AUTOMATIC VEHICLE COUNTING AND DETECTION SYSTEM USING CNN AND SVM

Project work submitted to Avinashilingam Institute for HomeScience and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Submitted By

M. Prabha (20PIT006)

Under the guidance of

Dr. T.Jayamalar M.C.A., M.Phil., Ph.D,NET

Assistant Professor, Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE ANDHIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

Coimbatore-641043

MAY 2022

AUTOMATIC VEHICLE COUNTING AND DETECTION SYSTEM USING CNN AND SVM

Project work submitted to Avinashilingam Institute for HomeScience and Higher Education for Women

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Submitted By

M. Prabha (20PIT006)

Under the guidance of

Dr. T.Jayamalar M.C.A., M.Phil., Ph.D,NET

Assistant Professor, Department of Information Technology

AVINASHILINGAM INSTITUTE FOR HOME SCIENCE ANDHIGHER EDUCATION FOR WOMEN

SCHOOL OF PHYSICAL SCIENCES AND COMPUTATIONAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

Coimbatore-641043

MAY 2022

DECLARATION

DECLARATION

I hereby declare that the project entitled **" AUTOMATIC VEHICLE COUNTING AND DETECTION SYSTEM USING CNN AND SVM "** is a record of the original work done by M.Prabha (20pit006) under the guidance of Dr.T.Jayamalar M.C.A..,M.Phil.,Ph.D., Assistant Professor, Department of Information Technology, School of Physical sciences and Computational sciences, Avinashilingam Institute for Home Science and Higher Education for Women, in the partial fulfillment for the degree of Master of Science in Information Technology and this project has not formed the basis for any Degree /Diploma / Associates.

Place:

Date:

Signature of the Candidate

Countersigned by

Dr. T. Jayamalar M.C.A., M.Phil., Ph.D.,

Assistant Professor and Head Department of Information Technology,

School of Physical Sciences and Computational Sciences

CERTIFICATE

CERTIFICATE

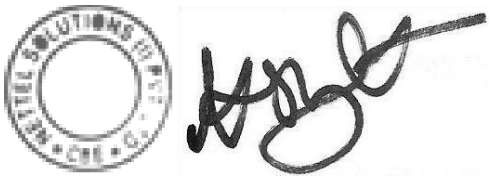
Net Tel Solutions India Pvt Ltd

Excellence in Service

10.05.2022
Coimbatore

TO WHOMSOEVER IT MAY CONCERN This is to confirm that Ms. Prabha.M (Reg No:20pit006) final year student of M.Sc (Information technology)from "Avinashilingam Institute for Home Science and Higher Education for Women" Coimbatore, has successfully completed his Project work on "**AUTOMATIC VEHICLE COUNTING AND DETECTION SYSTEM USING CNN AND SVM**" in our esteemed organization from Jan 2022 to May 2022.

For Net Tel Solution India Pvt Ltd



Authorized Signatory

CERTIFICATE

This is to certify that this project work entitled "**AUTOMATIC VEHICLE COUNTING AND DETECTION SYSTEM USING CNN AND SVM**" done by M.Prabha(20PIT006) has been submitted to Avinashilingam Institute for Home science and Higher Education for Women, Coimbatore-43 in partial fulfillment of the requirement for the award of the degree of MASTER OF SCIENCE ININFORMATION TECHNOLOGY. This Project has not found the basis for the award of any Degree/ Associate/ fellowship or similar title to any Candidate of any University. Certified as a bonafied record of the work submitted for the Viva voce held on _____.

Signature of the HOD

Signature of the Guide

Signature of External Examiner

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

I would like to express my sincere thanks to God Almighty, for his constant love and grace that he has showered upon me, which kept me in good health, and sound mind without which my project would not have reached a successful end.

I would like to express my deep sense of reverential gratitude and sincere thanks to **Shri, Dr.S.P.Thyagarajan, Ph.D, M.D, D.Sc, FAMS, FNASc, FIMSA, FABMS, FFTM (Glasgow, UK), Chancellor,**Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities during my course of study.

I owe my great deal of gratitude to **Dr.V.Bharathi Harishankar Ph.D., FRSA, Vice Chancellor,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for extending all resources that facilitated the smooth conduct of the project study.

I express my gratitude to **Dr. S. Kowsalya, M.Sc., M.Phil,Ph.D Registrar,** Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, for providing all facilities and support necessary for the study.

I wish to extend my sincere thanks **Dr.(Mrs.)G.Padmavathi M.Sc., M.Phil., Ph.D.**, **Dean School of Physical Sciences and Computational Sciences**, for her support andvaluable guidance.

I wish to extend my sincere thanks **Dr. D. ShanmugaPriya, M.Sc., M.Phil., Ph.D.,SET, Head, Department of Information Technology,** for imparting tremendous assistance and well-timed support for triumph of our project.

I heartily thank my esteemed project guide **Dr.T.Jayamalar, M.C.A., M.Phil., Ph.D.,NET, Assistant professor, Department of Information Technology,** for imparting tremendous assistance and well-timed support for triumph of our project.

I would like to express my sincere gratitude to all the staff members of the Department of Information Technology, for their constant encouragement and for the opportunity to do our project in this esteemed university. Last yet importantly, I would like to thank my parents, family members, friends and all well-wishers for their kind inspiration, blessings and encouragement during the course of project.

ABSTRACT

ABSTRACT

Vehicle detection, also known as computer vision object identification, is the study of how machines perceive instead of humans. A vehicle detection system's primary function is to identify one or more vehicles in the input images and real time video. The goal of the study is to find vehicles in images and videos. Google image website real time traffic dataset with the great resources for training the classifiers. The dataset is downloaded from google images website both in format of images and videos to successfully detect, classify and to count the number of vehicles. The dataset is train image processing algorithms designed for detection and tracking tasks. For the building and training of CNN and SVM models, the Python programming language was used as the development language. These image processing systems have been designed, trained, tested, and compared using evaluation metrics in order to pinpoint the flaws and strengths of each. To recognise, track, and count the vehicle in images and videos, the image processing algorithms such as,CNN YOLOV3 and SVM is implemented. The fundamental purpose and objective of the project is to create a system that can automatically recognise and track vehicles in photos and videos, whether they are stationary or moving. The results demonstrated that CNN based YOLOV3 does a good job of detecting and tracking vehicles.

CONTENTS

INDEX AND CONTENTS

LIST OF TABLE

LIST OF FIGURES

INTRODUCTION

CHAPTER-1

INTRODUCTION

1.1 Overview

Vehicle detection procedures on the road are used for vehicle monitoring, counts, traffic analysis, and vehicle classification in a variety of contexts. The main goal and aim of the project is to develop a system that can recognize and track cars in images and videos, whether they are stationary or moving. Vehicle detection and counting is a useful traffic analytics technique that may be used on highways and city streets in a variety of weather and traffic conditions. The system may be permanently established using existing or newly installed video cameras. To identify and categorize automobiles, a variety of approaches and procedures can be employed. CNN YOLOv3 and SVM were used in this study. Contactless detection and classification of numerous vehicle kinds (truck, bus, automobile, bike).To implement YOLOv3 classifier and SVM which are able to predict the class of the image whether it is Vehicles or Non-Vehicles. The algorithms used in these approaches include YOLOv3 and SVM, among others. The CNN algorithm namely YOLOv3 and SVM is used to detect and classify the vehicles in real time images and video. It is hard to manage them using old methods, so the project aim is to detect vehicles in an automated manner.

Now a days population and transportation systems are growing at the same time, the demand for management is growing as well. The world is rapidly becoming populous. The total number of machinery, including vehicles, grew at the same period. A new challenges such as traffic, accidents, and many others must be addressed. Because it is difficult to manage them using traditional techniques, emerging innovations and technologies have been discovered and produced to meet each and every goal that humanity strives to attain. One of these issues is traffic congestion on roads and in cities. Many methods, such as traffic lights and signs, have been used to address these issues.

These choices appear to be insufficient or ineffective on their own. In order to use automated video surveillance to provide data that can give meaning to a decision-

making process, new technologies such as object detection and tracking are devised. This phenomena has been used to a variety of topics. Object detection methods is one of the many parts of the upcoming Intelligent Transportation System (ITS). This technology detects cars, lanes, traffic signs, and vehicle movements. The capacity to recognise and categorise vehicles allows us to enhance traffic flows and roadways, avoid accidents, and track traffic offences and crimes.

Humans can quickly recognise vehicles in videos or images, as well as distinguish between different types of vehicles. The types of data have a significant impact on computer algorithms and programmes. Some obstacles, such as the weather or the amount of light available, might make the procedure easier or more difficult for human to detect the vehicles. At the same time, we have cars of all sorts and shapes. More than that, a new difficulty may be identifying moving objects in real time in a video when their size and form differ.

Vehicle detection and categorization may be accomplished using a variety of approaches and procedures. The algorithms used in these approaches include YOLOv3 and SVM, among others. Since the industry is centered on this system or Computer visionary, the sector is continually changing. The CNN algorithm namely YOLOv3 and SVM is used to detect and classify the vehicles in real time image and video.

1.2 Problem Statement

The problem statement of this project is defined below:

- To detect and count the moving vehicle using CNN and SVM in images and videos.

1.3 Aim of the Study

The goal of this study is to implement two algorithms namely, CNN algorithm using YOLOv3 and SVM, to recognise automobiles in photos and videos with high

accuracy and performance. This project will use the YOLOv3 classifier and SVM to determine if an image belongs to the Vehicles or Non-Vehicles category. If it's a vehicle, determine if it's a bus, truck, car, bicycle, or bike. The vehicle detector will also forecast the cars' bounding box coordinates.

1.4 Significance of the Study

The number of vehicles has been increasing steadily since the industrial revolution. Traffic is a new global challenge. People in cities spend the majority of their time stuck in traffic on their way elsewhere. It is critical for all countries throughout the world to have a digital traffic system that operates 24 hours a day, 7 days a week and makes duties simple and efficient. As a result, a computerised traffic system is ineffective without a reliable vehicle detection system. The economy, residents' lives, industry, and so on are all impacted by the system.

The contribution of the study is to reach a best performance where both the algorithms namely CNN and SVM can correctly recognise all sorts of objects or vehicles in images and videos using pipelines, contributions to the topic are essential.

1.5 Machine learning techniques

The tech transformation has accelerated the emergence of new difficulties that humans must deal with. Rapid technological advancements, human contact with electronic gadgets and diverse technologies, electronic records, and major online development all work together to produce a massive amount of data every second. Over the last two decades, organisations, colleges, scientists, and academics have been working to develop new patterns and innovations to use this data for a variety of reasons, such as detections, recognitions, investigations, differentiating pieces of proof, and making suggestions. Practically every business nowadays uses AI to improve and precision their working procedures and methods, such as in medical, engineering, finance, manufacturing, and so on. A few of these trends is Machine learning in computer technology, as well as man-made consciousness, which is heavily considered and focused by tech experts.

The term Machine Learning (ML) was coined by Arthur Samuel in 1959 and has since been developed and refined by a large number of scholars. ML is a branch of

Artificial Intelligence (AI) that use statistical tactics to provide capacity to computer applications and enable them to gain dynamically from data without the usage of previously defined routines (Koza et al, 1996). By evaluating and recognising instances and relationships between data and events, ML makes hidden data visible. For these reasons, ML employs computer algorithms; for the most part, models are developed to extract information from data and generate predictions based on comparable data. Software developers define traditional applications as functioning within confined and limited rules. Because constructing information-driven applications like computer visions or email sorting is almost impossible with typical programming strategies, algorithms' advancement and self-learning qualities encourage them to overcome traditional applications. These algorithms assist us in making better selections and provide consistency. The three primary types of ML are as follows:

- Supervised
- Unsupervised
- Reinforcement

1.5.1 Supervised Learning

For the most part, supervised learning is used to solve even-minded AI challenges. In supervised learning, there are two factors: one for inputs and one for outputs. Through a mapping task, the algorithms are aiming to master mapping between these factors (Russell and Norvig, 2016). The algorithms are built with data sources in mind, and the outputs are considered. As a coach or educator, we screen the learning during the training stage.

When the algorithms predict the output, they will verify whether the appropriate answer is accurate or incorrect, close to the output or far away. These techniques aid in the learning and improvement of algorithms. When the learning method reaches a satisfactory degree of accuracy, it is completed. When new data is received, it attempts to predict the output based on previous learnings and an estimated mapping between information inputs and outputs.

The last part examines supervised learning techniques in the categories of regression and classification. There are also doubts that these algorithms will be operational after the data has been categorised. Data is no longer free, and social event information for learning may be costly as a result of its use as another oil on the planet. A large number of supervised algorithms have been tested for the common sense element. Each of them has their own set of strengths and weaknesses. In machine learning, there is no one algorithm that performs best for all jobs. As a result, selecting an algorithm is an important issue that everyone working in machine learning should consider.

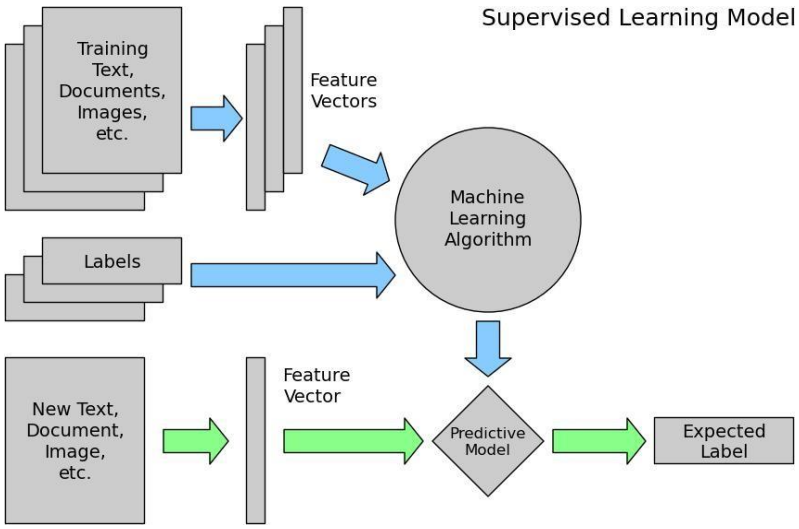


Figure 1.1: Diagram of the Supervised Learning

The following are some of the most commonly used managed algorithms:

- Support Vector Machines
- Naïve Bayes
- Logistic regression
- Linear regression
- Linear discriminant analysis
- Decision trees

- k-nearest neighbor
- Neural Networks

1.5.2 Unsupervised Learning

Unsupervised algorithms, unlike supervised algorithms, do not have correct solutions. There is no output variable and no guide or teacher to correct errors. The algorithms try to understand the information characteristics. They look for hidden and hidden instances in the dataset to predict the outcome based on the input variables alone. They have no names to learn and develop their prediction abilities (OFOR, 2018). Clustering and association problems collect unsupervised learnings.

