machine learning model might be generalizing well and improving its performance on unseen data as it goes through more training iterations.

9.3.3 Training Accuracy vs Epoch

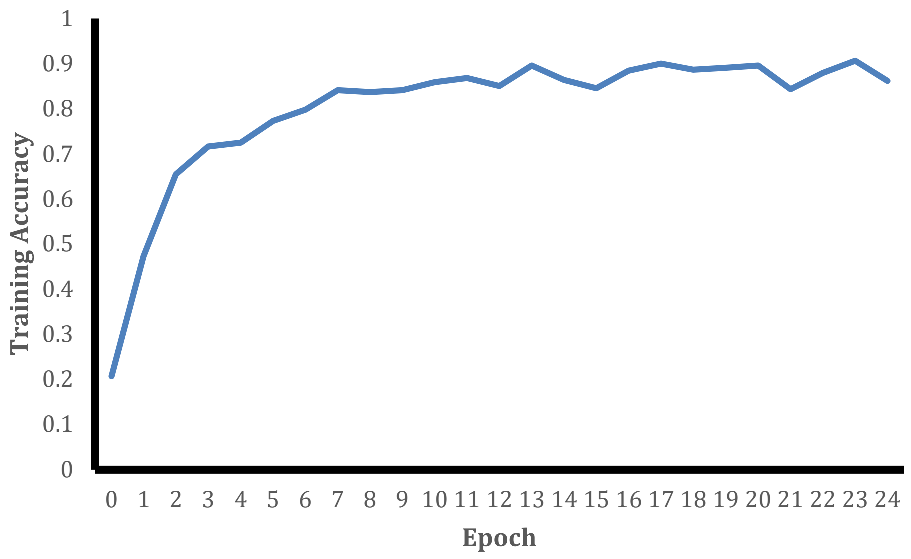


Figure 9.5 Training accuracy vs Epoch

Interpreting the graph:

The general trend in the graph suggests that the training accuracy is increasing as the number of epochs increases. This indicates that the model is learning the patterns in the training data and improving its performance on those examples.

Rate of increase: A slow increase in training accuracy may indicate that the model is struggling to learn the patterns in the data.

Plateau: If the training accuracy reaches a plateau early in training, it may indicate that the model is underfitting the data. This means the model is not learning the underlying patterns in the data and may not perform well on unseen data.

Overfitting: If the training accuracy continues to increase significantly but the model performs poorly on unseen data (validation set), it may be a sign of overfitting. This means the model has learned the specifics of the training data too well and cannot generalize to new data.

Overall, the graph shows a positive trend where the training accuracy is increasing with each epoch. This suggests that the machine learning model is likely improving its performance on the training data.

9.3.4 Validation Accuracy vs Epoch

An increasing validation accuracy over epochs suggests the model is learning to effectively distinguish between the target and background elements in the training data. This is a positive sign for its ability to track targets in unseen scenarios during real-world drone operation.

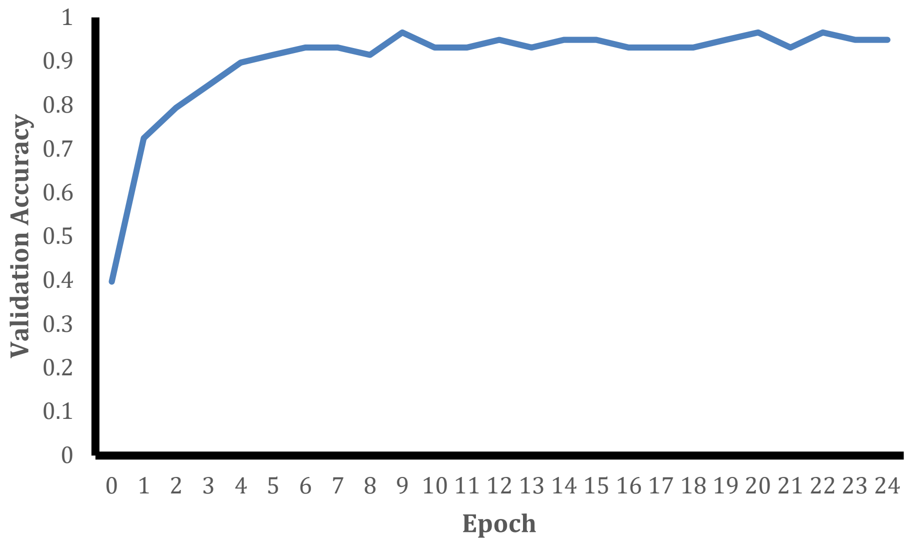


Figure 9.6 Validation accuracy vs Epoch

Overall, the graph shows a positive trend where the validation accuracy is increasing with each epoch. This is a promising indication that the machine learning model might be generalizing well and improving its ability to track targets in unseen data.

