DEEP LEARNING FOR OBJECT DETECTION IN AUTONOMOUS LAKE CLEANING

A Dual Degree Project Report

submitted by

DASARI ANANYANANDA

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY
&
MASTER OF TECHNOLOGY

INTERDISCIPLINARY DUAL DEGREE PROGRAM IN ROBOTICS

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2020

THESIS CERTIFICATE

This is to certify that the thesis titled **Deep Learning for Object Detection in Autonomous Lake Cleaning**, submitted by **Dasari Anantyananda (ME15B163)**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

(Somashekhar S Hiremath)
Research Guide
Dept. of Mechanical Engineering
Indian Institute of Technology Madras,
Chennai - 600 036

Place: Chennai

Date: 10th June 2020

ACKNOWLEDGEMENTS

I would like to thank **Prof. Somashekhar S Hiremath** for his vital support and guidance throughout this project. His guidance was invaluable in helping me accomplish this research work. I am grateful to him for providing me the opportunity to be a part of the world-class laboratory, **Precision Engineering and Instrumentation Laboratory (PEIL)** at IIT Madras. Thank you, Sir, for allowing me access to the high-tech facilities at the laboratory.

I would like to thank the review committee members, **Prof. Balaraman Ravindran**, **Prof. M Manivannan**, and **Prof. Krishna Vasudevan**, for their valuable suggestions during the mid-review, steering the work in the right direction. I would like to thank the members of PEIL, especially **Dr. Robins Mathew**, for providing assistance during the initial simulations. Also, I would like to thank my partner Manan, for his complete support and continuous motivation for the duration of this project. Without his crucial inputs and hard work, this project would not have reached this level.

I am also extremely grateful to the Department of Mechanical Engineering, IIT Madras, for providing me financial assistance for building the drone. I also thank the Department of Engineering Design, IIT Madras, for providing me valuable suggestions during my project and review. I am also thankful to the Central Electronics Centre and The Rotorcraft Lab at the Department of Aerospace Engineering, for their assistance during the building of the drone. I am also grateful to IIT Madras, for providing me lots of opportunities during my stay of five years here. Thank you for opening roads to industrial research and innovative study programs in cutting edge fields, promoting my self development and future aspirations.

Lastly, I thank my parents for being an unwavering source of support and encouraging me in all my endeavours, throughout my stay here.

Dasari Ananyananda

ABSTRACT

KEYWORDS: Lake, Robot, Quadcopter, Boat, Object detection, Neural Networks, ROS, Waypoint Navigation, Simulation, 3D Model

The problem of the pollution of our water bodies is very critical. The limited quantity of freshwater and the immense demand for it necessitates immediate measures to be taken to remedy the situation. More than 3 billion kilograms of garbage are dumped into the world's water bodies each year. The limited availability of freshwater, the growing population of the world, makes this problem more critical. As millions lose access to freshwater day by day, there is a clear need for sustainable solutions from cutting edge technology. The motivation to take up this huge and complex problem as my Dual Degree Project stems mainly from a desire to contribute towards society with the knowledge gained from our coursework. Also, the interdisciplinary demands of the solution inspired us to take up the work to improve our skills in areas untouched in class.

The project aims to tackle pollution with minimum human intervention to make it more cost-effective and efficient. Current methods of clearing litter in lakes are all dependent on human intervention and involvement throughout the process. The need for manpower is one of the main bottlenecks in the cleaning up process. Our solution involves the use of aerial robots to map the surface of the lake and identify the floating pieces of litter on the surface. To achieve this, we resorted to Deep Learning methods that have proven successful and efficient in the autonomous identification of objects. Our object detection module aims not only to identify and locate the litter but also to determine the material. This facility is of importance, especially when different types of materials require different methods of clearing up.

The detected litter locations are now sent to a coordinate transformation algorithm that uses drone sensor data to determine the absolute GPS coordinates of the litter. These coordinates are transferred to a way-point navigation algorithm that charts out an optimized distance path for collecting the litter. The collection is achieved by a marine robot, here a specially re-designed boat. The final boat simulations were performed, and a detailed analysis of the type of boat and the components required was conducted.

All the modules mentioned were developed on our own and fine-tuned for this project. The drone was assembled using standard commercially available components and using facilities at the PEIL lab, Rotorcraft lab, and Central Electronics Centre. Due to the complexity of the problem, this work and consequently, the thesis was divided between two Inter-Disciplinary Dual Degree Robotics students, Manan Maheshwari (ME15B118) and myself. Each module was developed indepedently and integrated later, to form the complete system.

Contents

List of Tables

Code Listings

List of Figures

ABBREVIATIONS

GPS	Global Positioning System
IMU	Inertial Measurement Unit
CV	Computer Vision
UAV	Unmanned Aerial Vehicle
AUV	Autonomous Underwater Vehicles
ROV	Remotely Operated Vehicles
DOF	Degrees of Freedom
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Network
GPU	Graphics Processing Unit
VGG	Visual Geometry Group
R-CNN	Region - Convolutional Neural Network
YOLO	You Only Look Once
SSD	Single Shot Detector
TSP	Travelling Salesman Problem
HSV	Hue, Saturation, Value
SURF	Speeded Up Robust Features
SSE	Sum of Squared Error
CE	Cross Entropy
NAG	Nesterov Accelerated Gradient
AdaGrad	Adaptive Gradient
RMSProp	Root Mean Square Propagation
ADAMS	Adaptive Moments
ANN	Artificial Neural Network
RBM	Restricted Boltzmann Machine
RGB	Red, Green, Blue
ReLU	Rectified Linear Unit

ILVRSC	ImageNet Large Scale Visual Recognition Challenge
RESNET	Residual Neural Network
TACO	Trash Annotations in Context
MAP	Mean Average Precision
IOU	Intersection Over Union
OS	Operating System
TP	True Positive
FP	False Positive
PC	Personal Computer
CSV	Comma Separated Values
RTF	Ready To Fly
ESC	Electronic Speed Controller
FPV	First Person View
DC	Direct Current
PDB	Power Distribution Board
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
Li-Po	Lithium Polymer
PWM	Pulse Width Modulation
BEC	Battery Elimination Circuit
VTOL	Vertical Takeoff and Landing
UART	Universal Asynchronous Receiver/Transmitter
RC	Radio Control
QGC	QGroundControl
HD	High Definition
ASV	Autonomous Surface Vehicle
OTG	On The Go
TDM	Time Division Multiplexing

NOTATION

d	Input layer dimension
J	Hidden layer dimension
K	Output layer dimension
W^h	Weight matrix between input layer and hidden layer
W^o	Weight matrix between hidden layer and output layer
w_{ij}^h	weight from i^{th} input node to j^{th} hidden node
w_{jk}^o	weight from j^{th} hidden node to k^{th} output node
b^h	Input layer bias vector
b^o	Hidden layer bias vector
ϵ_{SSE}	Sum of Squared Error
ϵ_{CE}	Cross Entropy Error
x	Input
y	True output vector
w	Weight of a single connection between two neurons
b	Bias
s^o	Predicted output vector
ϕ	Activation Function
a	Pre-activation vector
s	Post-activation vector
B	Batch Size
ΔW	Weight update
η	Learning Rate
H	Hessian Matrix
F_w	Feature map width
F_h	Feature map height
W_{im}	Width of image

H_{im}	Height of image
$\tilde{s}$	Stride Parameter
R_k	Receptive field size
P	Precision
R	Recall
x_c	x-coordinate of the top left corner of the predicted bounding box
y_c	y-coordinate of the top left corner of the predicted bounding box
w_c	Width of the predicted bounding box
h_c	Height of the predicted bounding box
θ	Orientation of drone with magnetic North
ω	Camera angle with the horizontal
β	Camera angle with the vertical
lat	Drone latitude
$long$	Drone longitude
fov_h	Field of view of camera - horizontal
fov_v	Field of view of camera - vertical
f	Force developed by each motor
k	Constant of proportionality
ω_i	Angular velocity of the i^{th} motor
F	Force vector
τ_M	Moment produced by each motor
τ	Torque vector

Chapter 1

INTRODUCTION

Water pollution is a severe environmental concern that has plagued humanity and wildlife alike from centuries. There are two types of water pollution - surface water pollution and groundwater pollution. Surface water pollution involves the contamination of lakes and rivers, usually due to rain water run-off or dumping of industrial waste. Groundwater pollution is mostly due to leaching from agricultural fields, open mines, etc., and is dependent on various factors, such as land porosity and seepage. Sources of water pollution are either point sources, like sewage discharge or non-point sources, such as agricultural leaching. Pollution of water bodies affects marine life, and through biomagnification affects humans too. Waterborne diseases have swept through many countries, due to poor management of lakes and rivers. Clogging up of water surfaces due to accumulation of floating litter reduces the amount of available oxygen, causing marine life to die out and the entire lake to dry up slowly, in a process known as eutrophication.

One of the most critical causes of water pollution is litter accumulation on the surface. This is a severe environmental problem contributing to problems such as loss of marine life, drying up of lakes, among others. Large garbage patches, such as 'The Great Pacific Garbage Patch,' are known to form in oceans, carried to one spot by water currents. A majority of garbage pieces commonly found in oceans consist of items that used daily, such as plastic bottles, plastic and paper bags, beverage cans, and so on. According to estimates, over 8 million tons of plastic enter the world's oceans every year. The Worldwatch Institute reports that at least 267 species of marine life are known to have suffered from ingesting trash or getting entangled in marine debris.

As shown in the pie-chart in Figure 1.2, the largest contribution to floating trash is from plastic and paper. Wood, cardboard, metal and glass form significant proportions. Glass and metal show a tendency to get stuck in weeds and shallow water. At the same time, paper and plastic are more prevalent in deeper regions.

Figure 1.1: Polluted water bodies

Source: *TiredEarth News*

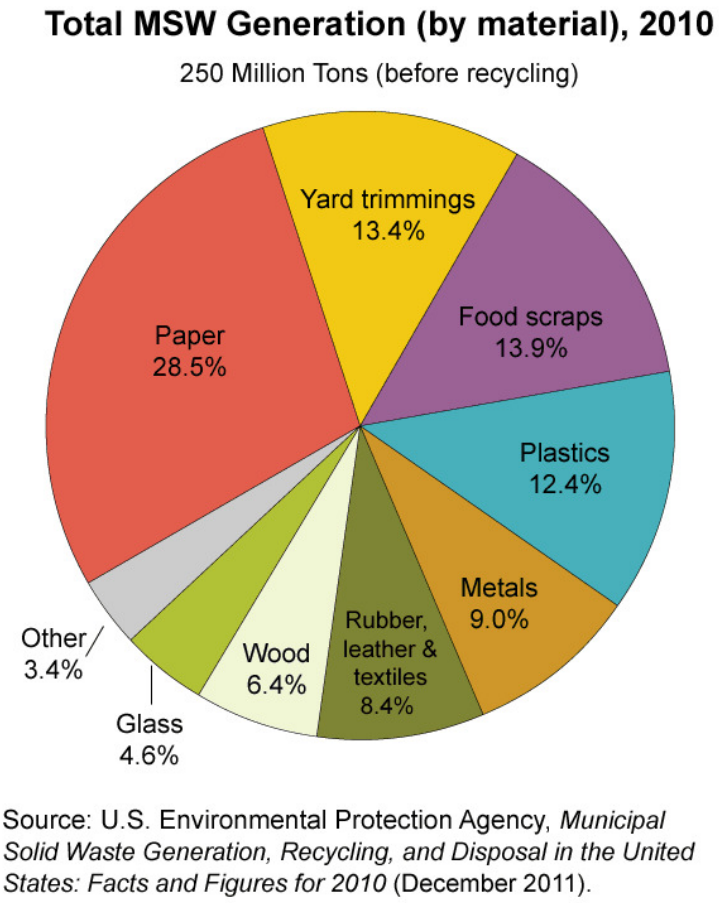


Figure 1.2: The most common types of floating trash

The situation in India is also similarly bad. There has been an alarming decrease in the amount of freshwater, leading to droughts in many areas, a recent example being the water crisis in the city of Chennai in 2019. Lack of adequate seasonal rainfall causes widespread damage, with thousands of people losing access to clean water. The amount of untreated waste water released from industries has been continuously increasing every year. There is an urgent need to augment water treatment and sanitation efforts to remedy the situation.

The primary type of garbage that is of particular danger is plastic. Plastic is almost non - degenerative and also extremely poisonous to marine life. Depending on size and density, plastic waste presents different levels of threat and requires varying methods of extraction. The three main types of plastic are :

- Nano-plastics, the smallest bits that are not even visible to the human eye, come from cosmetic products and consumer goods as well as through the breakdown of larger plastic particles. They are tough to detect or remove from the system. They enter the food chain through marine life and are eventually passed on to humans.
- Micro-plastics, in the millimetre size range come mainly from the breakdown of larger pieces of plastic, such as clothing, fishing nets, etc.
- Macro-plastics consist of large plastic items and include everything from small everyday objects such as plastic bags, packaging material, cosmetics and medical waste, to lost or discarded lines or fishing nets that can be miles in length. These items can choke or entangle birds and marine life.

This project focuses on the accumulation and disposal of mainly the third kind of plastic waste, in addition to other garbage commonly found in lakes. Plastic in the ocean accumulates in patches, but in inland lakes, the situation is very different. Strong winds break up the patches of litter and disperse them all over the lake, creating lots of difficulties during the cleaning process. A study by Matthew J. Hoffman and Christy Tyler, Professors at Rochester Institute of Technology, confirms this phenomenon by analysing garbage accumulation patterns in the Great Lakes in the United States of

America. It is estimated that around 10000 tons of plastic enter the Great Lakes annually. The team used computational models to map the movement of litter once it enters the water. It was found that litter patches were only temporary and were quickly dispersed by winds. Thus, just cleaning along the shores and targeting litter clusters is insufficient to deal with this problem. In reality, plastic waste does not just move with the current or the wind. It also floats or sinks, based on its size and density. The ability to sink changes due to the action of the sun, waves, and even micro-organisms. The best way to alleviate this problem would be to reduce plastic usage and step up efforts to clean water bodies.

Apart from plastic, the current solution also focuses on eliminating five other types of litter, which are paper, cardboard, metal, glass, and trash (miscellaneous objects, usually food waste) from the lake surface.

Current efforts to clean up lakes utilize marine robots with various modifications and which almost always work under manual control. These solutions are inadequate to tackle the extent of pollution in the water bodies. Fast, economical solutions are urgently required to tackle this menace. Existing techniques require a lots of resources and manpower that hinders most clean-up drives. These techniques also involve loud diesel boats that disturb marine life and further aggravate the problem. This suggests that there is a need for an autonomous trash detection and collection module that will be faster and easier to handle than current technology.

1.1 Motivation

Though three-fourths of the Earth's surface is covered by water, only 2.5% of that water is freshwater, available in lakes and rivers. For millions of living creatures, this is the only source of water. Human activity is causing a drastic drop in the amount of available freshwater. More than 70% of industrial waste, sometimes untreated, is dumped directly into rivers. A significant part of water pollution is also caused due to the dumping of sewage and litter. These activities cause a shortage of oxygen in water, posing a

severe threat to the marine ecosystem. Marine organisms also ingest litter, mistaking it for food. The pollutants so ingested travel slowly up the food chain, eventually reaching man.

Most of the litter is non-biodegradable, and continues polluting the water body for thousands of years. Ultimately, this affects humans, who are dependent upon rivers to fulfil their basic needs. There are millions of people around the world with minimal access to fresh water. The reduction in the availability of clean water poses a direct threat to their lives.