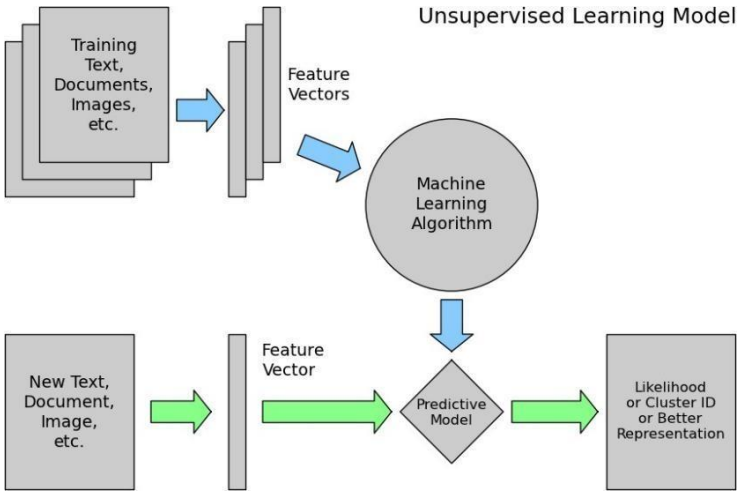


Figure 1.2: Unsupervised learning model diagram

Clustering: In this type of work, information is divided into groups, such as categorising clients based on their purchase habits.

Association: Algorithms are seeking to understand the criteria that may explicate the vast quantity of data, for example, if a customer buys a shirt, they are likely to buy pants as well.

To offer you some instances of unsupervised learning computations, consider the following:

- Apriori algorithm connection
- K-means clustering

1.5.3 Reinforcement Learning

Reinforcement learning functions similarly to how a kid learns in the early stages of life. When a child does admirably, the individual will be encouraged, and when a child performs poorly, the conclusion will be discipline or advice to avoid repeating the mistake. These algorithms are performing the same duty as an agent who acts as a youngster. The agent cooperates with the environment, and it is rewarded for properly doing tasks and punished for incorrectly executing them. When an agent is attempting to increase the awards while limiting the penalties.

Consider a self-driving vehicle: if it arrived at its destination without causing any accidents, exiting the roadway, or making abrupt stops, it will be rewarded, but if it did any of the above, it will be punished. As a result, the car will not cause the same accidents for which it was previously punished. These algorithms are also known as dynamic programming, and a great lot of research and development is being done to enhance them so that they may be used for a wide range of jobs in the near future.

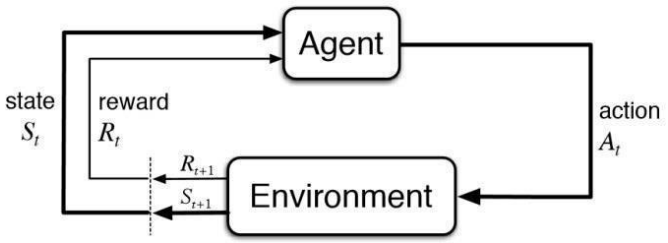


Figure 1.3: Reinforcement Learning model

1.5.4 Used Machine Learning Techniques

On the same dataset discussed before, the author built and developed two algorithms from the supervised category in this research. The following are the algorithms:

- Support Vector Machine (SVM)
- Decision Trees

a) Support Vector Machine (SVM)

This is a supervised machine learning approach that may be used to classify and predict data. In SVM, the main technique is to find a hyperplane that divides the dataset into two groups, as shown in Figure 1.4.

These are data points that are near to the hyperplane; for example, removing a point from the dataset will affect several things, including the role of the hyperplane splitter. This is one of the most crucial aspects of datasets.

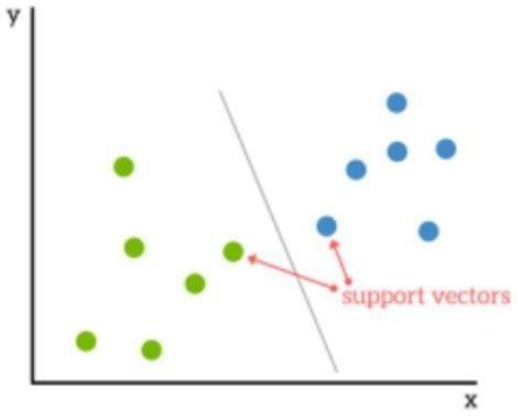


Figure 1.4: SVM splittion

b) CNN (using YoloV3)

A convolutional neural network (CNN/ConvNet) is a type of deep neural network used to evaluate visual images and real time video in deep learning. When we think about neural networks, we usually think of matrix multiplications, but this isn't the case with ConvNet. It employs a method known as Convolution. Multiple layers of artificial neurons make up convolutional neural networks. Artificial neurons are computational models that calculate the weighted sum of many inputs and output an activation value. Each layer creates many activation functions that are passed on to the next layer when you input an image into a ConvNet. The classification layer generates a series of confidence ratings (numbers between 0 and 1) that indicate how probable the picture is to belong to a "class" based on the activation map of the final convolution layer. If your ConvNet identifies cats, dogs, and horses, the last layer's output is the likelihood that the input image includes any of those animals.

You Only Look Once, Version 3 (YOLOv3) is a real-time object detection system that recognises specific things in films, live feeds, and photos. To detect an item, YOLO employs features learnt by a deep convolutional neural network.A convolution neural network is YOLO. There are 24 convolutional layers in total, followed by two fully linked layers. Each layer is significant in its own right, and the layers are distinguished by their functions.

— The ImageNet dataset, which is a 1000-class classification dataset, is used to pre-train the first 20 convolutional layers, followed by an average pooling layer and a fully connected layer.

— The dataset with the picture resolution of 224 x 224 3 is used for classification pretraining.

Three convolutional layers and one reduction layer make up the layers.

— To train the network for object detection, the last four convolutional layers are added, followed by two fully connected layers.

— Because object recognition necessitates more precision, the dataset's resolution is raised to 448 x 448.

— Finally, the bounding boxes and class probabilities are predicted by the final layer.

— The other convolutional layers all employ leaky ReLU activation, however the final layer uses linear activation.

— The input is a 448 × 448 picture, and the outcome is the identified object's class prediction within the bounding box.

1.6 Tools Used

This proposed system, like all other research and studies, used some software and tools to create models and tests. Various tools are required for such studies, such as the author's choice of Python as the model development programming language, the dataset containing a large number of vehicle as well as non-vehicle images used for training, or the many Python packages that are required or beneficial in order to generate ML models. This chapter introduces all of the tools that will be utilised in the proposed methodology.

1.6.1 Python

Python is a Guido van Rossum-created general-purpose programming language that is used for a variety of platforms including mathematics, computer GUIs, the web, and a variety of significant scientific applications. Python's fundamental goal was to make programming simple so that anybody in the world could create code. As a result, it is well-known for its simplicity. Python is a space-sensitive language. Ignoring the fact that Python was designed for children, it currently supports all programming languages in a wide range of industries, which is both astounding and incredible. Python can accomplish whatever that other programming languages can. Everything from the web to algorithms to desktop apps is covered.

Python has gained popularity in artificial intelligence and machine learning applications in recent years owing to its abundance of efficient and useful libraries that make the process much easier and faster. Experts in this discipline come from a variety of backgrounds. Python is the easiest and most convenient language to start with if they have no programming experience.

Despite the foregoing characteristics, the researcher's knowledge and self-interest prompted him to chose Python as the model development programming language. The following are the libraries that were used in this project.

a) Numpy

Numpy is a free and open source framework that uses multi-dimensional matrices and arrays to do computations. It has a variety of functions that make working with this sort of data simple. Arrays are used in data analysis to make it faster and more

efficient. As a result, this library aids data scientists in working with vast volumes of data. The models function often requires arrays as a parameter for detection and forecasting to operate quickly and reduce training prediction time.

b) Matplotlib

Plotting is becoming more popular in all industries as a way to see and analyse data. As a result, Matplotlib is used as a plotting library to construct various graphs and figures for a range of purposes. Matplotlib has the advantage of producing nice plots and graphs with only a few lines of code. To extract colour characteristics and construct histograms, matplotlib is utilised.

1.6.2 JupyterNotebook

Jupyter Notebook is an online tool that allows us to write and edit live scripts, equations, plaintexts, and visualisations. This is a free notepad that supports a variety of programming languages. This is used for a variety of things, including machine learning, numerical simulation, and data visualisation. This notebook was chosen by the researcher for creating legible code and applying machine learning algorithms.

1.6.3 Computer

The PC that is being used to train and test the models has the following properties:

Model: Hp

RAM: 16 GB

Processor: Core i7 Quad Core Graphic: Intel HD8 GB

LITERATURE REVIEW

CHAPTER-2

LITERATURE REVIEW

Vehicle identification and tracking have piqued experts' interest in recent decades. The subject received a lot of interest. Various sensory modalities have been utilised to identify things, particularly cars. LIDAR, radar, and computer vision are the three modalities. The allure created by enormous advances in picture processing. The earliest indicators and models of image processing are from the 1960s and 1970s, following which a variety of approaches and techniques were devised and suggested [6]. This chapter briefly reviews recent studies on vehicle identification and tracking by researchers.

Object detection describes the characteristics of an object that allow a computer to distinguish it from other things. Meanwhile, object categorization aims to determine if an object shares any similarities with another object category. We may assign an object tracker based on the object detection result. By re-detecting the target in the sequence frame after the first point of the target, the tracker is tracking it.

The project' main purpose and objective is to create a system that can automatically recognise and track vehicles in photos and videos, whether they are stationary or moving.The vehicle detection system [34] is to find one or more autos in input image using sliding windows. In vehicle detection systems, the sliding window technique is employed in two ways. According to [11], the vehicle detection system applies local characteristics using sliding window and Heatmap.

The capacity to identify the vehicle using the sliding window and HOG, according to [19]. The dataset is the image of unmanned aerial vehicle. The supplied image may be shown in many windows of varied sizes. It then checks to see if the target is within the window. The edge detection can be applied to discover objects in UAV photos [30] and track the vehicles. It shows how to use edge detection to identify fake things. To discover straight lines on a vehicle, this method use an edge detection technique. With the help of a larger threshold, the software suppresses background noises first.

According to [37], the colour characteristics and edge detection are used to separate and differentiate manufactured and natural objects. The difference is in collecting the nine attributes from the numerous colour channels of the source picture. These qualities are utilised to distinguish between manufactured and natural items by defining the edges and changes. One set of ethereal geo-referenced images and another set of non-geo-referenced aircraft photographs shot in the same place at different times are used [2]. The extraction of shadow and surface highlights are employed to construct a milestone recognition framework. These attributes are used in the recommended approach to offer information about surface introduction, shape, and shading.

The vehicle identifying thickness estimation approach is proposed by [8]. The angle vectors were established in the edge guide of the raised images. At the objective's limit, the heads of the inclination vectors should fundamentally alter, and the standard deviation of the slope vectors should be computed. As a consequence, by predefining the boundary, the vehicles may be detected inside the esteem. The researchers used aerial photographs obtained from the street in Turkey to get an F-measure accuracy of 86%. However, without the preparation, such discovery techniques suffer from a typical flaw: it's impossible to tell the target objects apart from the perplexing foundation conditions.

According to [31], a basic extractor enhancement strategy based on vehicle identification is offered. This approach employed asphalt division with 8-neighborhood fills to remove the street signs and separate the unanticipated foundation. The findings imply that the suggested method can prevent identification mistakes due by erroneous roadway markers. In any case, this identification approach is dependent on fixed camera discovery, which is counter to the theory's purpose, but the foundation extractor system might be utilised instead.It offered a two-step approach to dealing with the pre-programmed position of cars inside aeronautical symbols using aerial images [20].

According to [28], Haar classifiers are used for vehicle layout and an auxiliary check mechanism that uses UAV elevation and vehicle measurement restrictions to try to reject non-vehicle candidates. The Haar classifier generates a dependable location that is independent of vehicle shading, type, or configuration. To accomplish vehicle

introduction invariance, they employ four separate fall Haar classifiers created in test vehicle photographs organised into four positioned introductions.

It is observed that utilizing more warm symbolism for warm mark affirmation, which greatly enhanced execution [4].These authors studied the execution of two image fix descriptors, an altered Histogram of Oriented Angles (HoG) highlight and Histogram of Gabor Coefficients highlights. They looked examined K-Nearest Neighbors (k-NN), Random Forests (RF), and Support Vector Machines as factual grouping techniques (SVM). In the initial round of their calculations, they got an average localization rate of 85 percent and revealed that Random Forests with Histogram of Gabor Coefficients highlights was the highest performing classifier, properly identifying 98.9% of automobiles and 61.9 % of images.

According to [5], the vehicle localization in provincial circumstances is implemented using two stages. In the first stage, a Harris corner indicator is employed to indicate places of interest for the images. A successful sliding window approach is then implemented. Covering areas are gathered and refined based on the shading attributes of foundation zones. In this stage, the zones identified in the previous phase are subjected to picture grouping algorithms, which determine the closeness of a vehicle. With cutting-edge object identification algorithms, the sliding window characterization approach with SVM classifiers is applied to provide a vehicle localisation technique in unconstrained conditions [14].

According to [16], a nearby component-based technique for programmed vehicle recognition in high symbolism with low aims is proposed. Their technique was designed with the goal of removing the constraints associated with positioning tactics based on a vehicle's outward appearance, such as the rectangular form of the car and the proximity of frontal and back windows. Their strategy is based on extracting Scale-Invariant Feature Transform (SIFT) highlights from vehicle and foundation images. These features are used to describe a model that may be used to organise SIFT highlights isolated from the cars and foundation in a query picture using a Support Vector Machine (SVM) classifier. The SIFT highlights that are expected to have a place with a vehicle are then organised in the 2D image space into subsets that

are linked to specific cars. The bunching technique used by the developers is based on a modified Liking Propagation (LP) computation that is constrained by the spatial constraints imposed by the geometry of vehicles at the provided targets. In aeronautical symbolism of a parking garage with 105 vehicles, they achieved a grouping exactness of 95.2 percent, with no false-positive recognitions.