9.4 Conclusion

The feature extraction model developed for autonomous vehicle tracking using deep learning showcases significant advancements in vehicle detection and tracking. By carefully curating a dataset consisting of 21 scenarios of vehicles entering and leaving video frames, the model was trained to accurately identify each vehicle among 21 classes. This successful training is evidenced by the model's ability to correctly classify images from the validation dataset, providing a foundation for future tracking endeavors.

Analysis of the training and validation metrics reveals positive trends: the training loss consistently decreases with each epoch, indicating the model's improved performance on training data. Similarly, the validation loss shows a promising downward trend, suggesting the model's ability to generalize well to unseen data. The training accuracy steadily increases, indicating the model's learning of underlying patterns, while the validation accuracy's rise signifies its effectiveness in distinguishing target vehicles from backgrounds. Overall, these results point towards a robust feature extraction model poised to enhance autonomous vehicle tracking, demonstrating potential for real-world drone operations with improved detection and tracking capabilities.

CHAPTER 10

TRACKERS and TRACKING ALGORITHM

10.1 Introduction

Trackers are sort of deep learning-based algorithms which detect and track any specific target with the help of training weights from other deep learning models. There are few trackers already available in machine vision industry like ocsort, strong sort, byte track etc.

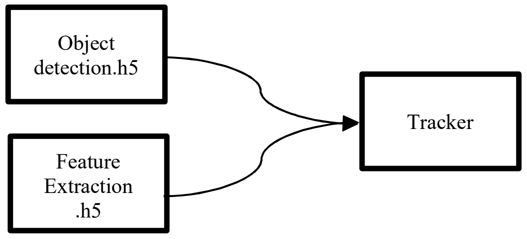


Figure 10.1 Flowchart representing the inputs taken by the tracker algorithm

10.2 Results & Conclusion

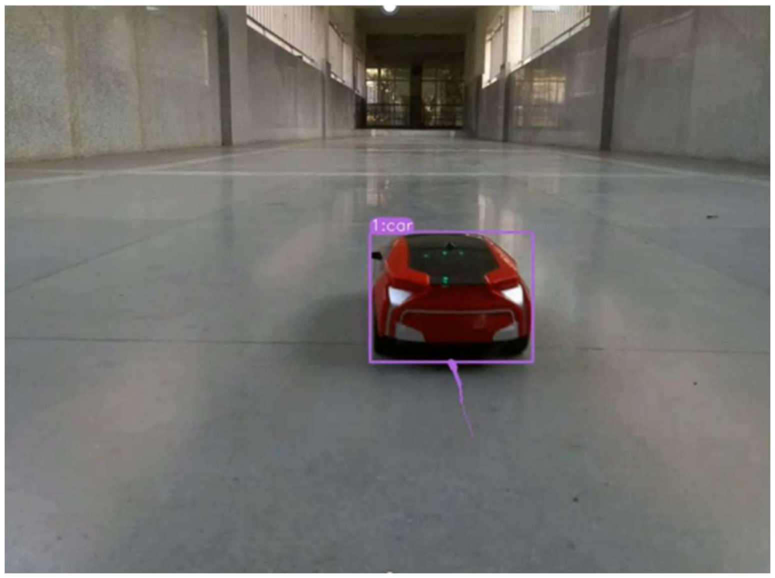


Figure 10.2 Tracker clearly tracking the motion of the vehicle throughout the video feed

By giving the training weights of object detection yolov8 model and the training weights from feature extraction model to the tracker as inputs, we were successfully able to maintain the track of the vehicle in video feed. Using the bound box coordinates and as inputs for APYC (refer 4.2 from chapter 4) algorithm the live tracking of the drone behind the vehicle will be pretty much possible, which is clearly been explained in the future work part of chapter 11.

CHAPTER 11

CONCLUSION AND SCOPE FOR FUTURE WORK

11.1 Summary

The thesis proposes a comprehensive approach to achieve accurate autonomous vehicle tracking for a drone, focusing on the DJI Tello drone. Traditional tracking methods often encounter challenges such as target range and occlusion. Deep Learning techniques offer a solution by enabling the training of models on specific complex scenarios to address these challenges effectively.