The best solution would be to minimize the dumping of waste into rivers and lakes. However, the actual waste must be cleared too. Since the earlier solutions, such as manual collection of trash with a diesel boat, are not effective or environmentally friendly, a new solution is needed. The current work leverages the advancements of technology in the computer vision and robotics fields to contribute to solving this problem. This project is an excellent opportunity to provide society with cleaner water bodies. It is also beneficial to the environment, providing marine animals and plants with better chances of survival due to lesser disturbance from humans. Also, the objective of complete system implementation and deployment provides a window into all the problems faced in developing an idea and implementing it in reality.

1.2 Thesis Organisation

In this thesis the object detection and aerial robot modules are described in detail. The basic idea and the motivation for the project is explained in Chapter 1. Section 1.2 provides a brief outline for the three primary modules of this project, which are - lake surface coverage, trash detection and retrieval of trash. Flow charts describing how each robot interacts with the other and how data is transferred are included here.

Chapter 2 gives a concise background of the sub-modules, namely aerial and marine robots, deep learning and waypoint navigation, that form the core of the project. This

background includes the historical developments in the area, as well as basic mathematical formulations. The integration of these sub-modules is vital to achieve the aim of autonomous litter collection.

The review of existing literature is detailed in Chapter 3. The chapter begins with a literature review of the complete trash collection system. An analysis of innovative navigation algorithms used in these systems is also provided. The chapter continues to detail studies on efficient collection systems for the marine robot. It then moves on to detection systems explaining the robots and the sensors used by previous studies in the area. An analysis explaining the merits and shortcomings of each study is provided, along with an explanation of how the current work aims to address these issues.

Chapter 4 introduces the first part of the solution methodology - Object Detection using Deep Learning. A succinct mathematical explanation is provided, first for a generic Deep Learning model and then for Convolutional Neural Networks in particular. A review of popular CNNs is also provided, that serves as the base for the model used in this project. The chapter then describes how the present model was determined to be the best one, by providing a comparison with other frequently used models. Having decided the model, the next section describes the architecture of the network and the nuances of the detection algorithm. The section then gives an account of all the approaches used for training the model, which vary in the type of data used and model parameters tuned. Continuing the flow, the next section provides a mathematical description of the coordinate calculation algorithm which transforms the detections into global GPS coordinates.

The building of the quadcopter is described in Chapter 5, starting with a short mathematical force and torque model for the same. The components used for the structure, propulsion system, flight controller, sensing, video and communication are listed, with an analysis of the factors that led to the selection of the particular model and brand. The process of assembly and usage is also explained. The chapter ends with the calculation of drone parameters. The findings and highlights of this work are then presented in the final chapter, with recommendations for future improvement.

1.3 Solution Methodology

The proposed solution combines the power of artificial intelligence wit the versatility of robotics to aid the cleaning process of the desired water body. The current approach consists of a collaborative approach, with an aerial vehicle detecting trash and a marine robot collecting it. The drones will be equipped with basic sensors, such as a GPS sensor, an altitude sensor and an IMU, as well as a camera. The drones follow an optimal surface area covering method that is planned in the mission controller, to map the entire lake and record a video from a fixed height. The video is then analysed by the main computer that houses a custom made trash detection module which detects the position and type of trash present in the video. A specially designed boat travels on an optimal path that connects the detected coordinates to collect the trash. Such an arrangement requires lesser resources as well as manpower. This system, once fully developed, would be completely autonomous. Various additional modules can be added to increase efficiency and flexibility. The three main modules and their interaction can be seen in Figure 1.3 and are also detailed in the following sub-sections. A detailed flow chart showing the sub-modules is shown in Figure 1.4.

1.3.1 Lake surface coverage

Mapping the surface of the lake is a preliminary step performed to identify lake boundaries and to determine flight parameters such as total flight time, etc. The availability of maps of the area would simplify this process as the boundary information can be fed into the system, thus avoiding a preliminary survey. However, it would be easier to fly the drone over the surface and map the area using boundary tracking. Once this is done, and the region of interest is mapped, the drone will attain a fixed altitude and record the video of the surface. The pose of the drone, determined by the GPS sensor is recorded at each frame of the video. This data is sent to the imaging module. The usage of multiple drones increases the area surveyed and reduces the time taken to map the surface.

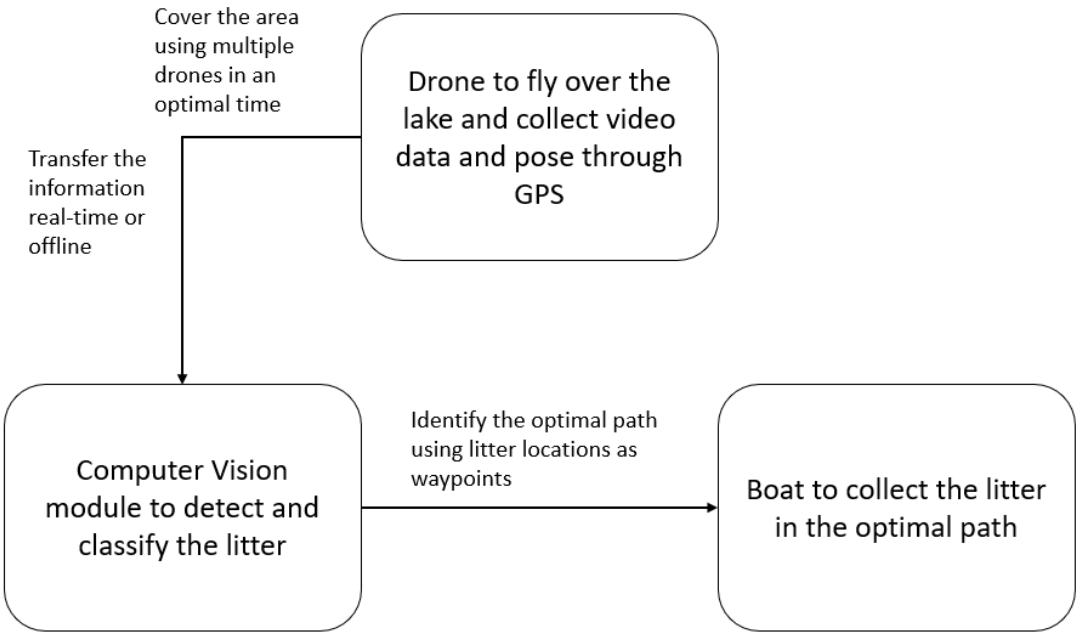


Figure 1.3: Basic System Flow Chart (*Generated in MS PowerPoint*)

1.3.2 Trash Detection

The computer contains the Computer Vision (CV) module that detects the presence of trash in the image and marks the corresponding coordinates. The video is sampled and converted to a list of images that is sent to the object detection module. This module returns the locations of the identified trash, as well as the trash category, in pixel coordinates. These coordinates are then converted to absolute world coordinates, by the Coordinate Calculation algorithm, using the per-frame GPS information obtained along with video data from the drone. The algorithm performs trigonometrical calculations based on altitude and camera angle information, to determine the latitude and longitude values of the trash, from the corresponding values of the drone. The CV module can also be modified to identify the shores or differentiate rocks and wildlife from trash.

1.3.3 Retrieval of trash

Once the drones finish mapping of the entire area, they return to the ground station for recharging. The identified trash coordinates are filtered, and the points that have a recurrence of more than 50% only are taken into consideration, to allow for misclassification and false positives in the detection. An optimal path is generated for efficient collection of all the pieces of litter, using the way-point navigation algorithm. To account for the movement of trash in the time before the deployment of the boat, a circle of probability is specified around the location given by the detection model. The specially modified boat is now activated and deployed. It follows the generated path, to facilitate easy and quick collection of trash. Modifications made to the boat include a conveyor belt system for trash collection, a tray to gather trash, and proximity sensors to detect obstacles in the path.

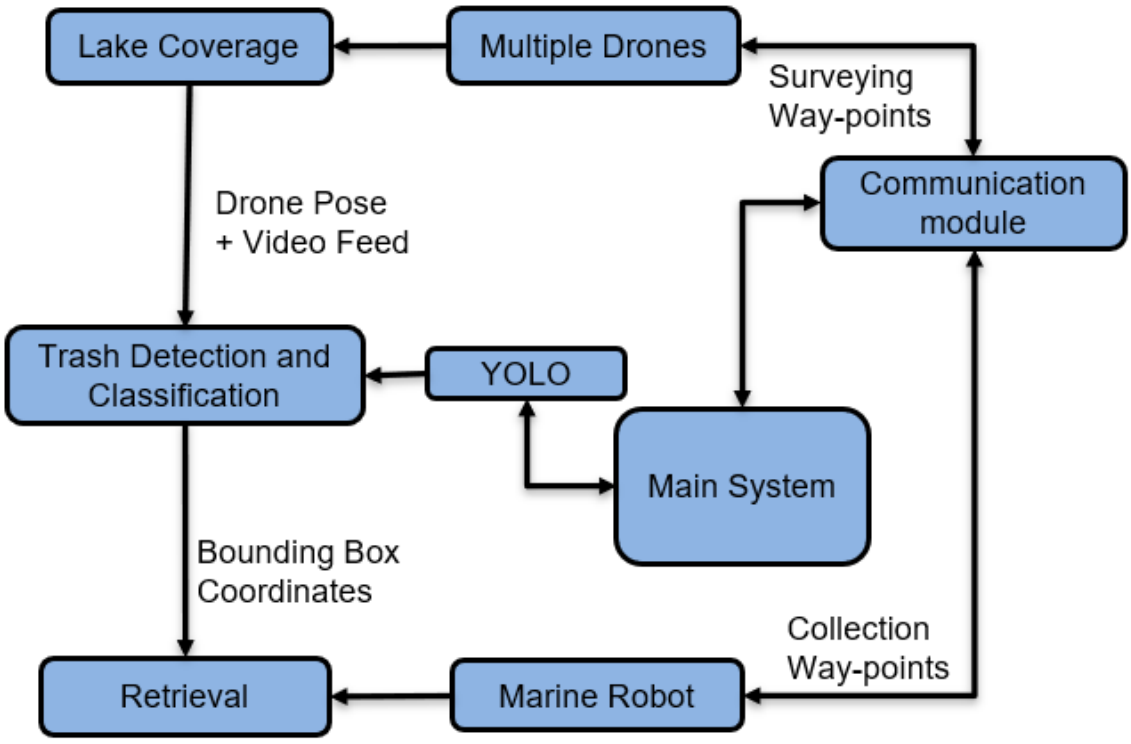


Figure 1.4: Detailed System Flow Chart (*Generated in MS PowerPoint*)

The main difficulty lies in the integration of various modules to facilitate autonomous use. The system can be modified in various ways to include more features and simplifications. For example, it can be extended to rivers and fast-moving water, or to track

litter and clear surface dredge and possibly, oil spills. Whatever the application, the problem lies in the procurement of proper images for training the detection module and also in ensuring compatibility between various components and the mission controller.

Once deployed, the system is expected to be of immense help to society by efficiently clearing floating litter. This is achieved with minimal human intervention, thus saving valuable resources. The system would be useful in urban areas with large lakes, that are covered with debris. In the current state, slow-moving rivers could also be cleaned using this system.

Chapter 2

BACKGROUND

This work involves the integration of several systems to achieve the objective of autonomous lake cleaning. It comprises of various separate modules, which need to communicate with each other in order to work autonomously. Each parameter and part of the system should be tuned accordingly. This section provides a short background for each module.

2.1 Aerial Robots

An aerial robot is defined as a powered vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable and can carry a lethal or non-lethal payload [21]. Aerial robots are usually used in the context of Unmanned Aerial Vehicles (UAVs) or drones.

Every UAV consists of a fuselage, avionics, actuators, energy storage systems, propulsion systems, launch and recovery modules and payload. The fuselage is the main body of the robot that houses the other components. Avionics consists of the flight controller, communication, navigation, and health monitoring systems. Actuators and propulsion systems provide the lift and thrust to the vehicle. Payload can be of many types, but generally takes the form of sensing equipment or delivery items.

UAVs are classified as Lighter than Air and Heavier than Air. Lighter than Air vehicles are buoyant in air, for example, airships and weather monitoring balloons. Drones and multi-rotors fall into the category of Heavier than Air vehicles. Detailed control algorithms have been developed to control the flight of heavier than air vehicles. Four

fundamental forces determine the flight of a UAV. These forces, as shown in Figure 2.1 are :

- Lift - The force that keeps the UAV airborne
- Drag - The retarding force
- Thrust - Provided by the engines, it propels the UAV forward
- Weight - Total load of the UAV

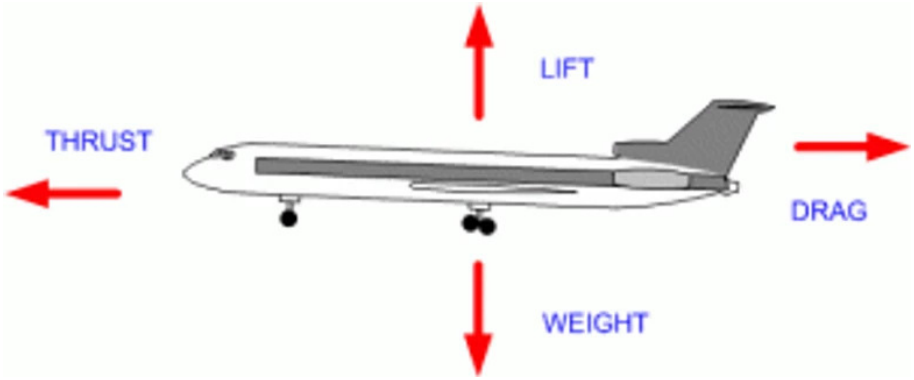


Figure 2.1: Basic Aerodynamic Forces

Source: *Aadarsh Poudel Geniuserc Post [23]*

Taking these fundamental forces into account, a mathematical model can be developed for a multi-rotor system, to determine the dynamics of motion. This dynamic model of the UAV is explained in detail in Chapter 5, for a quadcopter.

2.2 Marine Robots

Marine robots are termed as vehicles that are either submersible or travel on the surface of the water without any human onboard. They can be controlled either remotely or autonomously and can be programmed to carry payload or map areas. Submersible robots include Autonomous Underwater Vehicles (AUVs), Remotely Operated Vehicles (ROVs), etc. They are commonly used for oceanographic research and marine life expeditions. Surface Vehicles, on the other hand, consist mainly of different types of

boat-like vehicles, used for various purposes from research purposes to military applications.

Autonomous Surface Vehicles are more flexible than moored stations and are much cheaper than large boats and ships. They are flexible and can access shallow waters as well as narrow corners. These vehicles can be developed to complete a variety of navigation tasks and other complicated actions autonomously. They give mankind the potential to reach and research extreme conditions in the environment with little or no risk involved in the process.

These vehicles have multiple methods to get powered, including solar, wave, battery, and wind energy. They can be propelled by various means from classic rudder propeller systems to differential drive systems. The dimension of these vehicles depends upon the specific application it is designed to execute. Similarly, specific boat parameters like draft and height to gunwale can also be determined based on the mission environment. Surface vehicles operate in a 6 DOF environment and are influenced by a large number of forces, such as buoyancy, wind, wave force, etc. They are naturally unstable because of the momentum of moving water, and thus, the throttle needs to be armed to hold at a point in space. They can achieve a wide variety of tasks such as path and trajectory following, surveying and object collection. These vehicles are installed with multiple sensors to increase navigation capabilities, among other skills. Considering the fact that most of the oceans and water bodies around the world are unexplored, these robots have huge potential to assist human beings in many water-related activities.