S. No	Title of the paper	Author	Techniques used	Observation
1.	Vehicle detection in remote sensing imagery based on salient information and local shape feature'	Yu & Shi, 2019	Sliding window approach	The basic task of a vehicle detection system, according to this research, is to locate one or more automobiles in input photos. The sliding window approach is used in two ways in vehicle detecting systems.
2.	Adaptive Sliding-Window Strategy for Vehicle Detection in Highway Environments	Noh et al,2019	vehicle detection system using sliding window and Heatmap	The basic task of a vehicle detection system, according to this research, is to locate one or more automobiles in input photos. In a

					vehicle detection system that uses local characteristics, there are two techniques.
	3.	Online cascaded boosting with histogram of orient gradient features for car detection from unmanned aerial vehicle images	Su et al,2020	Sliding windows, HOG	Its ability to solelyrecognise pre-learned things might be a weakness or disadvantage. The second approach, which is based on sliding windows, operates differently. It may display the input image in several windows of various sizes.It then determines whether or not the target is within the window.
	4.	Vision-based detection and tracking of a mobile	Wang et al, 2021	Edge detection algorithm	In UAV footage, use edge detection to find the objects. A researcher

	ground target using a fixed-wing UAV			demonstrates how to distinguish fake items using edge detection. This approach uses an edge detection algorithm to find straight lines on a vehicle. The programme first reduces background noises with the aid of a higher threshold.
5.	Automatic salient object detection in uav imagery	Sokalski et al, 2021	colour features and edge detection	It separates and distinguishes artificial and natural things using colour features and edge detection. The distinction lies in obtaining the nine characteristics from the original image's multiple colour channels. These characteristics are used to define

				the edges and changes that distinguish artificial and natural things.
6.	Real-Time Obstacle Detection Approach using Stereoscopic Images	Baha,2021	Extraction of shade and surface	This study makes use of one set of ethereal geo-referenced photographs and another set of non-geo-referenced aviation photographs taken in the same region at a different time. The extraction of shade and surface highlights is used in this study to create a milestone acknowledgemen t framework. The suggested technique makes use of these features to provide information on surface

				introduction, shape, and shading.
7.	Survey of unmanned aerial vehicles (UAVs) for traffic monitoring	Kanistras et al, 2022	Thickness estimate method	It is identified that the cars may be identified inside the esteem by predefining the edge. The study used aerial photos taken from the street in Turkey and achieved an F-measure accuracy of 86 percent. However, without the preparation, such discovery procedures have a common drawback: the target items are difficult to distinguish from the puzzling foundation circumstances.
8.	A highway vehicle detection	Wei et al, 2021	extractor improvement approach using	It is suggested that the proposed technique is

	method based on the improved visual background extractor		8-neighborhood fillings	capable of avoiding identification errors caused by false street markings. In any event, this identification technique is based on fixed camera finding, which goes against the theory's goal.
9.	Region-based automatic mapping of tsunami-damaged buildings using multi-temporal aerial images	Susaki, 2020	Two-step approach using HOG and Heatmap	It offered a two-step approach to dealing with the pre-programmed position of cars inside aeronautical symbols.
10.	Detecting pedestrians using patterns of motion and appearance	Viola et al, 2020	Haar classifiers	They use four different fall Haar classifiers generated in test vehicle photos grouped into four positioned introductions to

				achieve vehicle introduction invariance.
11.	Thermal and narrow band multi spectral remote sensing for vegetation monitoring from an unmanned aerial vehicle	Berni et al, 2020	warm symbolism	Later, expanded this by using more warm symbolism for warm mark affirmation, which greatly enhanced execution.
12.	Robust vehicle tracking and detection from UAVs	Chen & meng, 2020	Harris corner indication, HoG, Histogram of Gabor Coefficients highlights, K-NearestNeighbors(k-NN), Random Forests (RF), and Support Vector Machines	They obtained an average location rate of 85 percent in the first phase of their calculations and discovered that Random Forests with Histogram of Gabor Coefficients highlights was the best performing classifier, capable of accurately characterising

				98.9% of vehicles and 61.9 percent of foundation pictures.
14.	Vehicle detection in aerial imagery: A small target detection benchmark	Razakarivony, 2021	sliding window, SVM	The sliding window characterisation procedure with SVM classifiers is used to offer a vehicle localization technique under unconstrained situations with cutting-edge object recognition algorithms.
15.	Robust vehicle detection in low-resolution aerial imagery	Sahli et al, 2021	Nearby component-based technique, Scale-Invariant Feature Transform, SVM, Liking Propagation	The bunching technique used by the developers is based on a modified Liking Propagation (LP) computation that is constrained by the spatial constraints imposed by the geometry of vehicles at the

					provided targets. In aeronautical symbolism of a parking garage with 105 vehicles, they achieved a grouping exactness of 95.2 percent, with no false-positive recognitions.	

The literature study in various field of vehicle detection and tracking is summarized in below table:

Table 2.1: Literature study

METHODOLOGY

CHAPTER 3

METHODOLOGY

All strategies and technologies for detecting and tracking vehicles are explained in this chapter. The techniques used are outlined in depth at the outset, including how they are employed in the context. Following that, the methodology includes image gathering, image pre-processing, and image classification algorithms. The image pre-processing includes the histogram of gradient (HOG), heat map, sliding window and image extraction to get the vehicle region in the input real image and video. Figure 3.1 depicts the full vehicle detection and tracking procedure. To finish each phase, only a few activities and tasks were required. The flow diagram depicts the complete procedure in detail.

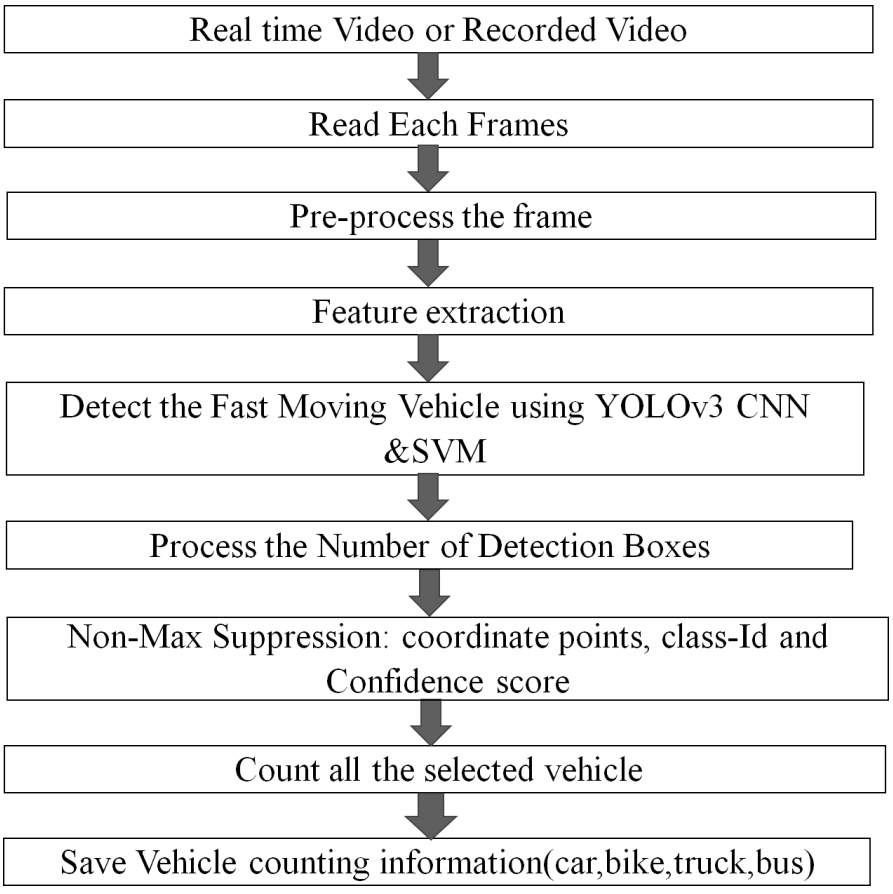


Figure 3.1: Model development summary

From the above figure it is observed that the real time video and images is used as input data for vehicle detection. The input video is read as per frames and preprocessing is done in each frames. This includes HOG, heat map and sliding window to extract the features that include vehicle information. Using the extracted information, the two model will be trained namely CNN and SVM. After model training, the vehicle is detected. The detected vehicles are again processed to draw the box around it to track the vehicle in the input video. By applying the non-max suppression the vehicles are counted based on class-Id, coordinates of vehicle box and confidence scores. The calculated vehicle count is stored as a result in CSV format.

3.1 Datasets

The real-time traffic statistics on the Google image page provides excellent resources for training classifiers. Figure 3.2 shows vehicle examples gathered from the Google website for training purposes. To properly identify, categorise, and count the number of vehicles, the dataset is acquired from Google Images in both image and video format.

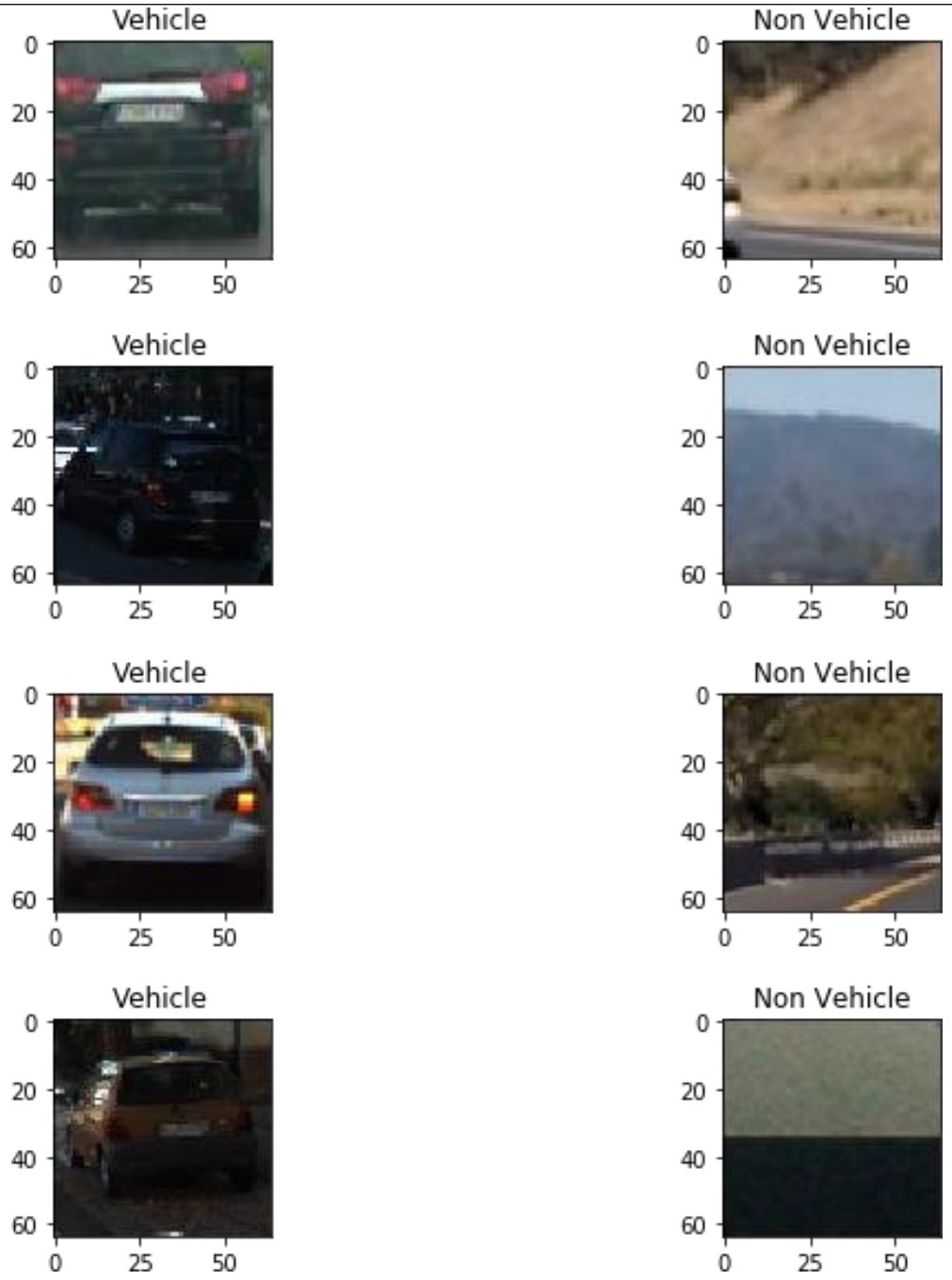


Figure 3.2: Vehicle and Non-Vehicle images

3.2 Image preprocessing

In order to operate a car safely, self-driving systems require vehicle detection and tracking. The purpose of this project is to create a software pipeline that can recognise automobiles in video.

It may be accomplished by doing the following tasks:

• On a labelled training set of photos, extract Histogram of Oriented Gradients (HOG) features and train a Linear SVM classifier.

• Search for automobiles in photos using a sliding-window approach and your learned classifier.

• Run the pipeline on a video stream and produce a frame-by-frame heatmap of recurrent detections to eliminate outliers and track identified cars.

• Calculate a bounding box for identified cars.

The photos of vehicles and non-vehicles are loaded into separate lists using Numpy arrays. To train the classifier to recognise automobiles efficiently, multiple feature extraction approaches were applied. While including three colour channels in a full resolution image may be inconvenient, spatial binning allows you to preserve enough information to aid in vehicle detection.

The vehicle is still easily recognisable by sight at 32 x 32 pixel resolution, as shown in the code in the appendix, implying that the key characteristics are maintained at this level. OpenCV's cv2.resize function is useful for scaling down an image's resolution (). The ravel() function in numpy may be used to convert this to a one-dimensional feature vector.

In order to operate a car safely, self-driving systems require vehicle recognition, categorization, and tracking. The main purpose of this project is to create a software pipeline for detecting automobiles in video and picture datasets.

3.2.1 Color histogram

In photography, a histogram is a graphical representation of the number of pixels in a picture that fall inside a given brightness or color range. The number of pixels for each luminance or brightness level from black to white is displayed in a conventional luminance histogram, for example. The bigger the peak on the graph, the more pixels at that brightness level. A color histogram works on the same principle, but instead of seeing degrees of black graphed, you'll see the number of pixels for each of the three major colors. A color histogram is a histogram that shows the color intensity of each RGB color channel separately.The boxes drawn around the automobiles are becoming

increasingly precise. The heatmap is then applied to each box to increase the pixel count by one. The image must then be given a threshold value to filter out low pixel cells, locate pixels associated with each car number, and construct the final bounding boxes.

3.2.2 Histogram of Oriented Gradients (HOG)

A function descriptor is a prospective of offering an photo or an picture spot that streamlines it by drawing out one of the most important stats and eliminating the remainder. The technique matters the variety of times a gradient positioning shows up in a certain area of a picture. The HOG work descriptor utilizes gradient path distributions (histograms) as functions. Since gradients have an excellent amplitude near to sides and edges, they are testing to see (areas of sudden deepness modifications). Gradients (x and y by-products) in an photo are significant because of the truth they include some range additional information regarding the geometry of an item compared to level areas. The complying with action is to select the appropriate specifications for educating the classifier to anticipate the vehicles in the input picture.

3.3 Classifiers

In this study, two different classifiers were employed.

• Provide assistance to vector machines (SVMs)

• CNN's use of YoloV3

SVMs are a class of supervised learning algorithms for classification, regression, and outlier identification. In this project, I've opted to utilise LinearSVC as a classifier. CNN is a supervised machine learning technique that is mostly used for data classification. The details about both the classifiers are explained below.

a) SVM

SVM stands for Support Vector Machine and is one of the most widely used Supervised Learning algorithms for Classification and Regression issues. However, it is mostly utilised in Machine Learning for Classification difficulties. The SVM algorithm's purpose is to find the optimum line or decision boundary for categorising n-dimensional space into classes so that additional data points may be readily placed

in the proper category in the future. A hyperplane is the name for the optimal choice boundary.

The extreme points/vectors that assist create the hyperplane are chosen via SVM. Support vectors are the extreme instances, and the method is called a Support Vector Machine. Face identification, image classification, text categorization, and other tasks may all benefit from the SVM algorithm. There are two types of SVM. They are explained below:

- Linear SVM is a classifier that is used for linearly separable data, which implies that if a dataset can be categorised into two classes using a single straight line, it is called linearly separable data, and the classifier is named Linear SVM.