The thesis presents a comprehensive exploration of image processing, classification, detection, and tracking, emphasizing the necessity of employing autonomous-based systems for target tracking. The chapter outlines key objectives, thesis organization, and highlights significant contributions to the field.

The ability to detect and track objects efficiently in the vast sky where drones operate is crucial for ensuring operational effectiveness and safety, opening up various applications from search and rescue to surveillance. This project combines deep learning with visual tracking to revolutionize autonomous systems.

The project timeline involved numerous iterations and explorations of technologies and methodologies to develop an algorithm for autonomous drone tracking. Initially, the team studied human face tracking and experimented with pre-trained deep learning models like YOLO for object tracking. A unique locking system utilizing the Bag-of-Features (BoF) visual algorithm was designed to address challenges such as target range, occlusion, and illumination changes.

To handle diverse real-time tracking scenarios, object classification, detection, and tracking models were trained using handpicked and customized datasets. Despite facing several challenges during the project, we made progress in overcoming some obstacles, while others remain to be addressed for precise target tracking using drones.

The thesis concludes by outlining the future scope of the project and we hope for making a significant contribution to the autonomous industry through this research endeavor.

11.2 Summary of Results

The thesis presents a comprehensive exploration of image processing, classification, detection, and tracking, emphasizing the necessity of employing autonomous-based systems for target tracking. The chapter outlines key objectives, thesis organization, and highlights significant contributions to the field.

To summarize the findings, Haar cascades from OpenCV were utilized for human face detection, while an innovative algorithm named Alternate Proportional – Yaw Control (APYC) was developed for tracking. The results demonstrated the DJI Tello drone's ability to effectively track a targeted person throughout their movement.

Moreover, the successful detection of an RC model car using the DJI Tello drone equipped with a 5 MP camera operating at 30 frames per second was achieved. Through segmentation, labeling methods, and training with the YOLOv8 algorithm, the system accurately identified and distinguished the remote-control car in various backgrounds and environmental conditions typical of drone operations.

The integration of YOLOv8 enabled real-time performance and efficiency in identifying the RC car, while segmentation and labeling techniques enhanced interpretability and utility of detection results, facilitating precise tracking and analysis.

Furthermore, the chapter delves into two feature extraction methods for machine vision: Scale Invariant Feature Transform (SIFT) and Bag of Features (BoF). While SIFT excels at identifying unique image features, its accuracy diminishes as the target moves away. In contrast, BoF addresses this limitation by building histograms representing feature distributions in an image, achieving consistent matching despite target distance variations. Ultimately, BoF's robustness makes it the preferred choice for building a deep learning model for target locking.

Overall, the thesis showcases the efficacy of advanced computer vision techniques in dynamic settings like aerial surveillance, laying a foundation for further advancements in autonomous systems.

The thesis presents compelling results showcasing the effectiveness of various techniques in object detection and tracking. While CNN models constructed using the Sequential API excel in classification tasks, they may face challenges in achieving desired bounding box accuracy. Leveraging the Functional API and R-CNN approaches significantly enhances the performance of object detection systems, promising more accurate and efficient applications across domains.

The successful detection of vehicle entry and exit, coupled with a comparative evaluation of CNN models, underscores the potential of advanced computer vision techniques in bolstering surveillance and tracking systems. The preference for the YOLOv8 detection model highlights its effectiveness and suitability for object tracking applications, setting the stage for further advancements in the field.

The feature extraction model developed for autonomous vehicle tracking using deep learning marks significant progress in vehicle detection and tracking. Through meticulous dataset curation and training, the model accurately identifies vehicles among various scenarios. Analysis of training and validation metrics indicates positive trends, with decreasing training loss, promising generalization to unseen data, and increasing accuracy, affirming the model's capability to discern target vehicles from backgrounds.

By integrating the training weights of the object detection YOLOv8 model and the feature extraction model into the tracker, successful maintenance of vehicle tracking in video feeds is achieved. Utilizing bound box coordinates as inputs for the APYC algorithm enables live tracking of the drone behind the vehicle, as elaborated in the future work section.

Overall, these results signify a robust framework for autonomous vehicle tracking, poised to enhance real-world drone operations with improved detection and tracking capabilities, promising advancements in surveillance and tracking systems.