2.3 Deep Learning - Object detection

Machine Learning is an iterative algorithmic process computers use, to extract patterns from data and make decisions. The field has grown tremendously, encompassing crucial areas, and becoming a vital part of life. In the most basic sense, machine learning seeks to obtain results without the need for continuous human involvement and programming.

Machine learning algorithms make use of specific attributes of the data called features. These features are pre-determined by the user, who processes the data in the form required. In cases where the number of features is huge, or features cannot be determined before-hand, these algorithms are not very efficient. These algorithms also encounter difficulties while extracting useful data from an extensive database. These problems led to the development of a new field called Feature Extraction. Deep Learning was initially introduced as an automatic feature extraction system, requiring minimum pre-processing by the user. Deep learning models have also shown the considerable prowess in identifying complex patterns after performing feature extraction. This is an old technique that has existed from 1940 and is known by different names such as - Cybernetics, Connectionism, etc. It was reintroduced as Deep Learning in 2007[4]. It is termed 'deep' because such systems often learn complex patterns by combining underlying simpler patterns. The sudden increase in popularity of this field was due to the rapid increase of high-speed computing resources in the early 21st century and the development of algorithms for training these networks.

The most popular of all Deep Learning models are Convolutional Neural Networks (CNN), which use mainly image data to identify similarities and also to determine patterns[17]. They take in image pixel information as input and learn patterns based on the RGB values of the pixels. More advancements in CNNs led to the development of sophisticated algorithms to process both images and videos and recognize objects present in them. New models such as Recurrent Neural Networks were developed to analyse sequential and temporal data. Rumelhart et al. and Hopfield's publications in 1980s [30] propose RNNs as weight sharing variants of CNNs. The concept of memory was introduced into the architecture by Schmidhuber and others in 1997 and named as Long Short Term Memory[10]. These models have led to great developments in object detection and tracking, image and video captioning, voice control, and self-driving cars, among others. The versatility of these algorithms has facilitated their percolation into areas of pure science and experimentation.

At the core of every deep learning model is a neural network, so named as it consists of nodes that are interconnected and fire in a specific sequence in response to input, very similar to the working of the brain. These neural networks have the power to ap-

proximate any type of function[19]. This led to their popularity in modelling complex patterns such as weather and climate studies, stock markets, and so on.

One of the fundamental problems that deep learning aims to solve is the identification and classification of objects in images. Object detection involves finding the presence of the object in the image, while classification determines the the category of the object as well. Object localisation not only detects and classifies an object in an image, but also determines its location in the image. Object localisation is highly important in areas such as obstacle avoidance and area mapping. The problem of object localisation has been solved by many leading researchers, using variations of CNNs, each of which has its merits and demerits. The last decade has seen great leaps in this area, with the introduction of Graphics Processing Units (GPUs) for model training. GPUs reduce the training time of neural networks by couple of orders of magnitude. Challenges such as ImageNet, conducted every year since 2010 have seen the development of CNNs like AlexNet, VGG, GoogLeNet, and RESNET, which sought to improve the accuracy of object detection and localisation. These networks differed in the number of layers, data preprocessing methods, and algorithm complexity, among others. Still better networks such as R-CNN, YOLO, and SSD were developed later, combining many modules of these early networks to achieve better results and more generalisability. Of these, a brief introduction is given in Section 4.2.

2.4 Waypoint Navigation

Waypoint navigation is one of those simple looking problems that are extremely complicated at their core. This module explains the problem of generating the trajectory for an Autonomous Surface Vehicle (ASV) to follow in ideal conditions, i.e, where there is no obstacle. A detailed discussion of the multi-node NP-complete Problem, famously known as the Travelling Salesman Problem is presented. This problem is different from other algorithms that find the shortest path, such as Dijkstra's and A*.

The objective of the Travelling Salesman Problem, is to cover all the defined ways

and return to the original node. The problem increases in complexity with an increase in the number of nodes. Solutions are presented for this problem, along with the corresponding codes. None of these solutions are efficient to the desired level. The branch and bound method worked very well and showed good speed in the trial example with IITM lake coordinates, leading to its selection for the current work.

The issue with using these algorithms with a large number of nodes lies in the requirement of more processing power and longer time to provide results. In such cases, a sub-optimal path can be accepted as the solution to save computational power and facilitate quick planning of missions. Several algorithms are introduced in the waypoint planning module to solve the TSP with a high number of nodes, which are, 'The Greedy Algorithm', 'Christofides Algorithm' and 'DoubleTree'. These algorithms promise the worst-case limit of the solution in terms of the optimal path. For example, the Christofides Algorithm guarantees that the length of the path determined will be within 1.5 times the length of the optimal solution. The chosen algorithm performed well to find the sequence of nodes to be traversed by the Autonomous Surface Vehicle.

Chapter 3

LITERATURE REVIEW

This chapter outlines some of the work done in this area by researchers and debates upon the merits and shortcomings of each. It also explains how the current work aims to solve the mentioned problem differently and the elucidates the particular issues faced, using this approach.

Autonomous river and lake cleaning has been implemented in various ways and various degrees of autonomy by many research groups worldwide. In 2013, Ro-boat, an autonomous intelligent river cleaning robot, was conceived[32]. It incorporates boat design and a computer vision algorithm, along with robotic arms and solar charging to make the system fully autonomous. The prototype was designed and built and is currently undergoing modifications to increase the volume of trash collected. The CV algorithm uses HSV Color Space and SURF to provide measurements to the Kalman filter, which ensures proper pollutant tracking. It is estimated that a system of 50 robots would be able to clean a river in a city in 6 months.

The Ro-boat is equipped with an array of solar panels and a double helix motor that uses lesser energy than conventional motors. The boat can clear all kinds of trash, including chemical waste. It works by sucking in water and filtering it. An added facility is the ability to sink partially and clean the bottom of shallow ponds.

Ro-boat, though efficient, would benefit from current state-of-the-art object tracking modules, which this work has implemented. Also, the involvement of drones in this work is expected to speed up the process of detection considerably than using only boat sensors for the task. However, the collection mechanism is something that could be a possible future modification, as a viable means of clearing liquid waste.

A study conducted by researchers at the Indian Institute of Technology Kanpur describes an autonomous boat system that uses a tactile sensor array to detect trash and a novel 'Recruitment Algorithm' for navigation[2]. They have designed the boat with a track belt system for trash collection and an oxygen pump to re-oxygenate lakes. The navigation algorithm is based on the method by which bees find nectar-filled flowers. When trash is found by one boat, a signal is sent to the other boats, which then travel to the location to aid in cleaning. This method, though useful, wastes a lot of resources due to the inefficient way of surface coverage. Tactile sensors have a small range and would not be of particular use in areas with high trash concentration. Also, the robots did not have a high speed either, reducing the area covered in a given amount of time. The system would also face problems with moving trash. Proper identification of the location of trash is essential to speed up the process. This reduces the load on the boat sensors, as now they are already provide with a first estimate of the location of the trash and thus can function faster.

Since 2008, Chinese researchers have been working on litter retrieval systems. Most of them used a single boat as the collector and hence were very slow and infeasible for large water bodies.

Other studies published in IJSART[12], JETIR, IJCMES[33], IJAER[28], and IRJET provide detailed design analyses and calculations for different models of trash collecting boats, using conveyors and belt drives as collection mechanisms. These studies show the viability of conveyor systems as efficient trash collection systems. The conveyor belt mechanism is shown to be superior to other methods of collection, such as pickup and filtration. All these systems are controlled remotely by humans and sometimes even lack sensors to identify litter. A suitable and optimal boat design can be developed using the ideas and mechanisms provided.

Urban Rivers, a Chicago based non-profit organization, developed a trash collection boat that cleans the Chicago River[1]. It works on remote operation by people controlling it through a website. The idea is to make a game-like system to encourage people to play and control it. The boat is equipped with cameras and GPS sensors, but the

identification of trash is done by humans. The model is still in the development phase, with the security of the boat and the main system itself, being a significant issue. This method, though innovative, depends heavily on people's participation and suffers from implementation problems. Also, there is no specific module to detect litter, thus necessitating human involvement.

Students from the Northwestern Polytechnical University in Xi'an, China developed an 'Orca Water Cleaning Robot' to clean the Yangtze River, in May 2019[11]. The boat is fitted with robotic arms to lift trash. The maximum payload is around 50 kg. The boat is also able to move at a moderately high speed of 1 m/s. The system also incorporates a water quality monitoring module. Work is on to include more functionality and sensors into the boat and introduce autonomous navigation. The system is efficient but is still not entirely developed for multiple robot interaction. This severely limits the speed of lake surface coverage. Also, the boat is remote controlled and can be made to follow a pre-programmed path. Sensors identify obstacles, reducing the load on the operator.

A paper published in IEEE 2018 presents a garbage pickup robot that travels on land[3]. The robot detects trash on the ground using a deep neural network and uses a novel navigation algorithm for guidance. It uses CNN to identify trash and the available area for movement. This work is very similar to the current project, in the fact that it uses a deep CNN to identify trash. They, however, use the CNN even to identify space for the robot to move. Also, the detection and collection modules are placed on the same robot, making the process time-consuming. In the case of the current project, the detection and collection modules are separated to reduce the load on each robot. Also, in the current work, navigation algorithms are used to identify the optimal path once the entire area is mapped, unlike the other methods where there is no optimal path. The robot simply moves in whatever direction it sees trash in, which can lead to a lot of unnecessary movement and incomplete trash picking. In the current approach, both complete trash collection and optimality of paths are ensured. Also, this robot cleans water, while it can be similarly used to identify trash on land. The collection could also be achieved by modifying a land vehicle.

Chapter 4

OBJECT DETECTION

In this section a detailed explanation for each of the object detection approaches is provided. An introductory background about neural networks is also given, along with mathematical derivation.

4.1 Object Detection - A Mathematical Treatment

4.1.1 Basic Neural Networks

Neural networks are at the heart of any deep learning model. Modelled on the working of the human brain, they consist of a group of individual nodes or neurons with interconnections between them, which, through a method of iterative error minimisation or training, learn patterns contained in data. A basic neural network is shown in Figure 4.1.

Each interconnection has a specific weight attached to it, that is initially assigned randomly or with the use of special initializations, and is later updated according to the error the network produces. In the network shown, there is an input layer of dimension d, one hidden layer of dimension J, and an output layer of dimension K. The weights between the input and the hidden layer are given by W^h, of dimension dXJ and the weight matrix between the hidden layer and the output layer is given by W^o of dimension JXK. The weight of the connection of the i^{th} input node to the j^{th} hidden node is given by w^h_{ij}. The weight of the connection of the j^{th} hidden node to the k^{th} output node is given by w^o_{jk}. In addition to the weights, vectors b^h and b^o, called bias vectors are also defined to account for the case when all the node values are zero.

Finally, the trainable parameters are given by :

$$\{W^h, b^h, W^o, b^o\} \tag{4.1}$$

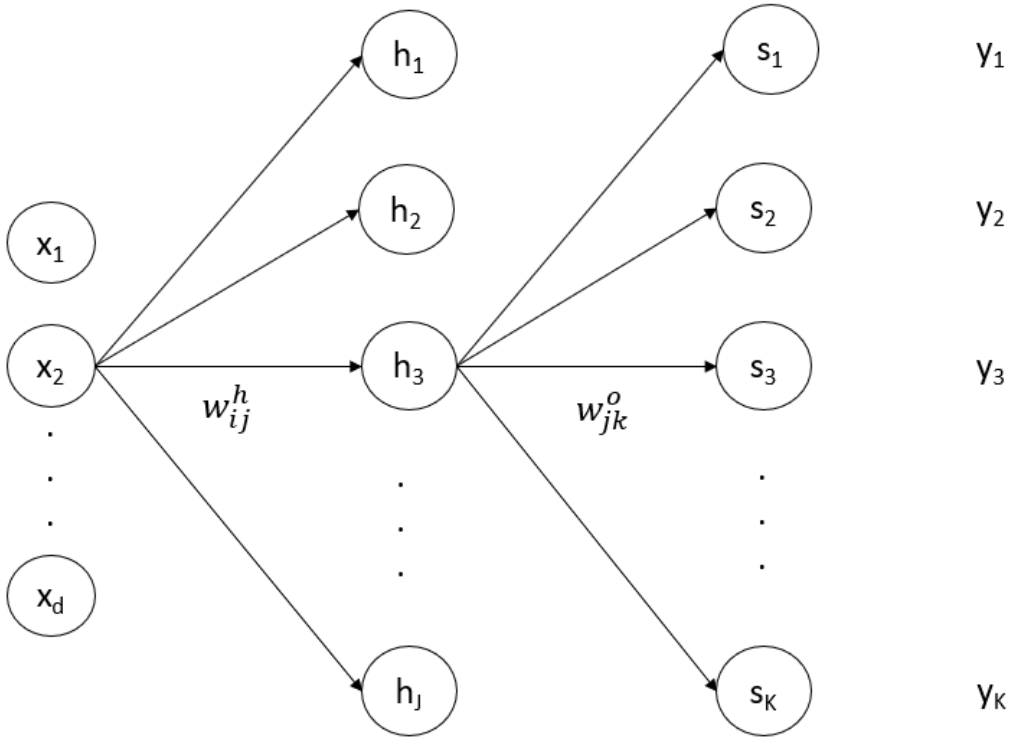


Figure 4.1: A Basic Neural Network (*Generated in MS PowerPoint*)

These parameters are updated based on the loss function after each data point is input, using a method called the Backpropagation algorithm. The loss function defines the difference between the true value and the predicted value. It is of many types, the most common ones being the Sum of Squared Error (SSE) and the Cross-Entropy (CE) functions.

$$\epsilon_{SSE} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{m=1}^{K} (y_{nm} - s_m^o)^2 \tag{4.2}$$

where N is the number of examples and K is the number of nodes in the output layer. The y vector is the true output vector.

Cross-Entropy over all the N examples :

$$\epsilon_{CE} = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{K} y_{nm} \ln s_m^o \tag{4.3}$$

where N is the number of examples and K is the number of nodes in the output layer. The y vector is the true output vector.

At each node, the input vector is transformed by a function known as the Activation function ϕ. Each node is divided into two parts as shown in Figure 4.2, one before activation and one after activation.

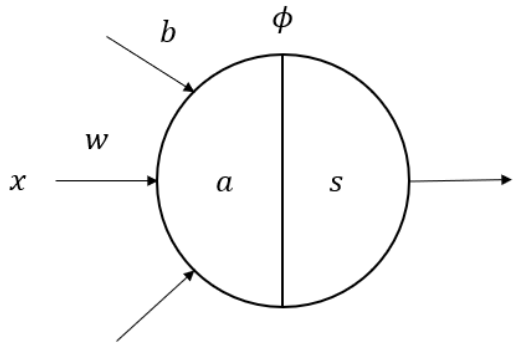


Figure 4.2: A single node of the network (*Generated in MS PowerPoint*)

The pre-activation vector is given by :

$$a = Wx + b \tag{4.4}$$

After activation, the output vector is :

$$s = \phi a \tag{4.5}$$

The vectorised equations for calculating the values at each node are as follows :
For the first hidden layer,

$$s^h = \phi^h a^h = \phi^h(W^h x + b^h) \tag{4.6}$$

For the second hidden layer,

$$s^o = \phi^o a^o = \phi^o(W^o s^h + b^o) \tag{4.7}$$

After defining the transformations at each node, the weights must be updated according to the magnitude of the loss function. This is again done in many ways, the three most

famous being Pattern learning, Batch learning, and Mini-batch learning. In the Pattern mode of learning, the weights are updated after the input of every example, which has the effect of faster convergence. In the batch mode of learning, weights are updated after all the examples are sent into the network, thus performing only one update for each epoch (One epoch is one pass of the network through all the examples). In the mini-batch mode of learning, a smaller batch size B is defined, and the weights are updated after sending in B data points.