- Non-linear SVM is a classifier that is used for non-linearly separated data, which implies that if a dataset can't be categorised using a straight line, it's non-linear data, and the classifier employed is called Non-linear SVM.

In this project, Linear SVM is used to classify and detect the vehicle and non-vehicle using SVM hyperplane. The working of SVM algorithm using Linear SVM is explained below. The dataset with two labels namely vehicle and non-vehicle with features. We need a classifier that can categorise the pair of coordinates (x1, x2) as vehicle or non-vehicle.

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes.

As a result, the SVM method aids in identifying the optimal line or decision boundary, which is referred to as a hyperplane. The SVM method determines the intersection of both classes' lines. Support vectors are the names given to these locations. Margin is the distance between the vectors and the hyperplane. The purpose of SVM is to increase this margin as much as possible. The ideal hyperplane is the one with the greatest margin.

The Algorithmic step of SVM is mentioned below:

- Data Pre-processing using min_max_scaler() and train_test_split() to split the dataset into train and test data.

- Fitting the SVM classifier to the train data
- Predicting the test data result
- Creating the confusion matrix
- Visualizing the prediction result using bar chart and table

b) CNN

Artificial Neural Networks are employed in picture, audio, and word categorization applications. For example, we utilise Recurrent Neural Networks, more exactly an LSTM, for predicting the sequence of words, and we use Convolution Neural Networks for picture categorization. Convolutional neural networks, often known as covnets, are neural networks with shared parameters. For example, the input image may be seen as a cuboid with length, breadth (image size), and height (as images generally have red, green, and blue channels). Extracting the little part of this image and running a little neural network with k outputs on it, then representing the results vertically. Slide that neural network across the whole image, and a new image with changing width, height, and depth is created. More channels are acquired instead of merely R, G, and B channels, albeit with a smaller width and height. Convolution is the name for this procedure. It's a normal neural network if the patch size is the same as the picture size. Because of the compact patch, there are fewer weights. Some of the process involved in CNN algorithm is listed below:

- A series of learnable filters makes up convolution-layers (a patch in the above image). Every filter has the same depth as the input volume and a modest width and height (3 if the input layer is image input).

- For example, the convolution on a 34x34x3 picture is carried out. Filters can be as little as (a * a * 3), where 'a' can be 3, 5, 7, or even smaller than the picture dimension.

- During forward pass, the dot product between the weights of filters and patch from input volume is computed by sliding each filter over the whole input volume step by step (each step is called stride and may have a value of 2 or 3 or even 4 for high dimensional pictures).

- Each filter's 2-D output is captured and stored together, resulting in an output volume with a depth equal to the number of filters. All of the filters are learned by the network.

A covnet is a series of layers, each of which uses a differentiable function to change one volume into another.

- Input layer: This layer stores the image's raw input
- Convolution layer: The output volume is computed by computing the dot product between all filters and picture patches in the convolution layer.
- Activation Function Layer: It applies an element-wise activation function to the convolution layer's output. RELU: max(0, x), Sigmoid: 1/(1+e-x), Tanh, Leaky RELU, and others are frequent activation functions. In this project, sigmoid is applied as activation function.
- Pool Layer: This layer is injected into the covnets on a regular basis, and its major job is to minimise the volume size, which speeds up computation, saves memory, and avoids overfitting. Max pooling and average pooling are two typical types of pooling layers.
- Fully-Connected Layer: This is a standard neural network layer that accepts input from the preceding layer, computes class scores, and outputs a 1-D array with the same size as the number of classes.

The algorithmic steps of CNN are explained below:

- Prepare dataset for training
- The vehicle and non-vehicle dataset is splitted into train and test data with the ratio of 80% and 20%.
- The training dataset is shuffled using random_seed in train_test_split()
- Assigning labels and features i.e., the label namely vehicle and non-vehicle as 0 and 1

- Homogenizing X and converting labels of training data into categorical data using min_max_scaler()

- Split X and Y for use in CNN.

- Define, collect and train the CNN model using train and test data

- Accuracy and score of model is calculated

3.3.1 Train and Test Split

The StandardScaler() system assumes that data within each point is regularly distributed and scales it so that it's now centred around 0 with a standard divagation of 1. The function transforms the' x' values to produce the gauged X affair.

There are a few libraries that can assist you partition the dataset. One of these is the 'sklearn' function 'train test split,' which helps split the dataset into train and test data for the classifier.

3.4 Sliding window

A sliding window is a thickish cell of defined range and height that" slides" over an image in computer vision(as the name implies). We would naturally take the window region and use an image classifier to see if the window contains an object that we are interested in. We can see all the bounding boxes for where my classifier reported positive findings in these three test prints. Each of the two motorcars has lapping findings, and there is a false positive discovery in the centre of the road in two of the frames. A heat- map is created from these findings in this step in order to aggregate imbrication findings and count false cons.

Heatmap and threshold limit are used to integrate imbrication findings and count false cons.

For the sliding window approach, hog sub-sampling is a more efficient option. The code only has to extract hog characteristics once, after which it may be subsampled to obtain all of the overlying windows. Each window is determined by a scaling factor, with a scale of 1 resulting in an 8 × 8 cell window. Each window's overlap is measured in cell distance. This indicates that with cells per step = 2, the search

window overlap would be 75%. To produce multiple-scaled search windows, execute the same method many times for different scale values. The hog subsampling reduces the time it takes to calculate HOG features, resulting in a greater throughput rate.

To achieve a more accurate result, I changed the stating position of the window search from 350px to 656px and lowered cells per step to one.

As previously stated, overlapping detections are combined and false positives are removed using the same heatmap and threshold with limit 1 approach.

3.5 Pipeline video

Finally, produce the pipeline movie by using the preceding approaches to process each frame of the picture and then creating the video from the processed frames.

The discovery buses overeater sub system excerpts all bounding boxes plant in the picture for the buses. The heat threshold function is used to integrate overlap findings, reduce false cons, and induce an affair with a bounding box.

RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Experimental Setup

There are various programming languages to pick from when constructing Machine Learning algorithms, such as MATLAB, Python, and R. Each alternative has its own set of benefits and advantages. As a result, the researcher has opted to create in Python due to its ease of use and extensive library of tools for various tasks. Numpy was used to import and analyse data, Matplotlib was used to visualise the extracted colour features, Scikit-Learn was used to partition data into train and test sections, and classifier functions were used to create models and train them using available data in the datasets.

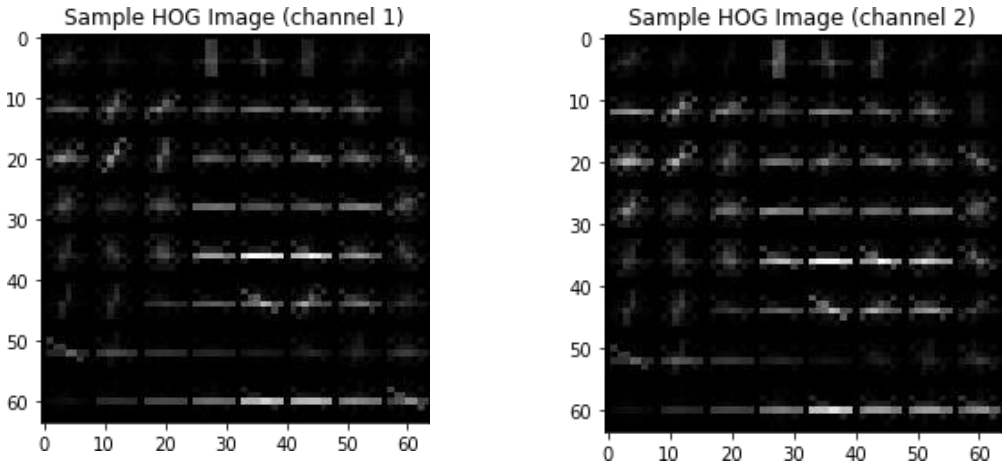
4.2 Implementation

As previously said, there are a number of libraries that can assist us in quickly implementing the models. These libraries must first be loaded into the notebook. The dataset had to then be read and loaded into arrays for processing. These tasks were completed using Numpy.

A histogram is an exact description of the dispersion of numerical information. It's a hand of histogram is a precise representation of the distribution of numerical data. Karl Pearson introduced it as a measure of the probability rotation of a nonstop variable(CORAL). It differs from a bar graph in that a bar illustration connects two factors, whereas a histogram connects just one. To produce a histogram, first" caddy"(or" pail") the range of characteristics — that is, resolve the whole range of rates into a series of intervals — and also count how numerous rates fall into each interval. The holders are generally determined asnon- covering intervals of a variable that are repeated. The holders(intervals) must be blob joined and are generally(but not always) of the same size.The data is binned once the colour feature has been extracted. "This modification constructs a 1-, 2-, or 3-dimensional grid spanning the simulation region and assigns each particle to one of the evenly sized bins," says the description.

It then applies a reduction operation to a specified particle attribute, mapping all of the particles in a cell to a single output value. This modifier may be used to project per-particle data to a structured grid, such as to coarse-grain atomistic data and construct a continuous field representation of a particle attribute. Different reduction processes, such as total, average(mean), minimum, and maximum, are available. The bin grid can be one-, two-, or three-dimensional, which means that the simulation domain can be split into equal-sized bins along one, two, or all three axes. The simulation cell borders are always parallel to the spatial bins." (2015, Chen)

The histogram of directed gradients is then obtained. "A feature descriptor used in computer vision and image processing for object recognition is the histogram of oriented gradients (HOG)." The approach counts the number of times a gradient orientation appears in a certain area of a picture. This approach is comparable to edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but it is distinguished by the fact that it is computed on a dense grid of evenly spaced cells and employs overlapping local contrast normalisation for enhanced accuracy." Figure 4.1 shows the HOG extraction results, as well as the following:



(c) HOG with channel 1 (d) HOG with channel 2

Figure 4.1: HOG features extraction from one sample of the vehicles

The following phase is data pre-processing; before training any algorithm, we must prepare data, clean it up, scale it, and so on. There is no need to clean this dataset because it is an imaging dataset. As a result, data splitting is required at this stage.

The train test split() method from the Scikit-Learn package divides the data into two parts: training and testing. In general, the percentages for both are estimated to be around 80% for training and 20% for testing. In addition, the data must be standardised. The StandardScaler() method was used to standardise the data in the SVC classifier operation. After all of these steps have been completed, the SVM classifier has been trained, and the results of the methods are shown in table 4.1.

Parameters	**Values**
Time	1.009 second
Accuracy	0.985
Error	0.9%

Table 4.1:SVM classifier training results

In the table, it is observed that the algorithm is trained in 0.9 seconds and has a 98.5 percent accuracy. In evaluation matrices, with those values. The sliding window is the next job after the model has been trained using the training portion of the dataset. Sliding window functionalities are available. Following the sliding, the system required to locate the windows on which the classifier would be run. The function that returns the refined windows where the classifier predicts an vehicles as the result. Then you must execute the procedure that draws the main window around the detected automobiles. Figure 4.2 shows the sliding window and the recognised vehicles in the drawn window surrounding the car in the images.

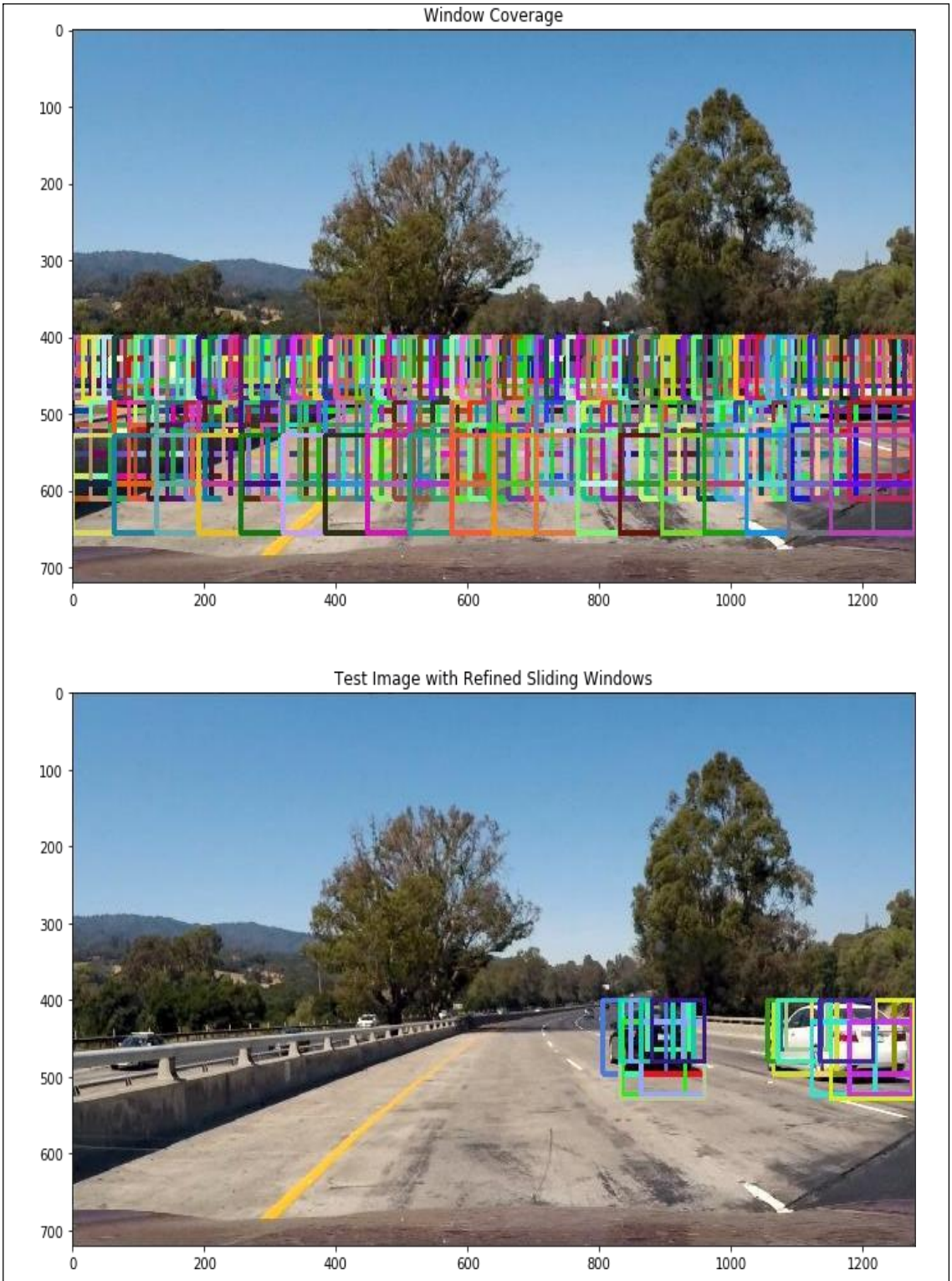


Figure 4.2: Sliding window with the refined sliding windows

The drawing of boxes around the autos is becoming more exact. After that, the heatmap is applied to raise the pixel count by one inside each box. Then we must apply a threshold value to the image in order to filter out low pixel cells, locate pixels

with each car number, and construct the final bounding boxes. Figure 4.3 and figure 4.4 depicts the testing heatmap pictures.