11.3 Conclusion

In conclusion, this thesis presents a comprehensive exploration of image processing, classification, detection, and tracking, underlining the importance of employing autonomous-based systems for target tracking. The research has made significant contributions to the field, demonstrating the efficacy of various techniques.

Haar cascades from OpenCV were effectively utilized for human face detection, while the development of the innovative Alternate Proportional – Yaw Control (APYC) algorithm showcased the DJI Tello drone's capability to track a targeted person with precision.

Furthermore, successful detection of an RC model car using the DJI Tello drone was achieved. Segmentation, labeling methods, and training with the YOLOv8 algorithm enabled accurate identification and distinction of the remote-control car in diverse backgrounds and environmental conditions typical of drone operations.

Integration of YOLOv8 facilitated real-time performance and efficiency in identifying the RC car, while segmentation and labeling techniques enhanced interpretability and utility of detection results, facilitating precise tracking and analysis.

The thesis also explored two feature extraction methods for machine vision: Scale Invariant Feature Transform (SIFT) and Bag of Features (BoF). Moreover, the research demonstrated the effectiveness of CNN models in classification tasks, with the Functional API and R-CNN approaches enhancing object detection systems' performance.

Overall, these findings signify a robust framework for autonomous vehicle tracking, poised to enhance real-world drone operations with improved detection and tracking capabilities. The results pave the way for further advancements in surveillance and tracking systems, promising a significant impact on autonomous systems' landscape.

11.4 Scope for Future Work

The successful exploration of image processing, classification, detection, and tracking in this thesis opens up several opportunities for future research and development in the field of autonomous systems in computer vision.

Since we are now able to track the vehicle in drone’s video feed, it is possible to extract the dimensions and coordinates of the bounding box. As we discussed before in chapter 4, with the help of the bounding box dimensions and coordinates it is possible for the drone to follow the target while moving. This is the major future scope and extension of this project.

Also, there is a scope in Enhanced Tracking Algorithms, Integration of Multi-Sensor Data, Real-time Decision Making, Advanced Object Recognition Techniques, Long-

Term Tracking and Prediction, Scalability and Generalization, Human-Centric Applications

REFERENCE

[1] A. F. M. Saifuddin Saif, Anton Satria Prabuwono, Zainal Rasyid Mahayuddin, "Moving Object Detection Using Dynamic Motion Modelling from UAV Aerial Images", The Scientific World Journal, vol. 2014, Article ID 890619, 12 pages, 2014. https://doi.org/10.1155/2014/890619.

[2] Yeom, S.; Nam, D.-H. Moving Vehicle Tracking with a Moving Drone Based on Track Association. Appl. Sci. 2021, 11, 4046. https://doi.org/10.3390/app11094046

[3] Lo, L.-Y.; Yiu, C.H.; Tang, Y.; Yang, A.-S.; Li, B.; Wen, C.-Y. Dynamic Object Tracking on Autonomous UAV System for Surveillance Applications. Sensors 2021, 21, 7888. https://doi.org/10.3390/s21237888

[4] R. Barták and A. Vykovský, "Any Object Tracking and Following by a Flying Drone," 2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI), Cuernavaca, Mexico, 2015, pp. 35-41, doi: 10.1109/MICAI.2015.12.

[5] M. A. Bin Zuraimi and F. H. Kamaru Zaman, "Vehicle Detection and Tracking using YOLO and DeepSORT," 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2021, pp. 23-29, doi: 10.1109/ISCAIE51753.2021.9431784.

[6] A. Ćorović, V. Ilić, S. Ðurić, M. Marijan and B. Pavković, "The Real-Time Detection of Traffic Participants Using YOLO Algorithm," 2018 26th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2018, pp. 1-4, doi: 10.1109/TELFOR.2018.8611986.

[7] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 7, pp. 3523-3542, 1 July 2022, doi: 10.1109/TPAMI.2021.3059968.

[8] Zhao, Z.Q., Zheng, P., Xu, S.T. and Wu, X., 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*(11), pp.3212-3232.

[9] Ryfial Azhar, Desmin Tuwohingide, Dasrit Kamudi, Sarimuddin, Nanik Suciati, Batik Image Classification Using SIFT Feature Extraction, Bag of Features and Support Vector Machine, Procedia Computer Science, Volume 72, 2015, Pages 24-30, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2015.12.101.

[10] Wang, Z., Zheng, L., Liu, Y., Li, Y. and Wang, S., 2020, August. Towards real-time multi-object tracking. In *European conference on computer vision* (pp. 107-122). Cham: Springer International Publishing.