The weights are updated in proportion to the change in the loss function value with a change in weight value. This is called the Gradient Descent Rule[13].
For the m^{th} example, the weight update is given by :

$$\Delta W(m) = -\eta \frac{\partial \epsilon}{\partial W}\bigg|_{W(m)} \tag{4.8}$$

where η is called the Learning Rate and determines the speed of convergence to the minima. It is a hyper-parameter that needs to be tuned to reach the optimum solution faster or more accurately. The partial derivative of the error function with respect to the weights is calculated by the Backpropagation Algorithm[**?**]. It starts from the output layer and traces back to the input layer, to examine the effect of the change of each weight upon the final loss function value.

Since there are two types of weights in the network defined above, there are two derivatives, one for W^h and the other for W^o. These are given by :

$$\frac{\partial \epsilon}{\partial w^o_{jk}} = \frac{\partial \epsilon}{\partial s^o_k} . \frac{ds^o_k}{da^o_k} . \frac{\partial a^o_k}{\partial w^o_{jk}} \tag{4.9}$$

$$\frac{\partial \epsilon}{\partial w^h_{ij}} = -\frac{1}{2} \sum_{m=1}^{K} \frac{\partial (y_m - s^o_m)^2}{\partial s^o_m} . \frac{ds^o_m}{da^o_m} . \frac{\partial a^o_m}{\partial s^h_j} . \frac{ds^h_j}{da^h_j} . \frac{\partial a^h_j}{\partial w^h_{ij}} \tag{4.10}$$

These values are calculated and substituted into Equation 4.8 to get $\Delta W(m)$. Equation 4.8 also has many variants such as Gradient Descent with Momentum[16], Nesterov Accelerated Gradient Descent, Adagrad[6], RMSProp, Adadelta[37], ADAMS[14], etc. All these are first-order optimization algorithms. Second-order optimization methods have also been devised, which use the second-order derivatives of the loss function in

calculating the weight update.

The most famous of these algorithms is the Newton's Method, which defines the weight update as :

$$\Delta W(m) = H^{-1} g|_{\theta(m)} \tag{4.11}$$

where, H is the Hessian matrix, given by :

$$H = \left[\frac{\partial^2 \epsilon}{\partial \theta_i \partial \theta_j} \right]_{i,j=1}^{M} \tag{4.12}$$

Due to the computational complexity in calculating the inverse Hessian, a class of algorithms called Quasi-Newton methods were developed that use various approximations for the inverse Hessian.

4.1.2 Convolutional Neural Networks

Images consist of a large number of pixels, which demand a correspondingly high number of input and output nodes. This prevents the use of simple networks such as ANNs, Autoencoders or RBMs from processing images, as these networks will take inputs from all the pixels at once, slowing down the training process. This is called Global Feature Extraction. A second method, called Local Feature Extraction, is used for image pixel analysis. In this method, only some parts of the image are taken as input for a fraction of neurons at a time, reducing the number of trainable parameters. The weights so trained are shared with the other neurons, which take different parts of the image as input, in a process called 'Weight Sharing', thus making sure that the entire image undergoes the same processing.

Convolutional Neural Networks have four types of layers, namely convolutional layers, activation layers, pooling layers, and fully connected layers. The critical operations that happen in each layer are outlined below. A generic CNN architecture is shown in Figure 4.3.

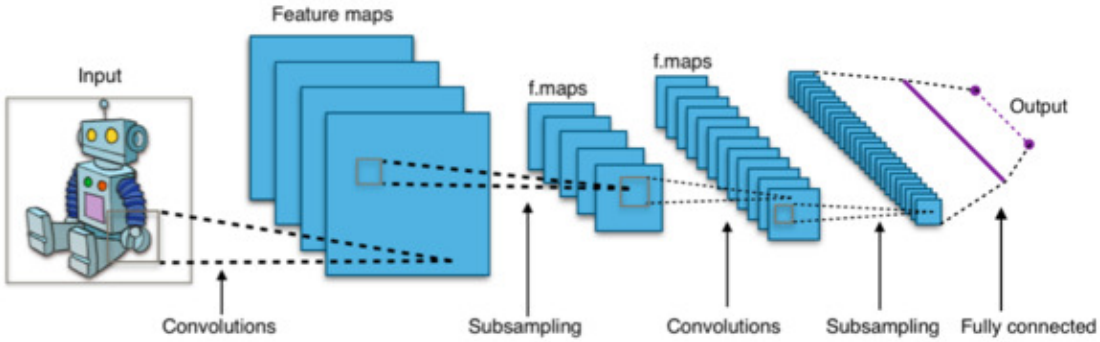


Figure 4.3: CNN Architecture

Source: *Wikipedia Article*

An image is divided into different parts to provide input to different neurons, using a 'Filter', also known as a 'Receptive field' or a 'Kernel'. The kernel is usually an array of numbers which when superimposed on the image, encloses some pixels, multiplies the pixel values with the kernel values, and returns the sum as a single number. This number is now considered as the representation of the group of pixels. The kernel now slides to next group of pixels and repeats the operation. The distance the kernel slides is defined by the 'Stride Parameter s.' This generates a smaller array of values as compared to the original number of pixels. The size of the kernel and the stride parameter are fixed beforehand. The new array generated is called a feature map of that particular kernel. The size of the feature map can be calculated using the simple formula :

$$F_w = \left\lfloor \frac{W_{im} - R_k}{\tilde{s}} \right\rfloor + 1 \tag{4.13}$$

$$F_h = \left\lfloor \frac{H_{im} - R_k}{\tilde{s}} \right\rfloor + 1 \tag{4.14}$$

Depending on the requirements, more Kernels can be used to generate more feature maps. Each kernel has a specific pattern of numbers that gives a high value if certain features are there in the area covered by it. This pattern is learned through training. Some examples of kernels are shown in Figure 4.4.

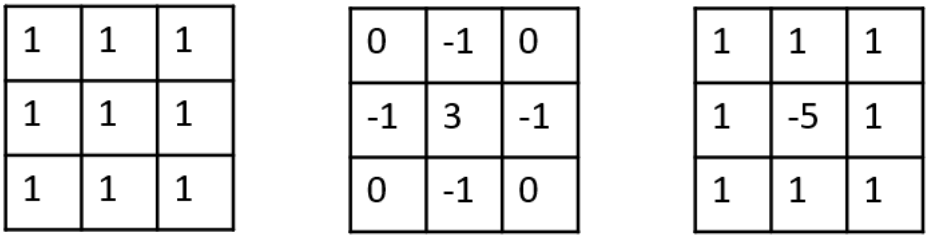


Figure 4.4: Different types of Kernels - Blur, Sharpening and Edge Detection (*Generated in MS PowerPoint*)

For example, sliding the edge detection kernel over the image will give a significant value of the result when there is an object edge present in that location, and provide a small output when the part of the image has no boundary. The generated feature map will reflect this situation. Passing this feature map through the next convolutional layer, where a new kernel is introduced, may identify that the edge is actually a curve, thus learning higher-level features of the object in the image.

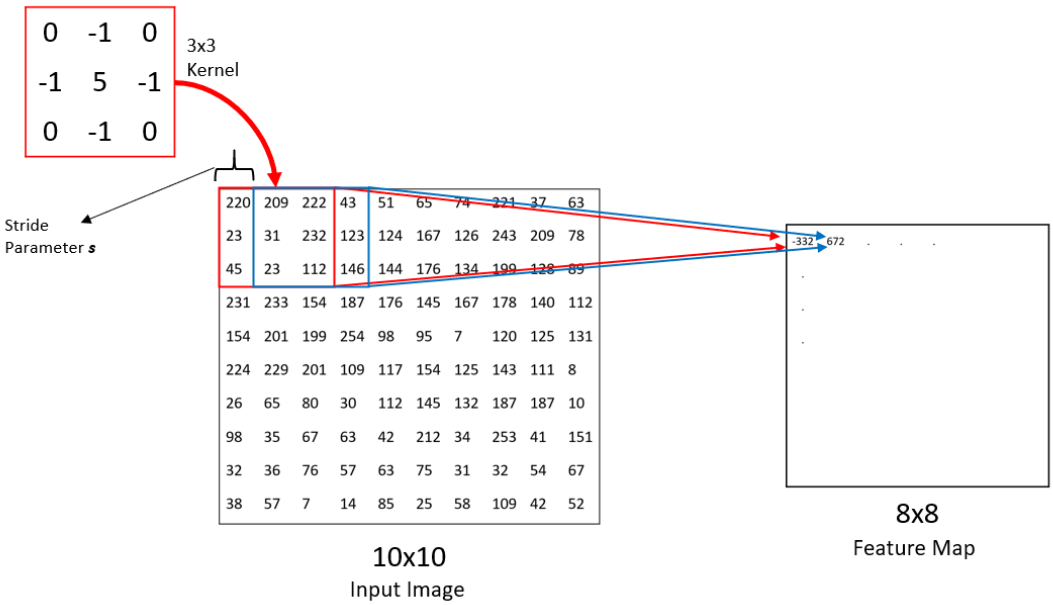


Figure 4.5: Operations in a Convolutional Layer (*Generated in MS PowerPoint*)

In Figure 4.5, shown above, the 3x3 kernel slides with a stride parameter of 1 and generates an 8x8 array from the input 10x10 image. This reduces the number of parameters from 100 to 64. Introducing more kernels (Figure 4.6) will create more feature maps, taking the final number of trainable parameters to *k*x64 for *k* kernels. Generating

a feature map in this fashion is called a convolution.

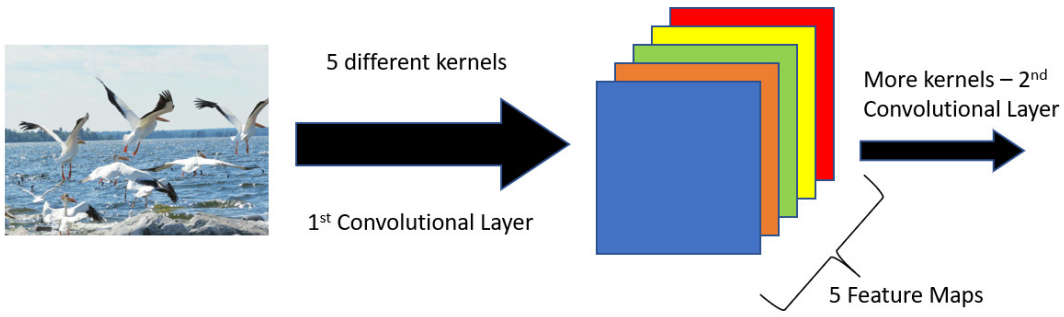


Figure 4.6: Example diagram showing multiple kernels for convolution (*Generated in MS PowerPoint*)

More convolutions are applied on the new feature maps, reducing parameters further. During the final training, the first convolutional layers learn to recognize low-level features such as points and edges in the images (Figure 4.7). The later layers learn more sophisticated patterns, combining edges and patterns as they are trained on the feature maps of the low-level features. Finally, the last convolutional layer is connected to a fully connected layer, that, based on the pattern of high-level features, identifies the class of the object.

CNN's also typically have pooling layers in between convolutional layers. Pooling layers work in the same way as convolutional layers. They perform simple tasks such as averaging all the elements in the kernel (Average Pooling - Figure 4.8) or returning only the most significant element of the kernel (Max Pooling). Pooling is another method of reducing parameters. The pooling kernel is pre-defined and is not trained.

During training, the class of the object is already known. The weights are trained using the backpropagation algorithm outlined in Section 4.1.1 and any of the optimizers described. The crux of the final weight matrix exists in the final fully-connected layers, which do not perform any convolution operations. Once the lower layers are trained to identify edges, corners, curves, and other low-level features, the final layers perform the

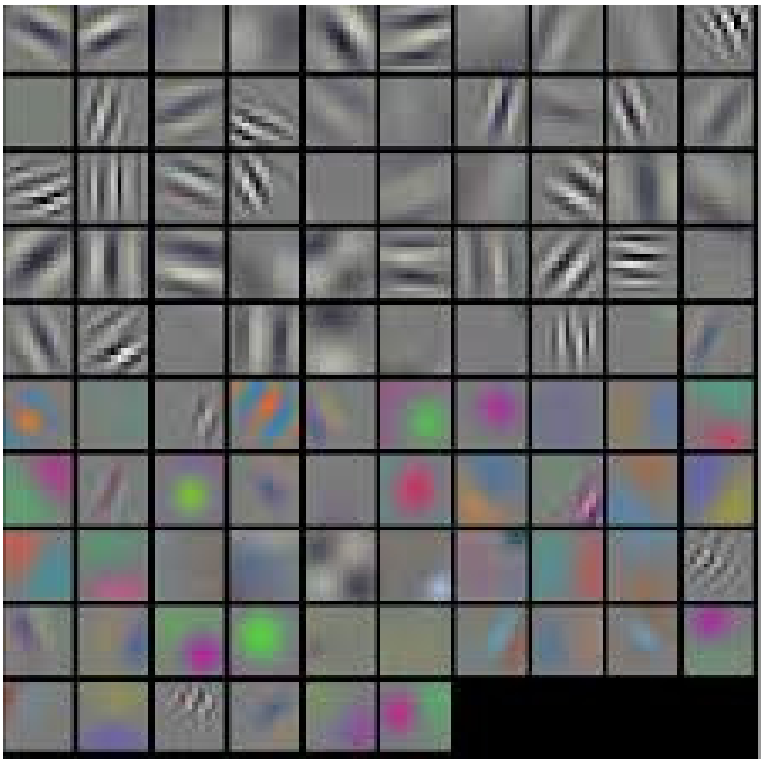


Figure 4.7: Visualisation of low level filters

Source: *Krizhevsky et al. [36]*

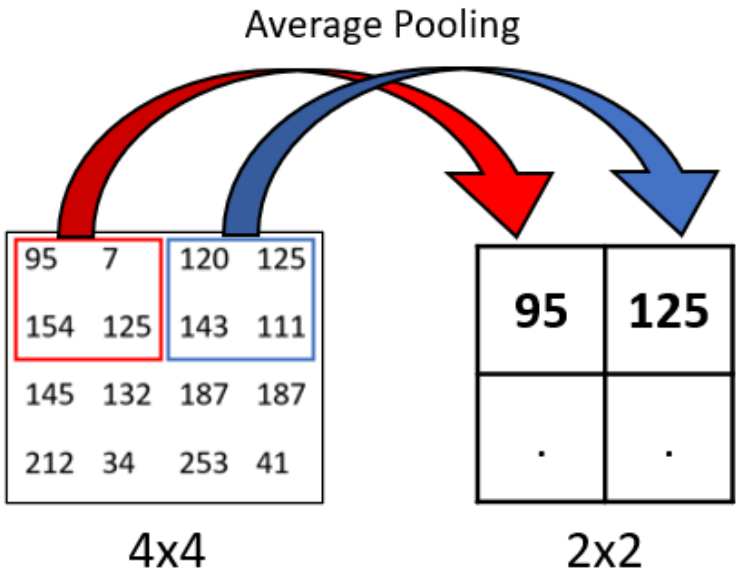


Figure 4.8: Pooling Layer - Average Pooling Example (*Generated in MS PowerPoint*)

task of classification and integration of these features. As all objects will have the same low-level features, sometimes, to train for new classes of objects, only the weights of the final fully-connected layers are updated. The lower convolutional layer weights are frozen and not updated during the training process. This method reduces computational load and training time, while giving good accuracy.