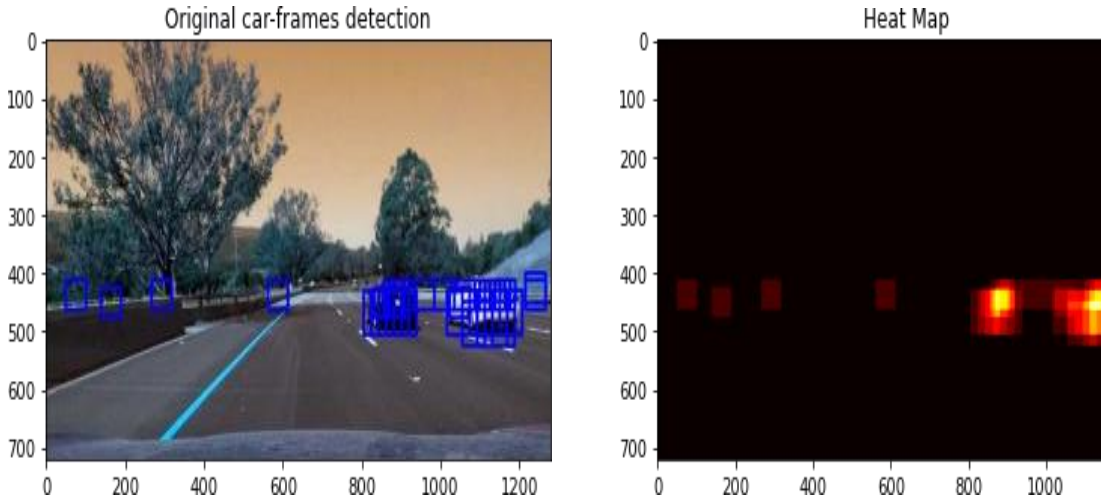


Figure 4.3: HeatMap on testing image (before threshold)

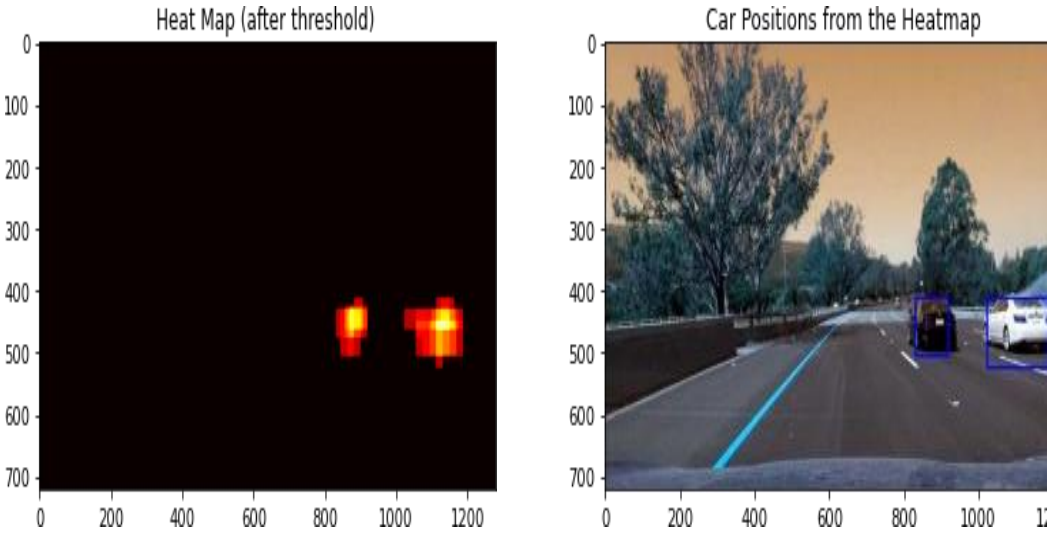


Figure 4.4: HeatMap on testing image(after threshold)

The models are now detecting automobiles and drawing boxes around them. The primary work has been completed. Although the technology is capable of detecting autos, it is unable to track them in video. A pipeline had to be defined in order to do this. To load a video, the pipeline function Object() { [native code] } function requires numerous arguments. The pipeline feature is now available. Figure 4.5 shows an example of the pipeline outcome on Test pictures.

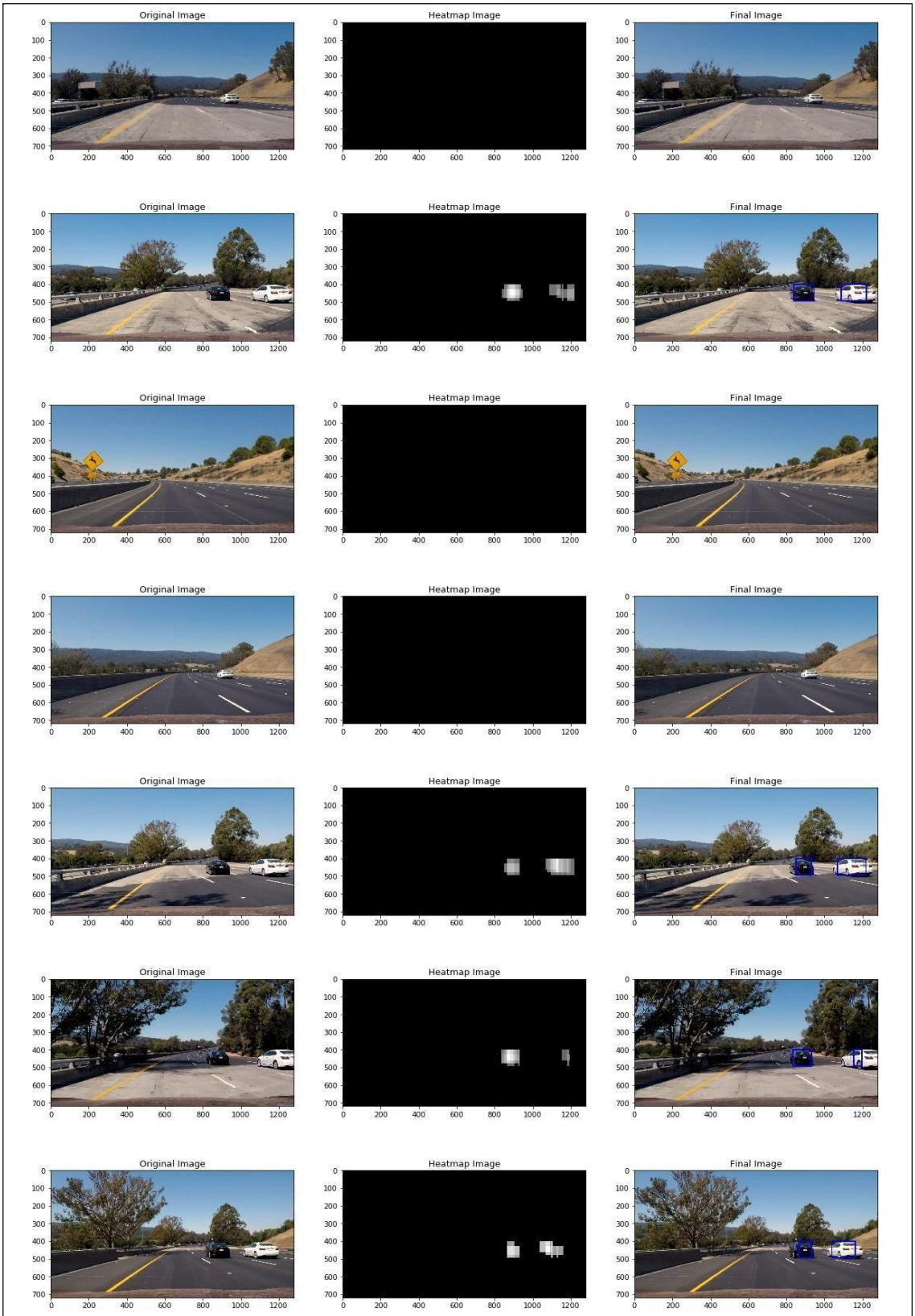


Figure 4.5: Pipeline sample on test images

Localizers or classifiers are used by prior detection systems to carry out the detection procedure. The model is then used to apply to a picture at various sizes and places. For detections, portions of the picture with high scores are examined.

The YOLO algorithm takes a different method. A single neural network is applied to the entire image using the method. The network then splits the picture into areas, generating bounding boxes and predicting probabilities for each. The projected probabilities are used to weight the produced bounding boxes.

The below figure 4.6 shows the counting of vehicles in images.

A	B	C	D	E
test1.jpg	6	0	0	0
test2.jpg	7	0	0	0
test3.jpg	9	0	0	0
t1.jpg	60	0	0	10
t2.jpg	103	6	3	12
test1.jpg	109	6	3	12
test1.jpg	6	0	0	0

Figure 4.6: counting of vehicles in images

Figure 4.6 shows the output of vehicle counting in images where it shows the image name, car count, motorbike/bike/bicycle count, bus count and truck count. It is output obtained from the detection of vehicles in images and classified them as bus, truck, car and bicycle. After classification, the number of vehicles in each classification is counted and store in the csv file.

The below figure 4.7 shows the counting of vehicles in videos.

Direction	car	motorbike	bus	truck
Up	20	0	2	0
Down	5	3	2	1

Figure 4.7: counting of vehicles in videos

Figure 4.7 shows the output of vehicle counting in videos where it shows direction of vehicle, car count, motorbike/bike/bicycle count, bus count and truck count. It is output obtained from the detection of vehicles in videos and classified them as bus, truck, car and bicycle. After classification, the number of vehicles in each classification is counted and store in the csv file. accuracy is calculated by calculating the mean of all accuracy predicted for all the vehicles in image and video.

Because of the similarities in pre-processing, we will skip over the repetitious subjects and show the algorithm's results in Table 4.2. In order to comprehend how CNN handles this type of input. And how accurate is the predicted result are.

Parameters	**Values**
Time	0.098 second
Accuracy	0.989
Error	0.07

Table 4.2:CNN classifier training results

Like shown above CNN obtained 98.9% accuracy in classification prediction of our dataset with 7% error and those values in evaluation matrices. The detected vehicles and their counts are stored separately for counting vehicles in images and videos.

4.3 Comparison and the Best Model among the Models

The two models were estimated on the same dataset using the same computer, despite the fact that ultimate of the methodologies were the same, in order to determine which algorithm was best **s**uited for this task. The content is sensitive and precise to estimate the findings because of the necessity of vehicle discovery and shadowing. The results of the two models are displayed as bar charts in a comparison manner in Figure4.8 in order to comprehend and compare them.

4.4 Evaluation metrics:

Utilizing the performance evaluation metrics provided by the scikit learn package during this performance evaluation. The major goal is to select the optimum model for our particular situation. The metrics includes accuracy score, error, precision, recall, f1score used as evaluation measures.

Accuracy

Let's look at the accuracy of the six categorization models we created. We may utilise the scikit-learn package's 'accuracy score' function to implement it in Python.

Error

One of the most important tasks in Machine Learning is to model data and predict outcomes using various Classification and Regression Algorithms. However, with so many algorithms to pick from, forecasting the final data is quite challenging. As a result, we must compare our models and select the most accurate.

We can figure out the scores of our ML Model using the sklearn package and then pick the method with the highest score to forecast our output. Another useful technique to improve our models is to calculate errors like mean absolute error and mean squared error and strive to reduce them. We use mean absolute error in this project.

Mean Absolute Error (MAE): It is the mean of all absolute error

Precision

perfection describes how multitudinous of the cases that were directly predicted turned out to be positive. Precision comes in handy when False Cons are further of a problem than False Negatives. The number of genuine cons divided by the number of anticipated cons is the perfection of a marker.

Recall

The term recall refers to how numerous of the factual positive cases our model duly prognosticated. It's a good statistic to use when False Negative is more dangerous than False Positive. The number of true cons divided by the total number of factual cons is called recall for a marker.

F1score

It presents a emulsion of Precision and Recall measures. When Precision equals Recall, it reaches its peak. The harmonious mean of delicacy and recall is the F1 Score.

Time

The number of times a statement is performed is referred to as its temporal complexity. It is also known as time complexity and it is referred as time.

	PRECISION	RECALL	F1	TIME	ERROR	ACCURACY
CNN	0.99%	0.99%	0.99%	0.89	0.9%	0.9892 %
SVM	0.98%	0.98%	0.99%	1.009	0.7%	0.9857 %

Table 4.4: Evaluation metrics result

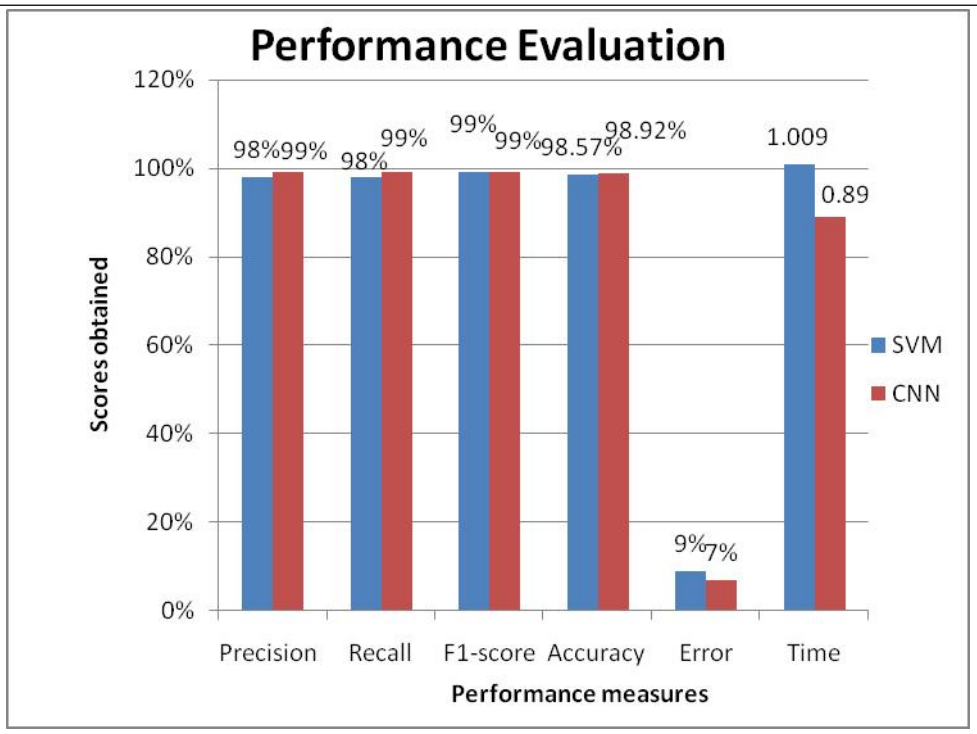


Figure 4.8: Performance Evaluation

Based on the above figure, it is known that the F1-score is same for SVM and CNN and obtained 99%. The precision for SVM is 98% where CNN obtained 99%. The Recall for SVM is 98% and for CNN is 99%. Finally, the accuracy for SVM is 98.57% but CNN obtained 98.92%. from the figure, it is concluded that CNN performs better than SVM for detecting the vehicle in real time video and images.