In the case of coloured images, the pixels hold three R, G, B color values, each ranging from 0 to 255. In this case, the filter used will be a 3D Filter, as illustrated in Figure 4.9. The rest of the training process remains the same as with black and white images.

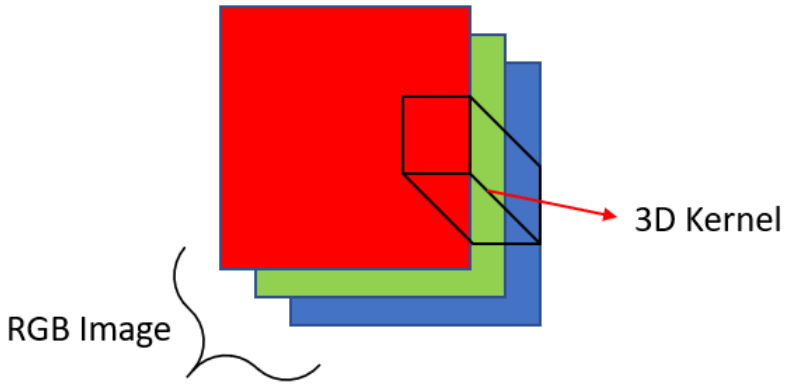


Figure 4.9: 3D CNN Kernel Visualisation (*Generated in MS PowerPoint*)

4.1.3 Important Networks

The ImageNet Challenge introduced object detection to the world. A few of the best networks that came out through this challenge are described here. The parameter used to estimate their performance is the top-5 error, which gives the fraction of the test examples for which the correct label is not among the five most probable labels.

LeNEt was the first network, developed by Yann Le Cunn, in 1998, using Sigmoid and Tanh activation to perform digit recognition[18]. It was a simple network with 60000 parameters that laid the foundation for sophisticated object identification later.

AlexNet was the first CNN that, in 2012, revolutionized object detection using ReLU activation, multiple GPUs, and overlapping pooling[15]. It achieved a top-5 error of 17%. AlexNet had 60 million trainable parameters, 650,000 neurons in 5 convolutional layers, three fully connected layers, and one softmax layer.

In 2014, another network called VGG was developed that used a smaller kernel size than AlexNet[31]. It also included 1x1 kernels for non-linear function approximation. This improved the top-5 error to 8%. VGG, due to small kernel sizes, could include a larger number of layers, with around 140 million parameters, leading to improved performance.

In 2014, researchers at Google Inc. developed the GoogLeNet, a 22 layer model that is most famous for the development of the Inception module[34] shown in Figure 4.10. The Inception Module uses different sizes of kernels on the same input and concatenates the output. Many such modules are stacked together. GoogLeNet succeeded in reducing the top-5 error to 6.67% and won the 2014 ILVRSC.

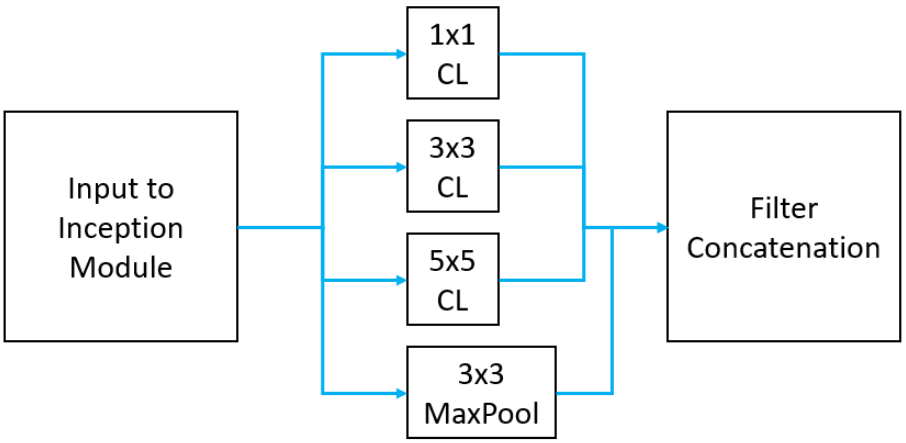


Figure 4.10: The Inception Module used by GoogLeNet (*Generated in MS PowerPoint - Adapted from Szegedy et al. [34]*)

In 2015, the Residual Neural Network or RESNET was developed by Microsoft, which solved the problem of training networks with a large number of layers[9]. RESNET introduced the 'Skip Connection', an identity mapping from previous layers to a layer in the front (Figure 4.11). This mapping added the residual from a previous layer and continued training. A function is used to match the dimensions between the layers. RESNET consisted of 152 layers and reduced the top-5 error to 3.6%.

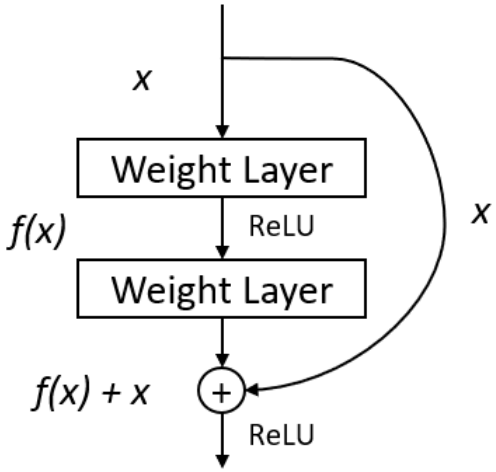


Figure 4.11: Skip Connection for RESNET (*Generated in MS PowerPoint - Adapted from Kaiming He et al. [9]*)

4.2 Model Determination

In recent years, several new and modified CNNs, mentioned in Section 2.3 were developed, each suitable for a particular type of data. Transfer learning was implemented on sample image data on each of these models to check the performance and to identify the best model for the current application.

The networks tried out were TrashNet[35], YOLOv2[25], R-CNN[8], SSD[20], and YOLOv3[27]. TrashNet is a modified form of AlexNet, tuned for trash detection. You Only Look Once, or YOLO is a highly sophisticated model that looks at the entire image and outputs classification. R-CNN and its variants are modifications to CNN, that

use region-based learning. SSD or Single Shot Detector outputs both classification and detection in one pass. The results of the training are shown in Table 4.1.

Table 4.1: Models tested for initial identification

Model	**Accuracy (%)**	**Speed (fps)**
TrashNet	75	8
YOLOv2	75	35
R-CNN	70	12
SSD	70	19
YOLOv3	80	45

Though the TrashNet model is reasonably accurate, it lacks speed, which is essential for real time usage. YOLOv2 and YOLOv3 show much better speed with excellent accuracy, with the only problem being the training time, which is quite high. R-CNN and SSD show decent accuracy, but once again, lack in the speed of processing. Keeping in mind the final goal of real-time operation, YOLOv3 was chosen as the model to be tested further.

4.3 Model Deployment

The biggest problem in trash detection is the non-availability of enough data for training. Either the dataset is too small or not in the suitable format for the task. Most researchers have used their own dataset, which has given satisfactory results. Existing online data sources were used for training. Trash Annotations in Context (TACO) is an evolving dataset that contains around 3000 images and annotations[24]. These images show trash with a variety of backgrounds, ranging from roads and grass to beaches and concrete. Moreover, most of the trash objects are quite small, occupying only around 1-10% of the image area. This will reduce the performance of CNN. It would also increase the training time. Also, the types of litter in the images consisted of more than 20 categories, which would be more challenging to train. Some sources contained images

of trash underwater, which resulted in poor quality images, unsuitable for the current application, as images captured by the drone camera would be of high-resolution. The primary aim was to sort the litter into six main categories. The second dataset (used by Trashnet) contained 2527 images consisting of 6 different classes - Cardboard, Glass, Metal, Paper, Plastic, and Trash. The number of images for each class is tabulated in Table 4.2.

Table 4.2: Number of images in each class in from the chosen dataset

Class	**Number of Images**
Cardboard	403
Glass	501
Metal	410
Paper	594
Plastic	482
Trash	137
Total	2527

These images contained close up views of litter in all angles and against a white background. The objects occupied 50-90 % of image area. The images were shot using Apple iPhone 7 Plus, iPhone SE and iPhone 5S. The picture resolution is 512x384. A sample image of each class is shown in Figure 4.12.

4.3.1 YOLO Architecture

Before explaining the architecture of YOLO, it is necessary to explain a few metrics which determine the performance of a model. In neural networks, the average loss is usually a key standard metric. In CNNs, other metrics such as Precision, Recall, F1 Score, Mean Average Precision (MAP) and Intersection Over Union (IOU) assume more significance.

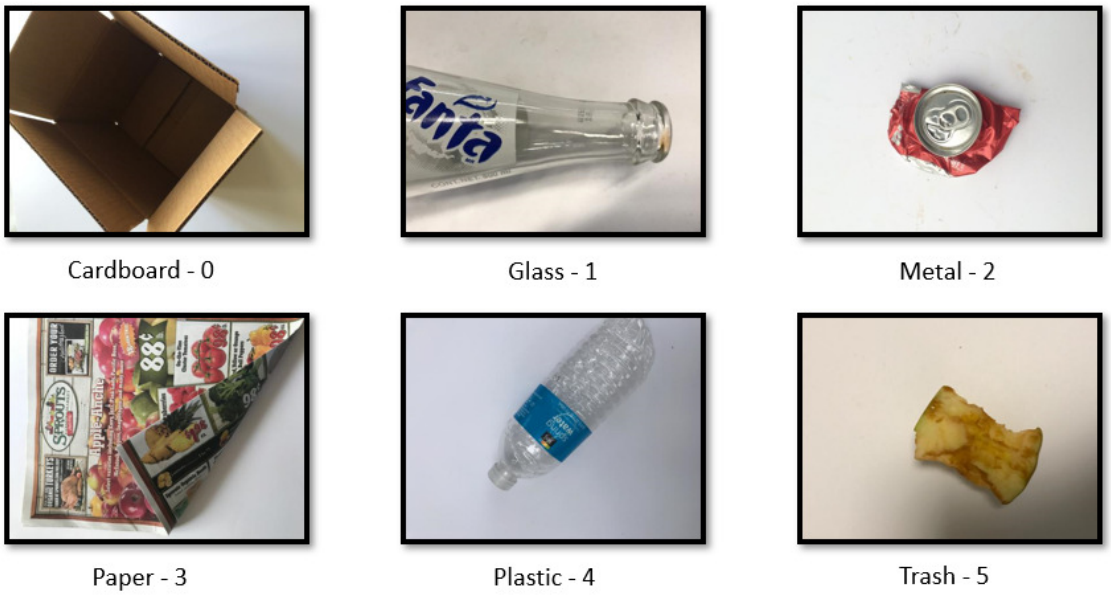


Figure 4.12: Sample images from the dataset with corresponding class labels

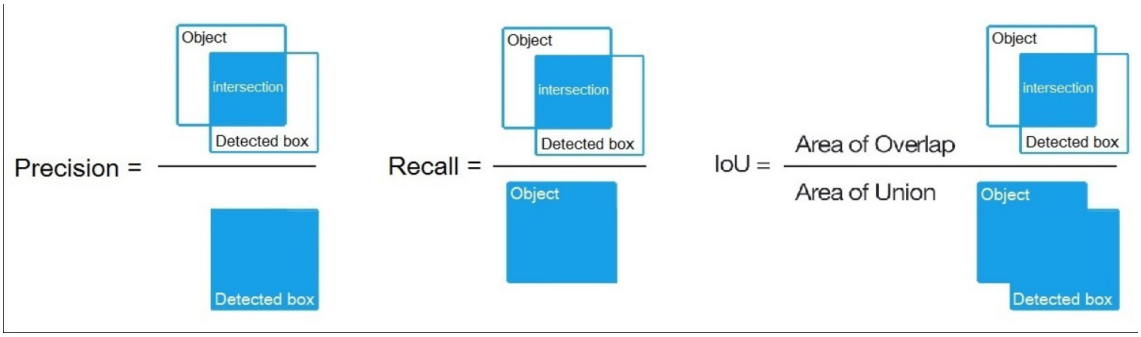


Figure 4.13: Evaluation Metrics for Object Detection (*Source : Alexey Bochkovskiy et al. [5]*)

As shown in Figure 4.13, Precision (P) is the amount of intersection as compared to the size of the detected box, while Recall (R) gives the amount of intersection as compared to the object size. The F1 score is given by the formula $\frac{2PR}{P+R}$. The IOU is the area of the intersection divided by the area of union of the bounding box and the object. The MAP is the mean value for the average precision for each class, where average precision is the average value of 11 points on the precision-recall curve, for different values of probability threshold of detection. These metrics provide a quantitative estimate of the performance of the model. A higher value for each of these metrics is usually an indicator of better performance[5].

YOLO is slightly different from the earlier CNNs. It does not use a slider or region-specific training. YOLO models the entire process of object detection as a single problem, in a single module, differing from models like R-CNN, where many separate modules must be trained to determine the final predictions. YOLO uses a modified version of the SSE loss function and the Leaky ReLU activation function. It is more generalizable than previous models. YOLO also is less likely to predict backgrounds as objects, a problem faced by R-CNN. In R-CNN, potential bounding boxes are first generated, over which a classifier is run, and post-processing is performed. The need to train each module makes R-CNN slower and computationally expensive. YOLO segments the image into an S x S grid. Each grid then predicts B bounding boxes, confidence scores and C class probabilities. A bounding box consists of five predictions - $x_c,\ y_c,\ w_c,\ h_c$, and confidence. These refer to the centre coordinates of the box, the width and height relative to the image and the confidence score of the class. The confidence score is defined as the prediction multiplied by the IOU. YOLO predicts multiple bounding boxes per grid cell, but outputs the prediction with the highest IOU. On the ImageNet challenge, YOLO gives an accuracy of 88%.

YOLO consists of 24 convolutional layers with 2 fully connected layers. It is inspired by the GoogLeNet Inception module but uses 1x1 and 3x3 convolutional filters. The architecture is shown in the figure below. A smaller version of YOLO exists, known as Tiny-YOLO, containing 9 convolutional layers.

YOLO finds it difficult to identify small objects in flocks or when the image is scaled or has a different aspect ratio. This is due to the limitation imposed on the grid cells that they can have only a fixed number of object classes inside them. The architecture of YOLOv3 is shown in Figure 4.14.

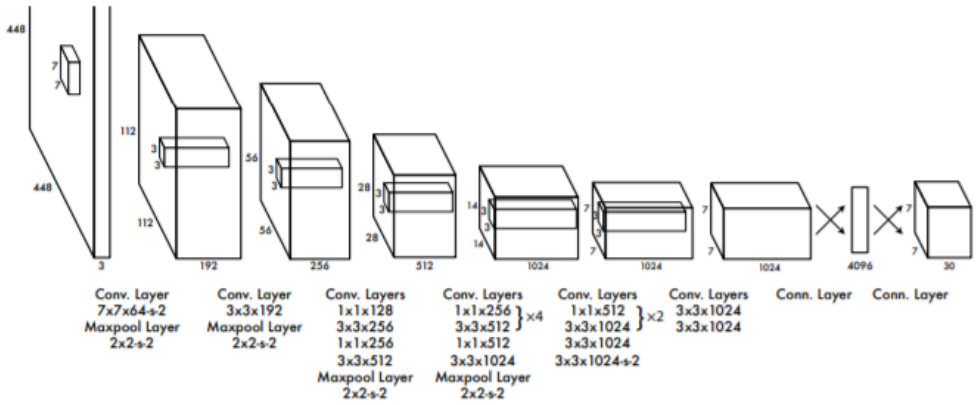


Figure 4.14: Architecture of YOLOv3 (*Source : Redmon et al. [27]*)

4.3.2 Approach 1

In the first method of training, the images from the original dataset were used without any modifications.

System specifications : Windows 10 OS, NVIDIA GEFORCE GTX 1050Ti GPU, Python 3.6, Tensorflow-GPU and Keras 2.1.2

Data augmentation was performed using HSV changes and 20% translation. The network was trained for 1200 iterations, each iteration taking 30s. The total training time was 17 hours. A batch size of 64 was used. Using an 80-20 ratio, the data was split into training and testing data. The loss graph for training is shown in Figure 4.15. The loss after 2000 iterations decreased to 0.25.

The results after training, tested on new images are shown in Figure 4.16 for single objects and in Figure 4.17 for multiple objects.

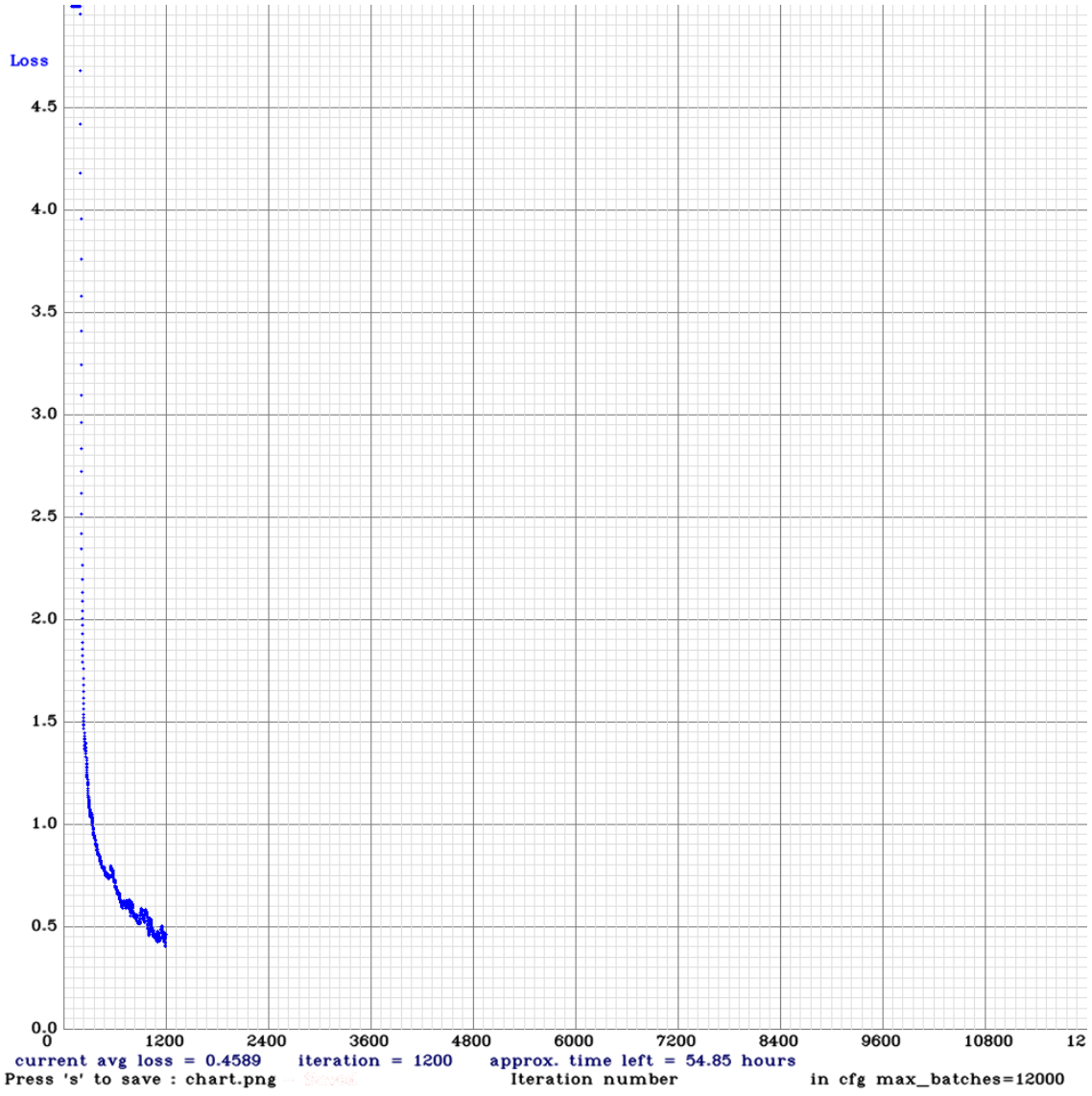


Figure 4.15: Loss Graph for 1200 iterations (*Generated using Python script*)

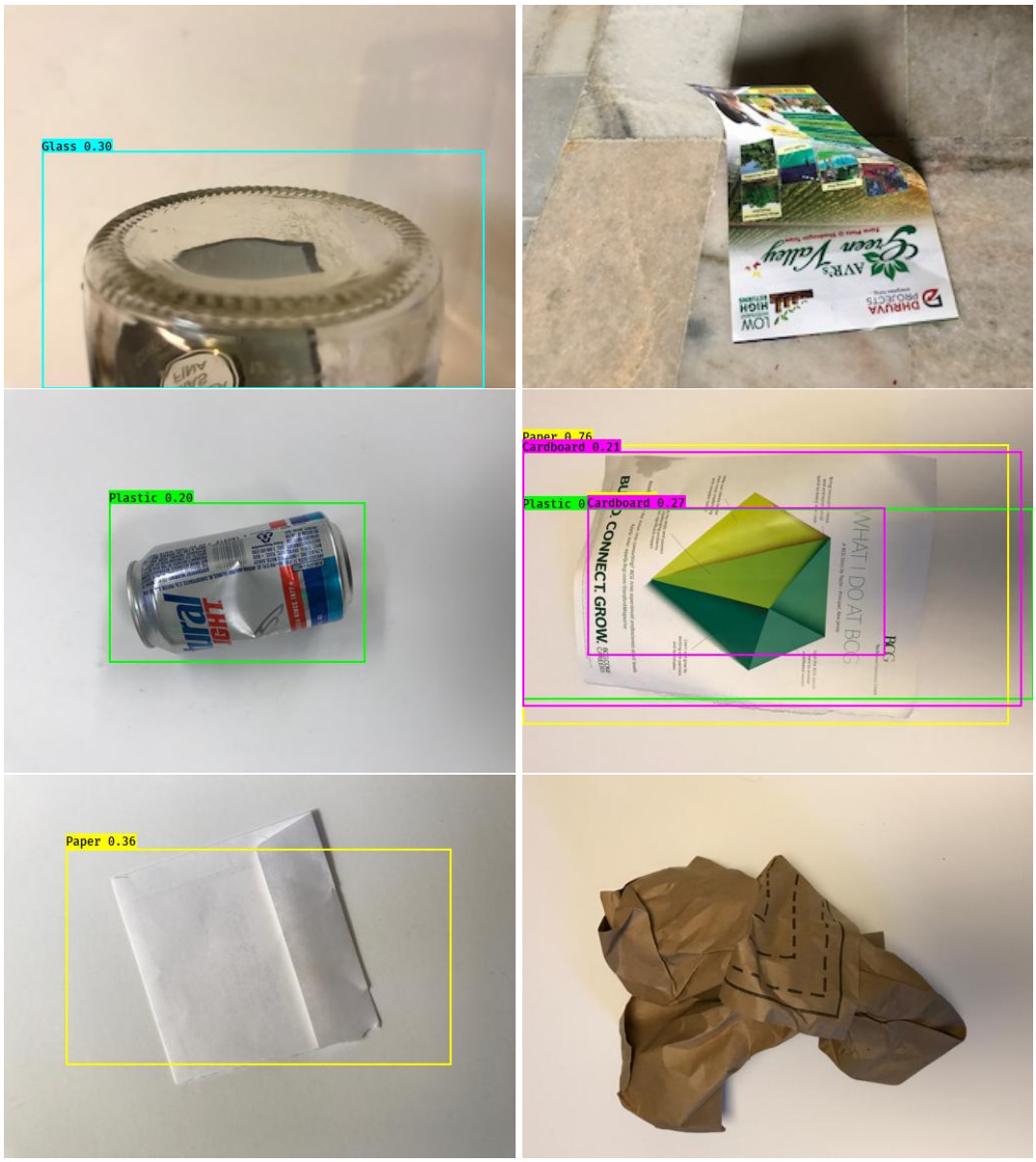


Figure 4.16: Detections for single object case - Approach 1 (*Generated using Python script*)

As can be seen in the images, for a single object, there are quite a few misclassifications as well as improper bounding box marking. In some cases, the object is not even identified. Reducing the threshold of detection might be useful in these cases, but it may also generate many bounding boxes for the same object - all the detections might have roughly the same probability. For images with more than one object, the model is highly irregular and error-prone. It draws large bounding boxes and identifies the background as an object too. A white background is given the 'paper' label, while a darker background is given the 'cardboard' label. It also covers the entire image with a

Figure 4.17: Detections for multiple objects case - Approach 1 (*Generated using Python script*)

Table 4.3: YOLOv3 results per class - Approach 1

Class	**Average precision (%)**	**TP**	**FP**
Cardboard	39.83	7	2
Glass	44.19	0	0
Metal	40.25	6	0
Paper	68.39	38	4
Plastic	39.72	15	5
Trash	7.81	0	0

Table 4.4: YOLOv3 overall results - Approach 1

Model Parameters	**Value**
MAP	40.03%
IOU	60.78%
Precision	0.86
Recall	0.13
F1 Score	0.23
Detection time	0.06s

single box and predicts three or four classes that could be inside it. Individual attention to each object is not seen in the detections. This is due to the objects in the training set occupying a large portion of the image area. YOLO struggles to identify objects that are on a much different scale than in the training set.

Also, looking at the average precision table (Table 4.3), it is seen that the model manages to identify paper to a decent level of accuracy, but fails badly in other cases. For the trash class, the model performs extremely poor. This could also be attributed to the larger number of paper samples in the training set. The MAP of 40% also reflects this poor performance (Table 4.4). A good IOU of almost 61% can be attributed to the drawing of large boxes, which is sure to increase the area of overlap. If the box is large, spanning the entire image, as in some cases, the area of the intersection will be nearly 100%, increasing the average IOU. This is reflected in the vast difference between the precision and recall values. The precision is nearly six times as large as the recall, showing that the object is huge and occupies a large portion of the detected box. The small value of the F1 score gives a final indication of the poor performance of the model. The detection time was found to be extremely good, at 60 ms per image, suitable for real-time implementation.

4.3.3 Approach 2

To remedy the problem of YOLO not identifying smaller objects, the original images were modified - re-scaled and pasted onto a white background. Each image was re-scaled by a factor of 8 and pasted at the centre of the white background. Two sample images are shown in Figure 4.18.

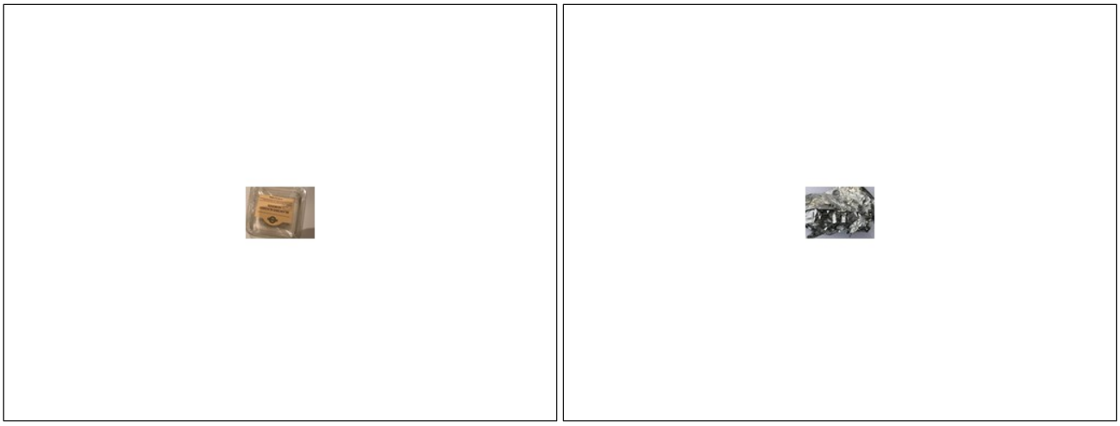


Figure 4.18: Rescaled Images for Approach 2 (*Generated using Python script and OpenCV packages*)

The new data was trained again using the same YOLO model for 3000 iterations. The final loss decreased to 0.1470 after 25 hours of training (Figure 4.19). The results for single object (Figure 4.20) and multiple objects (Figure 4.21) per image are shown.