CONCLUSION AND RECOMMENDATIONS

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

Vehicle detection and tracking are part of object detection, which is utilized in traffic, cities, and other areas, and the issue is becoming increasingly important. That goal is to use existing image pre-processing techniques and tools to increase the performance of these algorithms and models. This research implemented two-vehicle detection andtracking classifier algorithms. Support Vector Machine (SVM) and CNN using YOLOv3 are the two models. These two techniques were the most often applied in the existing work. As a result, these models are selected and compared in order to determine which model was the better of the two. Many strategies have been used to improve accuracy and get the best possible result. The models were trained using the same dataset, and the results revealed that CNN outperformed SVM.

FUTURE ENHANCEMENT

CHAPTER 6

FUTURE ENHANCEMENT

Looking back at the study's limitations, there are activities and alternatives that may be added to the research or worked on individually. Vehicle detection is most important topic and it is improving day by day. Although many existing works are related to the vehicle detection and tracking, the implementation of such technique in real working environment is still challenging one. The machine learning and deep learning algorithms are modified day by day, so developing the enhanced deep learning model for vehicle detection and tracking can be recommended.

In future, other deep learning techniques can be implemented for vehicle detection and tracking with increased performance.

REFERENCES

REFERENCES

1) Andrew, W. M. and Victor, M. (2003), Handbook of International Banking (London: Edward Elgar Publishing Limited), 350-358

2) Bambrick, N. (2016). Support Vector Machines for dummies; A Simple Explanation. Retrieved April 28, 2018, from AYLIEN | Text Analysis API | Natural Language Processing: http://blog.aylien.com/support-vector-machines-for-dummies-a-simple/pp.1-13.

3) Basak, D., Pal, S., &Patranabis, D. C. (2007). Support vector regression. Neural Information Processing-Letters and Reviews, 11(10), 203-224.

4) Berni, J. A., Zarco-Tejada, P. J., Suárez, L., &Fereres, E. (2009). Thermal and narrow band multi spectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. IEEE Transactions on Geoscience and Remote Sensing, 47(3), 722-738.

5) Chen, X., & Meng, Q. (2015, November). Robust vehicle tracking and detection from UAVs.In 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR) (pp. 241-246).IEEE.

6) Chen.X,(2015), Automatic Vehicle Detection and Tracking in Aerial Video, Doctor of Philosophy Thesis in Loughborough University.

7) Kalghatgi, M. P., Ramannavar, M., & Dr. Sidnal, N. S. (2015). A Neural Network Approach to Personality Prediction based on the Big-Five Model. International Journal of Innovative Research in Advanced Engineering, 2(8), 56–63

8) Kanistras, K., Martins, G., Rutherford, M. J., & Valavanis, K. P. (2015). Survey of unmanned aerial vehicles (UAVs) for traffic monitoring. Handbook of unmanned aerial vehicles,2643-2666.

9) Khan, K., Baharudin, B. B., & Khan, A. (2009, June). Mining opinion from text documents: A survey. In Digital Ecosystems and Technologies,2009. DEST'09. 3rd IEEE International Conference on (pp. 217-222).IEEE.

10) Koza, J. R., Bennett, F. H., Andre, D., & Keane, M. A. (1996). Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In Artificial Intelligence in Design'96 (pp. 151-170). Springer, Dordrecht.

11) Noh.S,Shim.DandJeon.M,(2015),'Adaptive Sliding - Window Strategy for Vehicle Detection in Highway Environments', IEEE transactions on Intell. Transport. Syst., Ofor, e.(2018). Machine learning techniques for immune therapy dataset classification (Master dissertation, Near East University).

12) Patel, P. J., Patel, N. J., & Patel, A. R. (2014). Factors affecting currency exchange rate, economical formulas and prediction models. International Journal of Application or Innovation in Engineering & Management(IJAIEM), 3(3), 53-56.

13) Pujari, M. V.,Sayyed, A. H., Shahani, H., Rupani, D., & Student, B. E. (2018). Forex Trading System.International Journal of Engineering Science, 17116.

14) Razakarivony, S., & Jurie, F. (2016). Vehicle detection in aerial imagery: A small target detection benchmark. Journal of Visual Communication and Image Representation,34, 187-203.

15) Russell, S. J., &Norvig, P. (2016). Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited.

16) Sahli, S., Ouyang, Y., Sheng, Y., & Lavigne, D. A. (2010, April). Robust vehicle detection inlow -resolution aerial imagery. In Airborne Intelligence, Surveillance, Reconnaissance(ISR) Systems and Applications VII (Vol. 7668, p. 76680G). International Society for Optics and Photonics.

17) Sayad,S.(2017).Support Vector Machine.Support Vector Machines.Np,ndWeb,26.

18) Schumaker, R. P., & Chen, H. (2009). Textual analysis of stock market prediction using breaking financial news: The AZFin text system. ACM Transactions on Information Systems(TOIS), 27(2), 12.

19) Susaki, J. (2015). Region-based automatic mapping of tsunami-damaged buildings using multi - temporalaerial images.Natural Hazards,76(1), 397-420.

20) Tenti, P. (1996). Forecasting foreign exchange rates using recurrent neural networks. Applied Artificial Intelligence, 10(6), 567-582.

21)The Federal Reserve Board. (2004) "FRB:Speech, Bernanke -- International Monetary Reform and Capital Freedom--October14, 2004".

22) Tsai, Y. C., Chen, J. H., & Wang, J. J. (2018). Predict Forex Trend via Convolutional Neural Networks. arXiv preprintar Xiv:1801.03018.

23) Tseng, F. M., Tzeng, G. H., Yu, H. C., & Yuan, B. J. (2001). Fuzzy ARIMA model for forecasting the foreign exchange market.Fuzzy set sand systems,118(1),9-19.

24)UCBWiki.(2016,February).Rein forcement Learning with Function approximation.RetrievedApril22,2018,fromUCBWIki:http://wiki.ubc.ca/images/b/bd/Rldiagram1.png

25) Van Gerven, M., & Bohte, S. (Eds.). (2018). Artificial neural networks as models of neural in formation processing. Frontiers Media SA.

26) Viola, P., Jones, M. J., & Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. International Journalof Computer Vision,63(2), 153-161.

27) Vyklyuk, Y., Vukovic, D., & Jovanovic, A. (2013). FOREX prediction with neural network:USD/EUR currency pair.Актуальніпроблемиекономіки, (10),261-273.

28) Wang, X., Zhu, H., Zhang, D., Zhou, D., & Wang, X. (2014). Vision-based detection and tracking of a mobile ground target using a fixed-wing UAV. International Journal of Advanced Robotic Systems, 11(9), 156.

29) Wei, P., Lu, X., Tang, T., Li, C., & Song, J. (2015, August). A highway vehicle detectionmethodbasedontheimprovedvisualbackgroundextractor.In 201512thInternational Conference on Fuzzy Systems and Knowledge Discovery (FSKD) (pp.1519-1524).IEEE.

ANNEXURE

ANNEXURE:

CNN CODING:

```
# Import necessary packages

import cv2

import csv

import collections

import numpy as np

from tracker import *

# Initialize Tracker

tracker = EuclideanDistTracker()

# Initialize the videocapture object

cap = cv2.VideoCapture('v1.avi')

input_size = 320

# Detection confidence threshold

confThreshold =0.2

nmsThreshold= 0.2

font_color = (0, 0, 255)

font_size = 0.5

font_thickness = 2

# Middle cross line position

middle_line_position = 225

up_line_position = middle_line_position - 15
```

```
down_line_position = middle_line_position + 15

# Store Coco Names in a list

classesFile = "coco.names"

classNames = open(classesFile).read().strip().split('\n')

print(classNames)

print(len(classNames))

# class index for our required detection classes

required_class_index = [2, 3, 5, 7]

detected_classNames = []

## Model Files

modelConfiguration = 'yolov3-320.cfg'

modelWeigheights = 'yolov3-320.weights'

# configure the network model

net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeigheights)

#net = cv2.dnn.readNet(modelWeigheights, modelConfiguration)

# Configure the network backend

net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)

net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

# Define random colour for each class

np.random.seed(42)

colors = np.random.randint(0, 255, size=(len(classNames), 3), dtype='uint8')

# Function for finding the center of a rectangle
```

```
def find_center(x, y, w, h):
    x1=int(w/2)
    y1=int(h/2)
    cx = x+x1
    cy=y+y1
    return cx, cy
    # List for store vehicle count information
temp_up_list = []
temp_down_list = []
up_list = [0, 0, 0, 0]
down_list = [0, 0, 0, 0]
# Function for count vehicle
def count_vehicle(box_id, img):
  x, y, w, h, id, index = box_id
    # Find the center of the rectangle for detection
    center = find_center(x, y, w, h)
    ix, iy = center
    # Find the current position of the vehicle
    if (iy > up_line_position) and (iy < middle_line_position):
        if id not in temp_up_list:
            temp_up_list.append(id)
```

```
    elif iy < down_line_position and iy > middle_line_position:
        if id not in temp_down_list:
            temp_down_list.append(id)
    elif iy < up_line_position:
        if id in temp_down_list:
            temp_down_list.remove(id)
            up_list[index] = up_list[index]+1
    elif iy > down_line_position:
        if id in temp_up_list:
            temp_up_list.remove(id)
            down_list[index] = down_list[index] + 1
    # Draw circle in the middle of the rectangle
    cv2.circle(img, center, 2, (0, 0, 255), -1) # end here
    print(up_list)
    print(down_list)
    print(temp_down_list)
    print(temp_up_list)
```

 # Function for finding the detected objects from the network output

```
def postProcess(outputs,img):
    global detected_classNames
    height, width = img.shape[:2]
    boxes = []
```

```
    classIds = []

    confidence_scores = []

    detection = []

    for output in outputs:

        for det in output:

            scores = det[5:]

            classId = np.argmax(scores)

            confidence = scores[classId]

            if classId in required_class_index:

                if confidence > confThreshold:

                    # print(classId)

                    w,h = int(det[2]*width) , int(det[3]*height)

                    x,y = int((det[0]*width)-w/2) , int((det[1]*height)-h/2)

                    boxes.append([x,y,w,h])

                    classIds.append(classId)

                    confidence_scores.append(float(confidence))

    # Apply Non-Max Suppression

indices=cv2.dnn.NMSBoxes(boxes,confidence_scores,confThreshold, nmsThreshold)

    # print(classIds)

    for i in indices.flatten():

        x, y, w, h = boxes[i][0], boxes[i][1], boxes[i][2], boxes[i][3]
```

```
        # print(x,y,w,h)
        color = [int(c) for c in colors[classIds[i]]]
        name = classNames[classIds[i]]
        detected_classNames.append(name)
        # Draw classname and confidence score
        cv2.putText(img,f'{name.upper()} {int(confidence_scores[i]*100)}%',
                    (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 1)
        # Draw bounding rectangle
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 1)
        detection.append([x, y, w, h, required_class_index.index(classIds[i])])
    print(detection)
    # Update the tracker for each object
    boxes_ids = tracker.update(detection)
    for box_id in boxes_ids:
        count_vehicle(box_id, img)
def realTime():
    while True:
        success, img = cap.read()
        img = cv2.resize(img,(0,0),None,0.5,0.5)
        ih, iw, channels = img.shape
        blob = cv2.dnn.blobFromImage(img, 1 / 255, (input_size, input_size), [0, 0, 0], 1,
crop=False)
```

```
        # Set the input of the network
        net.setInput(blob)
        layersNames = net.getLayerNames()
        outputNames = [layersNames[i - 1] for i in net.get Unconnected OutLayers()]
        # Feed data to the network
        outputs = net.forward(outputNames)
        # Find the objects from the network output
        postProcess(outputs,img)
        # Draw the crossing lines
        cv2.line(img, (0, middle_line_position), (iw, middle_line_position), (255, 0, 255), 2)
        cv2.line(img, (0, up_line_position), (iw, up_line_position), (0, 0, 255), 2)
        cv2.line(img, (0, down_line_position), (iw, down_line_position), (0, 0, 255), 2)
        # Draw counting texts in the frame
        cv2.putText(img, "Up", (110, 20), cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Down",(160, 20),cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Car:"+str(up_list[0])+""+ str(down_list[0]), (20, 40), cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Motorbike:  "+str(up_list[1])+""+ str(down_list[1]), (20,60), cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
        cv2.putText(img, "Bus:"+str(up_list[2])+""+ str(down_list[2]), (20, 80), cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)
```

```
        cv2.putText(img,  "Truck:"+str(up_list[3])+""+ str(down_list[3]), (20,100),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)

        # Show the frames

        cv2.imshow('Output', img)

        if cv2.waitKey(1) == ord('q'):

            break

    # Write the vehicle counting information in a file and save it

    with open("data.csv", 'w') as f1:

        cwriter = csv.writer(f1)

        cwriter.writerow(['Direction', 'car', 'motorbike', 'bus', 'truck'])

        up_list.insert(0, "Up")

        down_list.insert(0, "Down")

        cwriter.writerow(up_list)

        cwriter.writerow(down_list)

    f1.close()

    # print("Data saved at 'data.csv'")

    # Finally realese the capture object and destroy all active windows

    cap.release()

    cv2.destroy All Windows()


if__name__== '__main__':
```

```
realTime()
image_file = 'test1.jpg'
def from_static_image(image):
    img = cv2.imread(image)
    blob = cv2.dnn.blobFromImage(img, 1 / 255, (input_size, input_size), [0, 0, 0], 1, crop=False)
    # Set the input of the network
    net.setInput(blob)
    layersNames = net.getLayerNames()
    outputNames = [layersNames[i - 1] for i in net.getUnconnectedOutLayers()]
    # Feed data to the network
    outputs = net.forward(outputNames)
    # Find the objects from the network output
    postProcess (outputs,img)
    # count the frequency of detected classes
    frequency = collections.Counter(detected_classNames)
    print("count frequency")
    print(frequency)
    import pandas as pd
    data=pd.read_csv("static-data.csv")
    previous=data.tail(1)
    print(previous)
```

```
        car_total=frequency['car']-(previous["1"].to_numpy())

        motorbike_total=frequency['motorbike']-(previous["0"].to_numpy())

        bus_total=frequency['bus']-(previous["0.1"].to_numpy())

        truck_total=frequency['truck']-(previous["0.2"].to_numpy())

        print(car_total)

        #    lastrow=data[-1]#    print(lastrow)

        # Draw counting texts in the frame

  cv2.putText(img,"Car:"+str(car_total),(20,40), cv2.FONT_HERSHEY_SIMPLEX,
font_size, font_color, font_thickness)

    cv2.putText(img,"Motorbike:"+str(motorbike_total),(20,60),
    cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)

  cv2.putText(img,"Bus:"+str(bus_total),(20,80),cv2.FONT_HERSHEY_SIMPLEX,
font_size, font_color, font_thickness)

  cv2.putText(img,"Truck:"+str(truck_total),(20,100),
cv2.FONT_HERSHEY_SIMPLEX, font_size, font_color, font_thickness)

        cv2.imshow("image", img)

        cv2.waitKey(0)

        # save the data to a csv file

        with open("static-data.csv", 'a') as f1:

            cwriter = csv.writer(f1)

  cwriter.writerow([image,frequency['car'],frequency['motorbike'],frequency['bus'],
frequency['truck']])

        f1.close()
```

```
if__name__== '__main__':

    # realTime()

    from_static_image(image_file)
```

SVM CODING:

```
import cv2

import glob

import numpy as np

import time

import matplotlib.image as mpimg

import matplotlib.pyplot as plt

%matplotlib inline

from scipy.ndimage.measurements import label

from skimage.feature import hog

from sklearn.svm import LinearSVC

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import classification_report
```

#from moviepy.editor import VideoFileClip

 # Set a random seed for comparing runs

```
np.random.seed(seed=55)

# Get vehicles and non-vehicles filenames as a list

def training_images_list(glob_file_path):

    images_list = glob.glob(glob_file_path)

    return images_list

all_cars=training_images_list('Vehicle-Detection-and-Tracking
master/vehicles/*.png')

all_notcars=training_images_list('Vehicle-Detection-and-Tracking-master/non-
vehicles/*.png')

#print(all_cars)

# Select Train+Validation sample size and shares. Test set will take the
remaining images

sample_size = 7500

valid_share = 0.2


# Randomly select the cars and non-cars indexes

idx_cars =  np.random.choice(len(all_cars),    sample_size,replace=False)

idx_notcars = np.random.choice(len(all_notcars), sample_size,replace=False)

# Select test sample indexes as a complement of the training/validation

other_idx_cars=np.setdiff1d(np.arange(len(all_cars)),idx_cars,assume_unique=Tru
e)

other_idx_notcars=np.setdiff1d(np.arange(len(all_notcars)),idx_notcars,assume_un
ique=True)
```

 # Select filenames for car and non cars for training/validation

```
cars    =  list(np.array(all_cars)[idx_cars])

notcars = list(np.array(all_notcars)[idx_notcars])

# Select filenames for car and non cars for testing

cars_test    = list(np.array(all_cars)[other_idx_cars])

notcars_test = list(np.array(all_notcars)[other_idx_notcars])


print("Total vehicles:", len(all_cars))

print("- of which:")

print(" vehicles for training:",len(cars))

print("  - of which:")

print("    validation (share) :",valid_share)

print("  vehicles for testing     :",len(cars_test))

print(" ")

print("Total non-vehicles:",len(all_notcars))

print("- of which:")

print(" non-vehicles for training:",len(notcars))

print("  - of which:")

print("    validation (share):",valid_share)

print("  non-vehicles for testing :",len(notcars_test))
```

 # Code from `lesson_functions.py`

 # (see the lesson: Vehicle Detection and Tracking, track 32: "Search and Classify").

```
defget_hog_features(img,orient,pix_per_cell,cell_per_block,vis=False,feature_vec=True):
    ''' returns HOG features and/or visualization '''
    # Call with two outputs if vis==True
    if vis == True:
        hog_features, hog_image = hog(img, orientations=orient,
                                      pixels_per_cell=(pix_per_cell, pix_per_cell),
                                      cells_per_block=(cell_per_block, cell_per_block),
                                      transform_sqrt=True,
                                      visualize=vis, feature_vector=feature_vec)
        return hog_features, hog_image
    # Otherwise call with one output
    else:
        hog_features = hog(img, orientations=orient,
                           pixels_per_cell=(pix_per_cell, pix_per_cell),
                           cells_per_block=(cell_per_block, cell_per_block),
                           transform_sqrt=True,
                           visualize=vis, feature_vector=feature_vec)
        return hog_features
```

```
def bin_spatial(img, size=(32, 32)):
    ''' Computes binned color features '''
    color1 = cv2.resize(img[:,:,0], size).ravel()
    color2 = cv2.resize(img[:,:,1], size).ravel()
    color3 = cv2.resize(img[:,:,2], size).ravel()
    return np.hstack((color1, color2, color3))

def color_hist(img, nbins=32, bins_range=(0, 256)):
    ''' Computes color histogram features'''
    # NEED TO CHANGE bins_range if reading .png files with mpimg!

    # Compute the histogram of the color channels separately
    channel1_hist = np.histogram(img[:,:,0], bins=nbins, range=bins_range)
    channel2_hist = np.histogram(img[:,:,1], bins=nbins, range=bins_range)
    channel3_hist = np.histogram(img[:,:,2], bins=nbins, range=bins_range)
    # Concatenate the histograms into a single feature vector
hist_features=np.concatenate((channel1_hist[0],channel2_hist[0],channel3_hist[0]))

    # Return the individual histograms, bin_centers and feature vector
    return hist_features
```

```
def convert_RGB_color(image, color_space='RGB'):
    if color_space != 'RGB':
        if color_space == 'HSV':
            feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2HSV)
        elif color_space == 'LUV':
            feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2LUV)
        elif color_space == 'HLS':
            feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2HLS)
        elif color_space == 'YUV':
            feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)
        elif color_space == 'YCrCb':
            feature_image = cv2.cvtColor(image, cv2.COLOR_RGB2YCrCb)
    else: feature_image = np.copy(image)
    return feature_image

def extract_features(imgs, color_space='RGB', spatial_size=(32, 32),
                        hist_bins=32, orient=9,
                        pix_per_cell=8, cell_per_block=2, hog_channel=0,
                        spatial_feat=True, hist_feat=True, hog_feat=True):
    ''' Define a function to extract features from a list of images
```

```
This function call get_hog_features(), bin_spatial() and color_hist()

'''

# Create a list to append feature vectors to

vector_features = []

# Iterate through the list of images

for file in imgs:

    file_features = []

    # Read in each one by one

    image = mpimg.imread(file)


    # Apply color conversion if other than 'RGB'

    feature_image = convert_RGB_color(image, color_space = color_space)

    if hist_feat == True:

        # Apply color_hist()

        hist_features = color_hist(feature_image, nbins=hist_bins)

        file_features.append(hist_features)


    if hog_feat == True:

    # Call get_hog_features() with vis=False, feature_vec=True

        if hog_channel == 'ALL':

            hog_features = []

            for channel in range(feature_image.shape[2]):
```

```
                hog_features.append(get_hog_features(feature_image[:,:,channel],
                                    orient, pix_per_cell, cell_per_block,
                                    vis=False, feature_vec=True))
            hog_features = np.ravel(hog_features)
        else:
            hog_features = get_hog_features(feature_image[:,:,hog_channel], orient,
                    pix_per_cell, cell_per_block, vis=False, feature_vec=True)
        # Append the new feature vector to the features list
        file_features.append(hog_features)

    if spatial_feat == True:
        # Apply bin_spatial()
        spatial_features = bin_spatial(feature_image, size=spatial_size)
        file_features.append(spatial_features)

    vector_features.append(np.concatenate(file_features))

# Return list of feature vectors
return vector_features
### Set parameters (image 64x64 pixels)

color_space = 'YCrCb'     # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
```

```
orient         = 9         # HOG orientations
pix_per_cell   = 8         # HOG pixels per cell **2 (was 8)
cell_per_block = 2         # HOG cells per block **2
hog_channel    = 'ALL'         # Can be 0, 1, 2, or "ALL"


spatial_size = (16, 16) #(32, 32) # Spatial binning dimensions
hist_bins      = 16        # Number of histogram bins (default 32)
# Select features
spatial_feat   = True      # Spatial features on or off
hist_feat      = False # Histogram features on or off
hog_feat       = True      # HOG features on or off
# Set the number of randomly select features to be used
feature_size = 3030 #2800
def reduce_features(some_features,idx):
    ''' Reduce the overall number of feature used (like dropout)
    Pick only some of the features as specified by the (idx) column index'''


    some_features=np.array(some_features)
    some_features=some_features.transpose() [idx]
    some_features=list(some_features.transpose())
    return some_features
# Extract features from cars and non cars images
```

```
t0=time.time()


car_features = extract_features(cars, color_space=color_space,
                        spatial_size=spatial_size, hist_bins=hist_bins,
                        orient=orient, pix_per_cell=pix_per_cell,
                        cell_per_block=cell_per_block,
                        hog_channel=hog_channel, spatial_feat=spatial_feat,
                        hist_feat=hist_feat, hog_feat=hog_feat)


print(len(car_features[0]),'Car features ... extracted')


idx = np.random.choice(len(car_features[0]), feature_size)


car_features = reduce_features(car_features,idx)
print(len(car_features[0]),'Car features ... used')


notcar_features = extract_features(notcars, color_space=color_space,
                        spatial_size=spatial_size, hist_bins=hist_bins,
                        orient=orient, pix_per_cell=pix_per_cell,
                        cell_per_block=cell_per_block,
                        hog_channel=hog_channel, spatial_feat=spatial_feat,
                        hist_feat=hist_feat, hog_feat=hog_feat)
```

```
    print(len(notcar_features[0]),'Non car features ... extracted')

    notcar_features = reduce_features(notcar_features,idx)

    print(len(notcar_features[0]),'Non car features ... used')



    X = np.vstack((car_features, notcar_features)).astype(np.float64)

    t1 = time.time()



    print('\nHOG parameters')

     print('Using:',orient,'orientations',pix_per_cell,'pixels per cell and',
     cell_per_block,'cells per block')

    print(round(t1-t0, 2), 'seconds to extract',len(car_features[0]),'features ...')

    #%whos ndarray

    # Example: feature extraction on a sample image

    import matplotlib.pyplot as plt

    import matplotlib.image as mpimg


 sample_image=mpimg.imread('Vehicle-Detection-and-Tracking
   master/vehicles/78.png')

    plt.imshow(sample_image)

    plt.title("Sample image")

    plt.show()

    mpimg.imsave('Vehicle-Detection-and-Tracking-
master/output_images/KITTI_sample78.png',sample_image)
```

```
    # Hog-channel 0
    sample_feature_vec, hog_sample = get_hog_features(sample_image[:,:,0], orient, pix_per_cell,  cell_per_block, vis=True, feature_vec=True)
    plt.imshow(hog_sample, cmap='gray')
    plt.title("Sample HOG Image (channel 0)")
    mpimg.imsave('Vehicle-Detection-and-Tracking-master/output_images/KITTI_hog_sample78_0.png',hog_sample, cmap='gray')
    plt.show()
    # Hog-channel 1
    sample_feature_vec, hog_sample = get_hog_features(sample_image[:,:,1], orient, pix_per_cell, cell_per_block, vis=True, feature_vec=True)
    plt.imshow(hog_sample, cmap='gray')
    plt.title("Sample HOG Image (channel 1)")
    mpimg.imsave('Vehicle-Detection-and-Tracking-master/output_images/KITTI_hog_sample78_1.png',hog_sample, cmap='gray')
    plt.show()
    # Hog-channel 2
    sample_feature_vec, hog_sample = get_hog_features(sample_image[:,:,2], orient, pix_per_cell, cell_per_block, vis=True, feature_vec=True)
    plt.imshow(hog_sample, cmap='gray')
    plt.title("Sample HOG Image (channel 2)")
    mpimg.imsave('Vehicle-Detection-and-Tracking-master/output_images/KITTI_hog_sample78_2.png',hog_sample, cmap='gray')
```

```
    plt.show()

    # Fit a per-column scaler

    X_scaler = StandardScaler().fit(X)


    # Apply the scaler to X

    scaled_X = X_scaler.transform(X);


    # Define the labels vector

    y = np.hstack((np.ones(len(car_features)), np.zeros(len(notcar_features))))


    # Split up data into temp randomized training and validation sets

    rand_state = np.random.randint(0, 100)

    X_train, X_valid,y_train,y_valid=train_test_split(scaled_X, y,test_size=valid_share,
random_state=np.random.randint(0, 100))

    print ("Selected data set dimensions:", scaled_X.shape,y.shape)

    print ("Training set:  ", X_train.shape,y_train.shape)

    print ("Validation set:  ", X_valid.shape,y_valid.shape)


    del scaled_X, y

    # Select classifier

    svc = LinearSVC()

    # Set the parameters by cross-validation
```

```
tuned_parameters = [{'C': [0.05,0.1,0.25,0.5,0.75,1.]}]


# Code more than inspired by the GridSearchCV help\
scores = ['precision']
for score in scores:
    print("# Tuning hyper-parameters for %s" % score)
    print()


    clf=GridSearchCV(svc,tuned_parameters,cv=5, scoring='%s_macro' % score)
    clf.fit(X_train, y_train)
    print("Best parameters set found on development set:")
    print()
    print(clf.best_params_)
    print()
    print("Grid scores on development set:")
    print()
    means = clf.cv_results_['mean_test_score']
    stds = clf.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, clf.cv_results_['params']):
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params))
    print()
```

```
print("Detailed classification report:")

print()

print("The model is trained on the full development set.")

print("The scores are computed on the full evaluation set.")

print()

y_pred = clf.predict(X_valid)

print(classification_report(y_valid, y_pred))

print()


# Other monitoring indicators

print('\nTemp training data set:',X_train.shape,y_train.shape)

print('     Train accuracy: ', round(clf.score(X_train, y_train), 4))

print('     Valid accuracy:  ', round(clf.score(X_valid, y_valid), 4))


if (len(X_train[y_valid!=y_pred])>0):

    dec_fun = clf.decision_function(X_train[y_valid!=y_pred])


print("\nDecisionfunction:misclassifiedmin/max:",round(min(dec_fun),2),round(max(
dec_fun),2))

    plt.plot(np.sort(dec_fun))

    plt.show()
```

```
del  X_valid,y_valid,y_pred
# Testing
cars_test_features = extract_features(cars_test, color_space=color_space,
                    spatial_size=spatial_size, hist_bins=hist_bins,
                    orient=orient, pix_per_cell=pix_per_cell,
                    cell_per_block=cell_per_block,
                    hog_channel=hog_channel, spatial_feat=spatial_feat,
                    hist_feat=hist_feat, hog_feat=hog_feat)
print(len(cars_test_features[0]),'cars_test_features ... original')
cars_test_features = reduce_features(cars_test_features,idx)
print(len(cars_test_features[0]),'cars_test_features ... actual')

notcars_test_features = extract_features(notcars_test, color_space=color_space,
                    spatial_size=spatial_size, hist_bins=hist_bins,
                    orient=orient, pix_per_cell=pix_per_cell,
                    cell_per_block=cell_per_block,
                    hog_channel=hog_channel, spatial_feat=spatial_feat,
                    hist_feat=hist_feat, hog_feat=hog_feat)
print(len(notcars_test_features[0]),'notcars_test_features ... original')
notcars_test_features = reduce_features(notcars_test_features,idx)
print(len(notcars_test_features[0]),'notcars_test_features ... actual')
```

```
    X_test = np.vstack((cars_test_features, notcars_test_features)).astype(np.float64)
    y_test=np.hstack((np.ones(len(cars_test_features)),np.zeros(len(notcars_test_features))))

    # Apply the scaler to X
    scaled_X_test = X_scaler.transform(X_test);

    y_pred = clf.predict(scaled_X_test)

    print(classification_report(y_test, y_pred))

    print()


    # Still monitoring
    print('\nTemp training data set:',X_train.shape,y_train.shape)

    print('     Train accuracy: ', round(clf.score(X_train, y_train), 4))

    print('     Test accuracy:  ', round(clf.score(scaled_X_test, y_test), 4))
    # Single function (from lesons) that can extract features using hog sub-sampling and make predictions

    deffind_cars(img,ystart,ystop,scale,svc,X_scaler,orient,pix_per_cell,cell_per_block, spatial_size, hist_bins,


color_space,hog_channel,cells_per_step,idx,spatial_feat=True,hist_feat=False,clip = False):


        bbox_list=[]
        out_img = np.copy(img)
```

```
    if clip:
        img = img.astype(np.float32)/255
    img_tosearch = img[ystart:ystop,:,:]
    ctrans_tosearch = convert_RGB_color(img_tosearch, color_space=color_space)

    if scale != 1:
        imshape = ctrans_tosearch.shape
        ctrans_tosearch=cv2.resize(ctrans_tosearch,(np.int(imshape[1]/scale),
np.int(imshape[0]/scale)))
    if  (hog_channel=='ALL'):
        ch1 = ctrans_tosearch[:,:,0]
        ch2 = ctrans_tosearch[:,:,1]
        ch3 = ctrans_tosearch[:,:,2]
    else:
        ch1 = ctrans_tosearch[:,:,hog_channel]
    # Define blocks and steps as above
    nxblocks = (ch1.shape[1] // pix_per_cell)-1
    nyblocks = (ch1.shape[0] // pix_per_cell)-1
    nfeat_per_block = orient*cell_per_block**2
    # 64 was the orginal sampling rate, with 8 cells and 8 pix per cell
    window = 64
    nblocks_per_window = (window // pix_per_cell)-1
```

```
    #cells_per_step = 2  # Instead of overlap, define how many cells to step (2)
    nxsteps = (nxblocks - nblocks_per_window) // cells_per_step
    nysteps = (nyblocks - nblocks_per_window) // cells_per_step
    # Compute individual channel HOG features for the entire image

hog1=get_hog_features(ch1,orient,pix_per_cell,cell_per_block,feature_vec=False)
    if (hog_channel=='ALL'):

hog2=get_hog_features(ch2,orient,pix_per_cell,cell_per_block,feature_vec=False)

hog3=get_hog_features(ch3,orient,pix_per_cell,cell_per_block,feature_vec=False)
    #count = 0
    for xb in range(nxsteps):
        for yb in range(nysteps):
            #count +=1
            ypos = yb*cells_per_step
            xpos = xb*cells_per_step
            # Extract HOG for this patch

hog_feat1=hog1[ypos:ypos+nblocks_per_window,xpos:xpos+nblocks_per_window].r
avel()
            if (hog_channel=='ALL'):
```

```
hog_feat2=hog2[ypos:ypos+nblocks_per_window,xpos:xpos+nblocks_per_window].r
avel()

hog_feat3=hog3[ypos:ypos+nblocks_per_window,xpos:xpos+nblocks_per_window].r
avel()
            hog_features = np.hstack((hog_feat1, hog_feat2, hog_feat3))
        else:
            hog_features = np.hstack((hog_feat1)
        xleft = xpos*pix_per_cell
        ytop = ypos*pix_per_cell

        # Extract the image patch
        subimg=cv2.resize(ctrans_tosearch[ytop:ytop+window, xleft:xleft+window],
(64,64))
        # Get color features
        if spatial_feat:
            spatial_features = bin_spatial(subimg, size=spatial_size)
        if hist_feat:
            hist_features = color_hist(subimg, nbins=hist_bins)
        # Scale features and make a prediction
        if hist_feat:
            test_features = X_scaler.transform(np.hstack((
```

```
                    hist_features,hog_features,spatial_features)).reshape(1, -1))
            else:
                temp = np.hstack((hog_features,spatial_features)).reshape(1, -1)
                temp = reduce_features(temp,idx)
                test_features = X_scaler.transform(temp)
            test_prediction = svc.predict(test_features)
            if test_prediction == 1:
                xbox_left = np.int(xleft*scale)
                ytop_draw = np.int(ytop*scale)
                win_draw = np.int(window*scale)
                cv2.rectangle(out_img,(xbox_left,
ytop_draw+ystart),(xbox_left+win_draw,ytop_draw+win_draw+ystart),(0,0,1.),6)
                bbox_list.append([(xbox_left,
ytop_draw+ystart),(xbox_left+win_draw,ytop_draw+win_draw+ystart)])


    #print("Number of windows examined:",count)
    return out_img,bbox_list
```