Table 4.5: YOLOv3 results per class - Approach 2

Class	**Average precision (%)**	**TP**	**FP**
Cardboard	58.12	33	19
Glass	70.73	55	16
Metal	46.56	60	85
Paper	83.63	69	4
Plastic	63.36	51	19
Trash	46.79	18	33

The single object identification showed an improvement, with smaller bounding

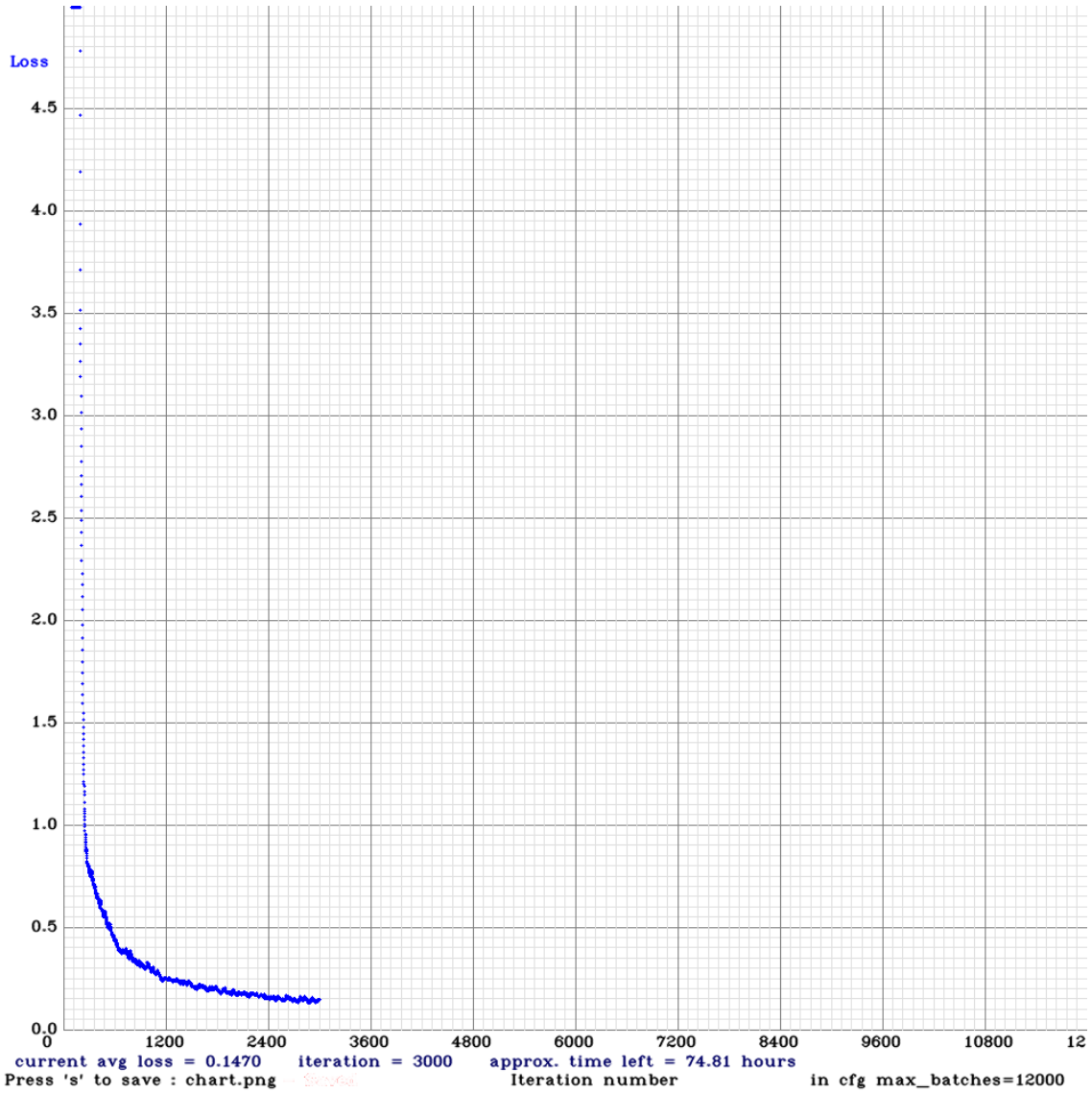


Figure 4.19: Loss Graph for 3000 iterations (*Generated using Python script*)

Table 4.6: YOLOv3 overall results - Approach 2

Model Parameters	**Value**
MAP	61.53%
IOU	45.53%
Precision	0.62
Recall	0.57
F1 Score	0.59
Detection time	0.07s

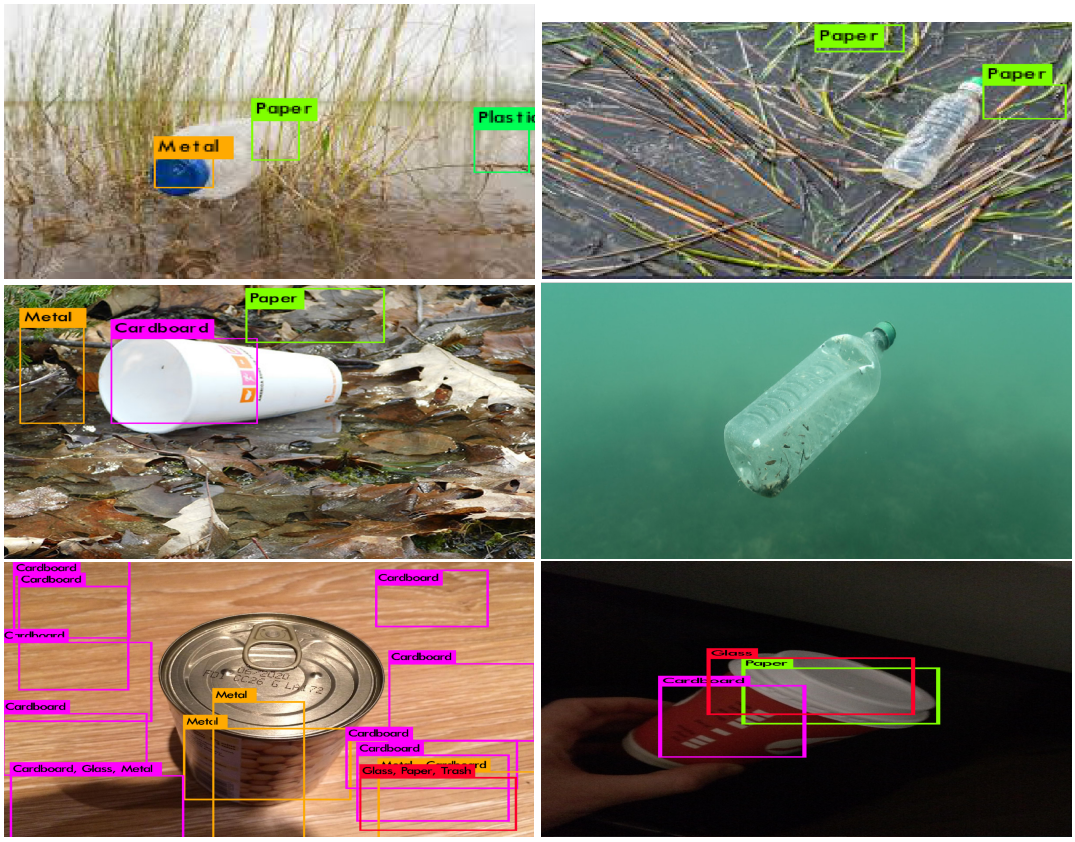


Figure 4.20: Detections for single object case - Approach 2 (*Generated using Python script*)

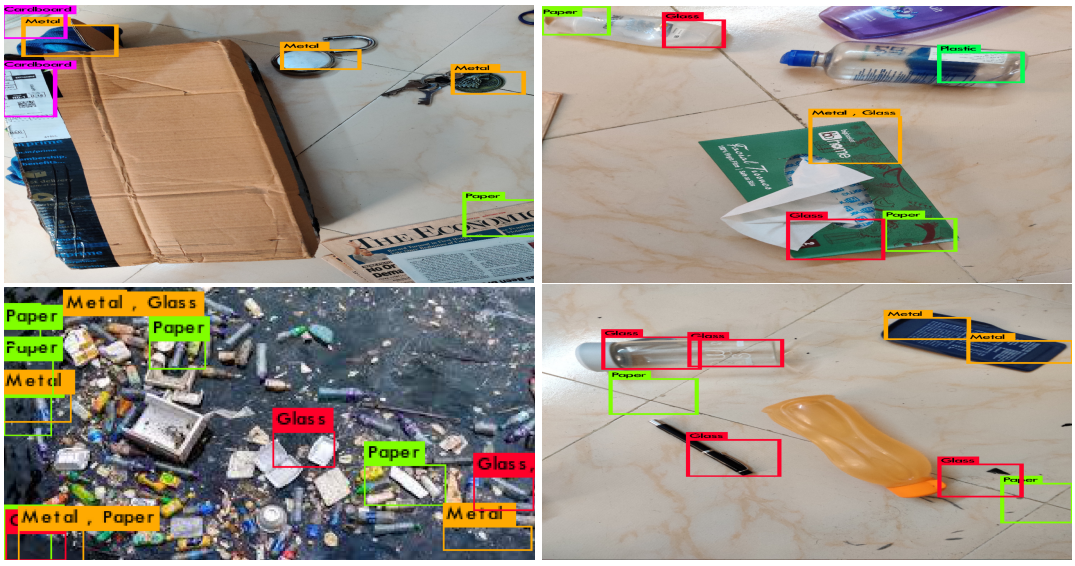


Figure 4.21: Detections for multiple objects case - Approach 2 (*Generated using Python script*)

boxes being drawn, covering portions of the objects in some cases. The boxes were displaced from their ideal location, however, as can be seen from the precision and recall values. Some cases had no detections, while others had problems due to a dominating background. The images of the test data were also re-scaled to 832x832, to improve their resolution, thus increasing detection accuracy. The detection of multiple objects per image also improved considerably with an MAP of 61.53% (Table 4.6). But the predicted bounding boxes were of small size and were unable to enclose the entire object if it was of a bigger scale than the objects in the original image. This is reflected in the decreased value of IOU at 45.53%. In some cases, two or more boxes were drawn for a large object. For small objects (1~-10% of image size), the predicted bounding box was fairly accurate. It also predicted a muddle of trash quite well. The model still faced problems where it identified areas of the background as paper. It also faces difficulties with watermarks or lettering on cardboard or bottle labels and identifies them as paper. This is mostly due to a large number of training samples in the paper category being newspapers.

The model was able to correctly identify the lock and key as metal objects, even though no such type of object was present in the metal training images, showing the improvement in the detection of classes other than paper. The average precision table (Table 4.5 shows the improvement in each class, especially glass, trash, and plastic. Still, quite a few misclassifications, false positives, and false negatives were observed. This improvement could be due to two reasons - The objects in the randomly tested images being of roughly the same size as the objects in the image or the model being trained for more number of iterations. The loss in the first approach was decreasing; however, it hadn't leveled off completely. In this case, the loss more or less seemed to have reached a point of stability. The model was trained for a smaller number of iterations to reduce over-fitting. However, these results show that the model is far from over-fitting and, in fact, might actually benefit from more training. In the next approach, the training was increased further to eliminate any performance problems due to insufficient training.

4.3.4 Approach 3

In this approach, changes were made to the re-scaling function to include various sizes from original size to $(1/10)^{th}$ the size. Also, this time a variety of backgrounds were added instead of only a white background. This was done to explore the importance of the background on predictions. The original image was re-scaled and also translated in 2D randomly by a maximum of 50% of image dimensions. This approach also included negative images - images in which there was no object present, as part of the dataset. This is necessary for the model to learn to differentiate trash from birds and animals or rocks in the water. These images were scraped from different sources online. Sample images from the final dataset are shown below. All the images were re-scaled to 512x384. The model was trained for 5000 iterations, for around 36 hours. The average loss levelled out to around 0.1847. The loss graph is shown in Figure 4.22.

Figure 4.22: Dataset modified with negative images for Approach 3 (*Generated using Python script and OpenCV packages*)

The results are presented below for both single object (Figure 4.24) and multiple objects per image (Figure 4.25). The MAP and AP tables are also shown (Table 4.7 and Table 4.8 respectively).

The detection accuracy improved, with an MAP of 57.80%. The IOU value further decreased, while the F1 score did not register much change. In fact, classes like cardboard and paper showed a decrease in average precision. The detected boxes were once again quite large, though not as large as in Approach 1. The identification of single objects was decent enough, but the multiple object detection was not up to the mark. Small boxes were being drawn but only in very few cases. Most of the time, a large box encompassing all the trash was being drawn. One significant improvement was the reduction in the labelling of the background as objects. This could be attributed to either

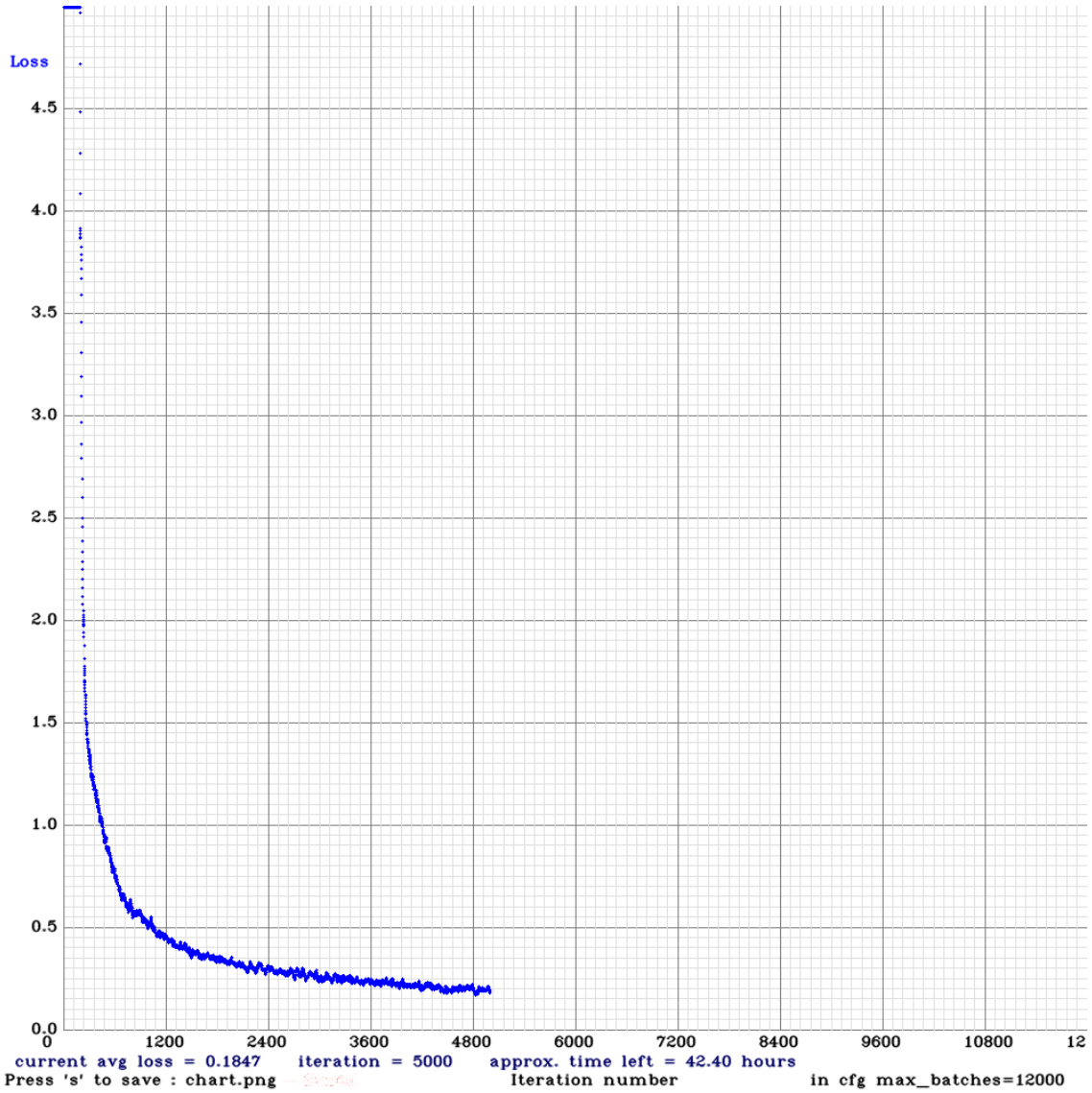


Figure 4.23: Loss chart for 5000 iterations (*Generated using Python script*)

Table 4.7: YOLOv3 results per class - Approach 3

Class	**Average precision (%)**	**TP**	**FP**
Cardboard	39.06	242	594
Glass	73.37	338	114
Metal	63.94	259	157
Paper	62.86	410	435
Plastic	60.17	247	90
Trash	47.38	78	112

Table 4.8: YOLOv3 overall results - Approach 3

Model Parameters	**Value**
MAP	57.80%
IOU	37.99%
Precision	0.51
Recall	0.62
F1 Score	0.56
Detection time	0.06s

Figure 4.24: Detections for single object case - Approach 3 (*Generated using Python script*)

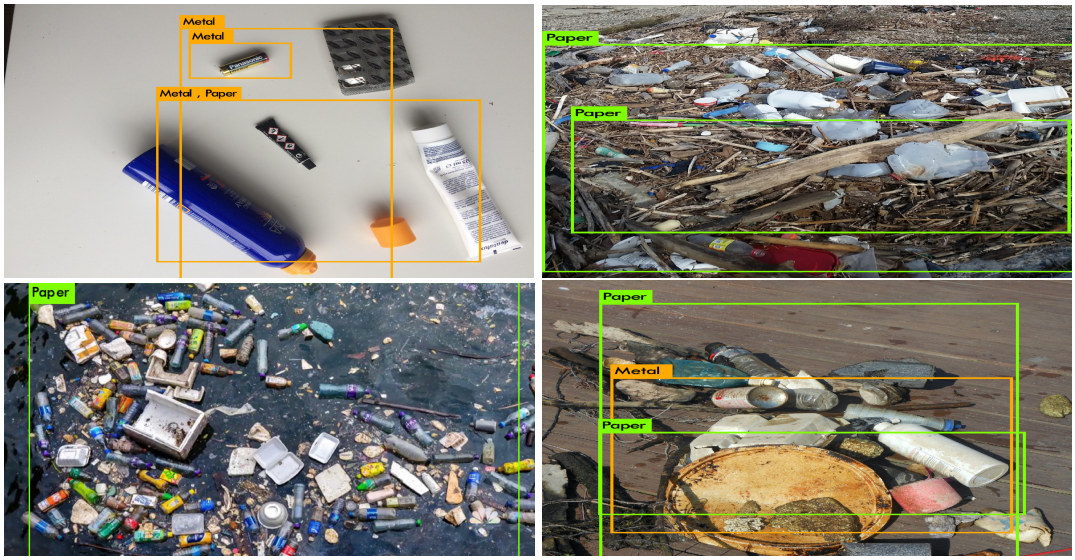


Figure 4.25: Detections for multiple objects case - Approach 3 (*Generated using Python script*)

the introduction of backgrounds or the introduction of negative images. The introduction of a large number of size ranges, however, did not produce the expected result. The boxes were still huge and not precise. Reducing the re-scaling factor also was not very successful. This led to the exploration of slightly modified versions of YOLO, detailed in Section 4.3.5.