 # Function from the lessons that can extract features using hog sub-sampling and make predictions

```
def add_heat(heatmap, bbox_list):
    # Iterate through list of bboxes
    for box in bbox_list:
        # Add += 1 for all pixels inside each bbox
```

```
        # Assuming each "box" takes the form ((x1, y1), (x2, y2))
        heatmap[box[0][1]:box[1][1], box[0][0]:box[1][0]] += 1
    # Return updated heatmap
    return heatmap
def apply_threshold(heatmap, threshold):
    # Zero out pixels below the threshold
    heatmap[heatmap <= threshold] = 0
    # Return thresholded map
    return heatmap
def draw_labeled_bboxes(img, labels, clip = False, thickness=4):
    # Iterate through all detected cars
    for car_number in range(1, labels[1]+1):
        # Find pixels with each car_number label value
        nonzero = (labels[0] == car_number).nonzero()
        # Identify x and y values of those pixels
        nonzeroy = np.array(nonzero[0])
        nonzerox = np.array(nonzero[1])
        # Define a bounding box based on min/max x and y

bbox=((np.min(nonzerox),np.min(nonzeroy)),(np.max(nonzerox),np.max(nonzeroy)))
        # Draw the box on the image
        if clip:
```

```
            cv2.rectangle(img, bbox[0], bbox[1], (0,0,255), thickness)
        else:
            cv2.rectangle(img, bbox[0], bbox[1], (0.,0.,1.), thickness)
    # Return the image
    return img
def draw_boxes(img, bboxes, color=(0., 0., 1.), thick=6):
    ''' Define a function to draw bounding boxes'''
    # Make a copy of the image
    imcopy = np.copy(img)
    # Iterate through the bounding boxes
    for bbox in bboxes:
        # Draw a rectangle given bbox coordinates
        cv2.rectangle(imcopy, bbox[0], bbox[1], color, thick)
    # Return the image copy with boxes drawn
    return imcopy
def clear_undersized_rectancle(d1,d2):
    ''' Clear rectangles less than min_area wide'''
    min_area = 64*64
    if (d1*d2)<min_area:
        return True # rectangle will disappear
    else:
        return False #there is a "big" rectangle
```

```
    def exist_rectangle(boxes,sized_rect):
        ''' Check the existence of overlapping frames and select one'''
        ax = sized_rect[0][0]
        ay = sized_rect[0][1]
        bx = sized_rect[0][0]
        by = sized_rect[1][1]
        cx = sized_rect[1][0]
        cy = sized_rect[1][1]
        dx = sized_rect[1][0]
        dy = sized_rect[0][1]
        for i in range(len(boxes)):
            box = boxes[i]
            if((ax >= box[0][0]) & (ax <= box[1][0]) & (ay >= box[0][1]) & (ay <= box[1][1]) | \
               (bx >= box[0][0]) & (bx <= box[1][0]) & (by >= box[0][1]) & (by <= box[1][1]) | \
               (cx >= box[0][0]) & (cx <= box[1][0]) & (cy >= box[0][1]) & (cy <= box[1][1]) | \
               (dx >= box[0][0]) & (dx <= box[1][0]) & (dy >= box[0][1]) & (dy <= box[1][1]) ):
                return True
        return False
    def size_rectangle (labels, half_h, half_v):
```

```
    ''' Create a standrd rectangle around the midponint of
    labels resulting from a heat map
    if rectangle is at least min_size x min_size'''
    my_box = []
    for i in range(labels[1]):
        n_rect = i + 1

d1=np.max((labels[0]==n_rect).nonzero()[0])np.min((labels[0]==n_rect).nonzero()[0])
d2=np.max((labels[0]==n_rect).nonzero()[1])np.min((labels[0]==n_rect).nonzero()[1])
        if (clear_undersized_rectancle(d1,d2)):
            continue
h=np.int((np.min((labels[0]==n_rect).nonzero()[1])+np.max((labels[0]==n_rect).nonz
ero()[1]))/2)
v=np.int((np.min((labels[0]==n_rect).nonzero()[0])+np.max((labels[0]==n_rect).nonz
ero()[0]))/2)
        sized_rect = [(h-half_h,v-half_v),(h+half_h,v+half_v)]
        if (exist_rectangle(my_box,sized_rect)):
            pass
        else:
            my_box.append(sized_rect)
    return my_box
# Selecting the area of the image where cars can be
# and the dimensions of the searcing squared frames (here the array has 4 elements)
# Vertical (y axis) top limit where start search (for each of the 4 squared frames)
```

```
ystarts =   [400, 430,  440]     # top reference to start sliding


# How big are the frame with respect to the standard 64x64 pixel

scales  =    [.8,  1.2,  1.5]  # how much yo scale sub images

len_scales = len(scales)


# Set the number rows (from the top limit) where windows slide

rows    = np.multiply([2., 2.,   1.] ,64) # how many rows to slide from top

# Calculate the bottom line where stop searching

ystops  = np.add(ystarts,np.multiply(scales,rows)).astype(int) #bottom reference

# Set the overlapping

cells_per_step = 2

# Set the heatmap threshold to select frames that identifiy cars

heat_threshold = 2

image=mpimg.imread('Vehicle-Detection-and-Tracking
master/output_images/My_test_image.jpg')

# Uncomment the following line if you extracted training

# data from .png images (scaled 0 to 1 by mpimg) and the

# image you are searching is a .jpg (scaled 0 to 255)

image = image.astype(np.float32)/255.

draw_image = np.copy(image)

# Set the list containing box references
```

```
bbox_list=[]

t0=time.time()

# Repeat for each scale factor used in sliding

for i in np.arange(len_scales):


   scale = scales [i]

   ystart=  ystarts[i]

   ystop = ystops [i]

   _,bbox_list_temp = find_cars(draw_image, ystart, ystop, scale,

                        clf, X_scaler,

                        orient, pix_per_cell, cell_per_block,

                        spatial_size, hist_bins,

                        color_space, hog_channel,cells_per_step, idx,

                        spatial_feat=True,hist_feat=False)

   bbox_list.extend(bbox_list_temp)

t1 = time.time()

print(round(t1-t0, 2), 'seconds to find car(s)...')

# Show the "original" image

plt.imshow(image)

plt.title("Test image")

mpimg.imsave('Vehicle-Detection-and-Tracking-
master/output_images/real_image.jpg',image)
```

```
plt.show()

# Show the "first" frame detection

out_img = draw_boxes(draw_image,bbox_list)

plt.imshow(out_img)

plt.title('Original car-frames detection')

mpimg.imsave('Vehicle-Detection-and-Tracking-
master/output_images/Raw_detection.jpg',out_img)

plt.show()

# Create an image similar to one shown above

heat = np.zeros_like(out_img[:,:,0]).astype(np.float)

# Add heat to each box in box list

heat = add_heat(heat,bbox_list)

plt.imshow(heat, cmap='hot')

plt.title('Heat Map')

mpimg.imsave('Vehicle-Detection-and-Tracking-
master/output_images/Heat_map.jpg',heat,cmap='hot')

plt.show()

# Apply threshold to help remove false positives

heat = apply_threshold(heat,heat_threshold)

plt.imshow(heat, cmap='hot')

plt.title('Heat Map (after threshold) ')

mpimg.imsave('Vehicle-Detection-and-Tracking-
master/output_images/Heat_map_w_threshold.jpg',heat, cmap='hot')
```

```
plt.show()

# Visualize the heatmap when displaying

heatmap = np.clip(heat, 0, 255)

# Find initial boxes from heatmap using label function

labels = label(heatmap)

draw_img = draw_labeled_bboxes(np.copy(image), labels)

plt.imshow(draw_img)

plt.title('Car Positions from the Heatmap')

mpimg.imsave('Vehicle-Detection-and-Tracking-
master/output_images/Heat_car_positions.jpg',draw_img)

plt.show()

# Resize boxes with a "standard" box

new_labels = size_rectangle (labels,75,50)

# Draw final boxes from heatmap using label function

draw_img = draw_boxes(np.copy(image), new_labels)

plt.imshow(draw_img)

plt.title('Standard Frame Car Positions')

mpimg.imsave('Vehicle-Detection-and-Tracking-
master/output_images/Final_car_positions.jpg',draw_img)

plt.show()

def process_image(video_img):

    global heat_memory
```

```
    # Empty list of upper-left lower-rigth points of rectangles
    bbox_list_clip = []
    draw_image = np.copy(video_img)
    plt.imshow(draw_image)
    plt.title('Empty list of upper-left lower-rigth points of rectangles')
    plt.show()
    # Apply HOG and SVM to find cars with sliding windows
    for i in np.arange(len_scales):
        scale = scales [i]
        ystart= ystarts[i]
        ystop = ystops [i]

print(draw_image,ystart,ystop,scale,clf,X_scaler,orient,pix_per_cell,cell_per_block,
spatial_size, hist_bins,
              color_space, hog_channel, cells_per_step, idx)
          _,bbox_list_temp_clip = find_cars(draw_image, ystart, ystop, scale,
                                  clf, X_scaler,
                                  orient, pix_per_cell, cell_per_block,
                                  spatial_size, hist_bins,
                                  color_space, hog_channel,cells_per_step, idx,
                                  spatial_feat=True,hist_feat=False,clip = True)
        bbox_list_clip.extend(bbox_list_temp_clip)
```

```
print(bbox_list_clip)

# Create an image similar to the video image

heat_clip = np.zeros_like(video_img[:,:,0]).astype(np.float)

plt.imshow(heat_clip)

plt.title('Create an image similar to the video image')

plt.show()


# Add heat to each box in box list

heat_clip = add_heat(heat_clip,bbox_list_clip)

# Add car frames from previous images to the heat

for i in np.arange(len(heat_memory)):

    heat_clip = add_heat(heat_clip,heat_memory[i])


# Apply an increased threshold to remove outliers

heat_clip = apply_threshold(heat_clip,heat_threshold + 4)


# Visualize the heatmap when displaying

heatmap_clip = np.clip(heat_clip, 0, 255)


# Find boxes from heatmap using label function

labels_clip = label(heatmap_clip)
```

```
# Draw boxes around car with a "standard" rectangle
new_labels  = size_rectangle (labels_clip,80,40)

# Create result image to display in the video
result = draw_boxes(draw_image, new_labels, color=(0, 0, 255))

# Rearrange history/stack frames
for i in np.arange(len(heat_memory)-1):
    heat_memory[i] = np.copy(heat_memory[i+1])
# Update stack
heat_memory[len(heat_memory)-1] = np.copy(bbox_list_clip)

return result
from moviepy.editor import VideoFileClip
# Set a list to store previous frames
n_frames = 6
heat_memory = list([] for i in np.arange(n_frames))

# Apply car detection to video-clip
white_output = 'my_project5_video.mp4'
#clip1 = VideoFileClip("test_video.mp4")
```

```
    clip1=VideoFileClip("Vehicle-Detection-and-Tracking-
master/output_images/test_video.mp4")#.subclip(40,42)

    white_clip = clip1.fl_image(process_image)

    %time white_clip.write_videofile(white_output,audio=False)
```