4.3.5 Approach 4

In the next attempt, the re-scaling parameters, as well as the model itself were modified. In order to speed up training, Tiny-YOLO, the faster version of YOLO, containing 9 convolutional layers instead of 24[27] was used. This model was trained for the dataset of the Approach 3, to check the drop in accuracy. The accuracy of predictions was found to be almost on par with YOLO. This led to the adoption of Tiny-YOLO for the new data. Tiny-YOLO was trained for both the data with the backgrounds and for the data with an only white background. Total training time was 7 hours. The final average loss decreased to 0.1686 (Figure 4.26), with a MAP of 81%. The results for the data with only white backgrounds were better. These results are shown in Figure 4.27 for single object images and in Figure 4.28 for multiple object images.

Table 4.9: Tiny-YOLO results per class - White background images dataset

Class	**Average precision (%)**	**TP**	**FP**
Cardboard	86.12	66	14
Glass	84.58	85	43
Metal	87.39	66	9
Paper	90.42	100	20
Plastic	69.92	69	38
Trash	66.16	20	18

Table 4.10: Tiny-YOLO overall results - White background images dataset

Model Parameters	**Value**
MAP	80.76%
IOU	58.77%
Precision	0.74
Recall	0.80
F1 Score	0.77
Detection time	0.03s

Table 4.11: Tiny-YOLO results per class - Multiple background images dataset

Class	**Average precision (%)**	**TP**	**FP**
Cardboard	19.29	33	16
Glass	37.11	112	91
Metal	37.04	81	38
Paper	48.07	148	39
Plastic	40.20	111	54
Trash	34.50	33	25

Table 4.12: Tiny-YOLO overall results - Multiple background images dataset

Model Parameters	Value
MAP	36.04%
IOU	49.47%
Precision	0.66
Recall	0.20
F1 Score	0.31
Detection time	0.03s

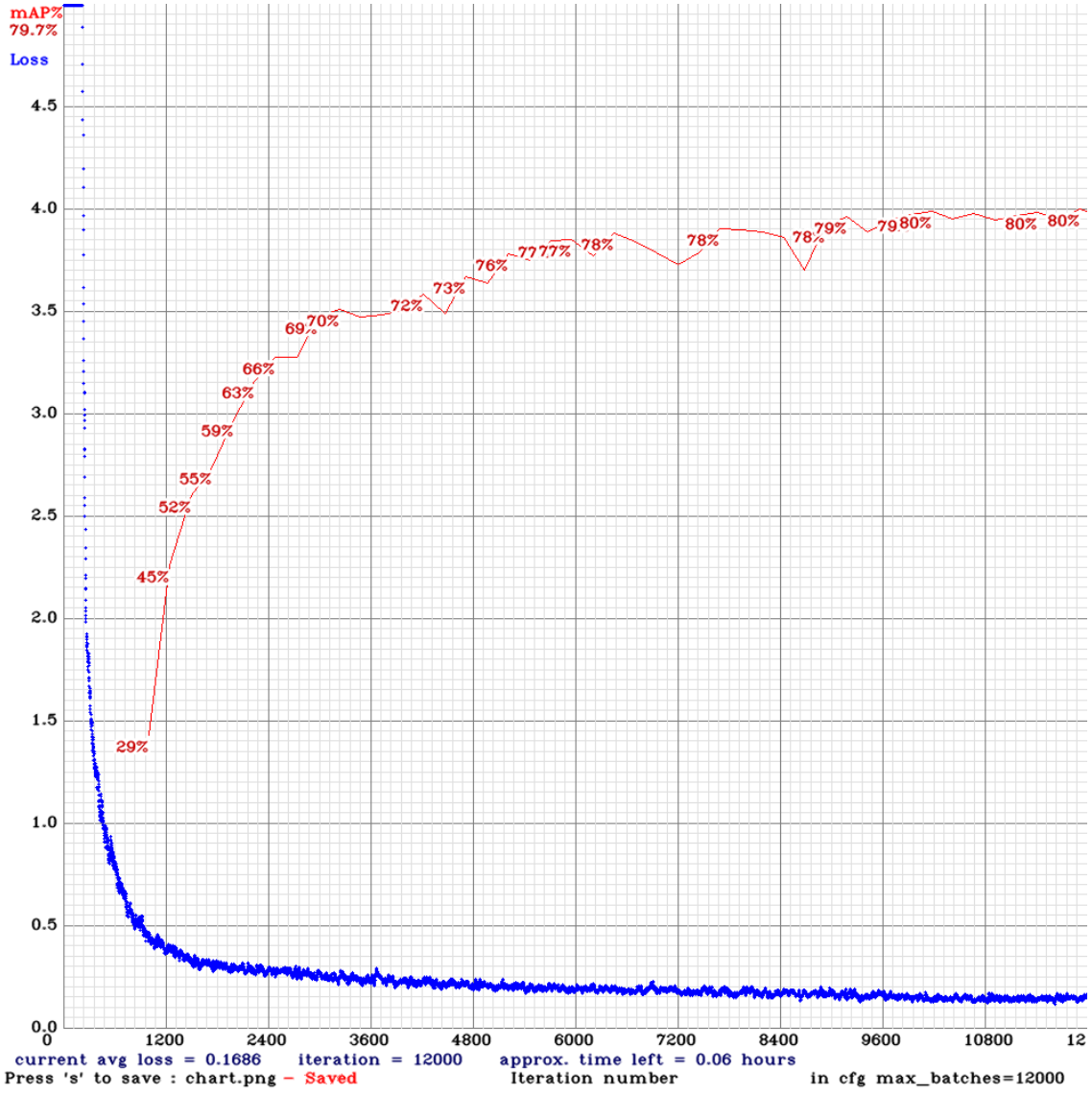


Figure 4.26: Loss Graph with MAP Calculation for 12000 iterations for Tiny-YOLO (*Generated using Python script*)

Figure 4.27: Detections for single object case - Approach 4 (*Generated using Python script*)

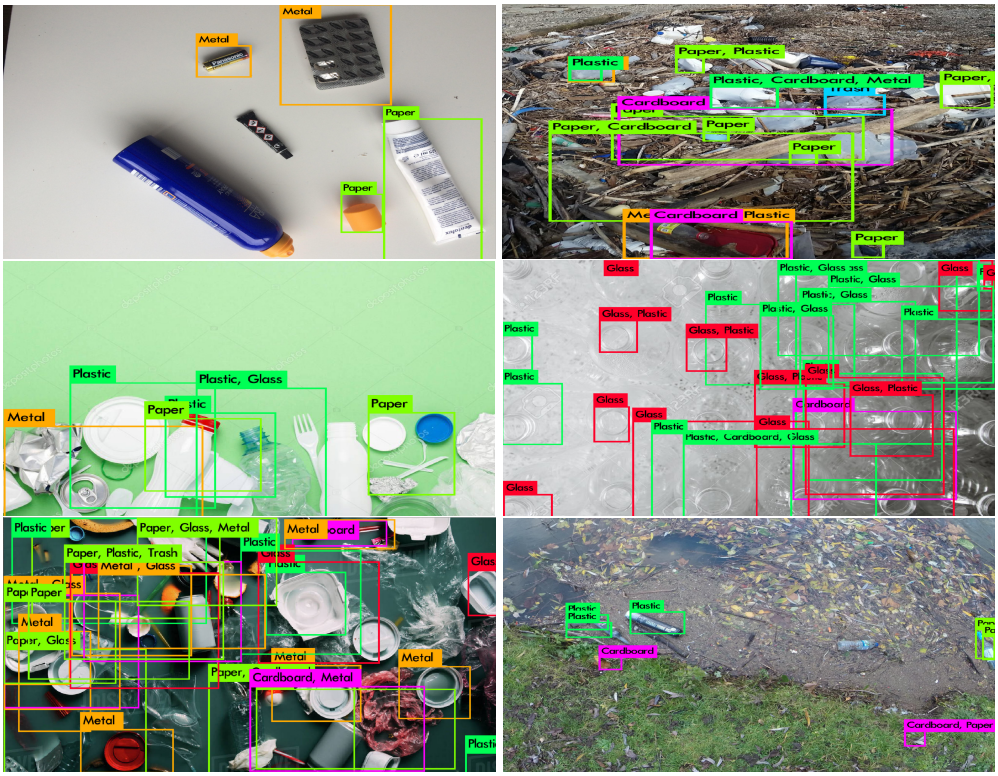


Figure 4.28: Detections for multiple objects case - Approach 4 (*Generated using Python script*)

Of the two types of data used for training, the model performed better on training with images resized and pasted on a white background rather than on different backgrounds. This could be attributed to the higher training required for images with different background, which Tiny-YOLO would not be able to perform. This model showed an MAP of 80.76% (Table 4.10) compared to the other at 36.04% (Table 4.12). The IOU values are also higher for the case with the white background, at 58.77% as compared to 49.47%. Though this is not a very high value, the results are satisfactory as can be seen in the detection images. The F1 score for the white background case is 0.77 with very good precision and recall values. The average precision of all the classes has improved significantly with paper and metal being detected very accurately (Table 4.9). The speed of detection was 30 ms per image, improving on the earlier 60 ms per image. The average precision for the different background case is tabulated in Table 4.11. The detections on the different backgrounds with white background image training were also very good, with an MAP of 76%. This gives the conclusion that training the model on images with a plain white background was sufficient for detection, showing the Tiny-YOLO model's generalisability. Negative images were still used, however. The single object detection shows the bounding boxes covering the object to a good portion and also were of the correct size. The model also could identify a single piece of trash even when it was occluded partially by other debris such as leaves and weeds. Though some inaccuracies still exist, the model is accurate enough, and the coordinate transformation algorithm can make adjustments to fine-tune this. Multiple object detection, too, was satisfactory, with a muddle of objects also being marked individually to a good extent. Though not really required, as the individual coordinates would be very close, this result could be of great use in future modifications. Tiny-YOLO's resolving power can be seen in the examples of detections shown. The average loss also leveled out to around 0.17 after 12000 iterations taking 7 hours. The MAP was also calculated at regular intervals and leveled out at about 80% after 7000 iterations. The model was checked with all the weights saved at each interval to identify any over-fitting.

Finally, Tiny-YOLO and Approach 4 were chosen for object detection.

```
 CUDNN_HALF=1
net.optimized_memory = 0
mini_batch = 1, batch = 64, time_steps = 1, train = 0

 seen 64, trained: 128 K-images (2 Kilo-batches_64)
Enter Image Path: data/obj/cardboard389.jpg: Predicted in 51.448000 milli-seconds.
Cardboard: 46%  (left_x:  217   top_y:  165   width:   68   height:   41)
Enter Image Path: data/obj/cardboard391.jpg: Predicted in 51.789000 milli-seconds.
Cardboard: 46%  (left_x:  211   top_y:  170   width:   82   height:   49)
Enter Image Path: data/obj/cardboard390.jpg: Predicted in 49.817000 milli-seconds.
Cardboard: 53%  (left_x:  204   top_y:  168   width:   98   height:   52)
Enter Image Path: data/obj/cardboard392.jpg: Predicted in 49.854000 milli-seconds.
Enter Image Path: data/obj/cardboard396.jpg: Predicted in 48.952000 milli-seconds.
Cardboard: 51%  (left_x:  225   top_y:  168   width:   70   height:   51)
Enter Image Path: data/obj/cardboard394.jpg: Predicted in 48.497000 milli-seconds.
Cardboard: 63%  (left_x:  210   top_y:  171   width:   83   height:   49)
Enter Image Path: data/obj/cardboard393.jpg: Predicted in 48.288000 milli-seconds.
Cardboard: 37%  (left_x:  222   top_y:  163   width:   78   height:   61)
Enter Image Path: data/obj/cardboard395.jpg: Predicted in 48.535000 milli-seconds.
Enter Image Path: data/obj/cardboard397.jpg: Predicted in 49.634000 milli-seconds.
Cardboard: 86%  (left_x:  214   top_y:  168   width:   72   height:   51)
Paper: 51%      (left_x:  214   top_y:  168   width:   72   height:   51)
Enter Image Path: data/obj/cardboard4.jpg: Predicted in 48.306000 milli-seconds.
Cardboard: 96%  (left_x:  213   top_y:  171   width:   78   height:   45)
Enter Image Path: data/obj/cardboard398.jpg: Predicted in 48.367000 milli-seconds.
Cardboard: 95%  (left_x:  212   top_y:  169   width:   81   height:   50)
Enter Image Path: data/obj/cardboard399.jpg: Predicted in 48.035000 milli-seconds.
Cardboard: 74%  (left_x:  236   top_y:  171   width:   62   height:   49)
Enter Image Path: data/obj/cardboard40.jpg: Predicted in 48.632000 milli-seconds.
Cardboard: 60%  (left_x:  215   top_y:  171   width:   67   height:   45)
```

Figure 4.29: Part of the detected results (*Generated using Python script*)

4.3.6 Other models

A modified version of YOLO that can detect both large and small objects in images exists. This model was also trained for 3000 iterations. The results were not as good as expected. YOLO9000 is an object detecting model that has been trained to identify 9000 different classes of objects[26]. This model was also tested on the current dataset. It failed to identify trash and instead labeled them as artifacts and items. Hence, the only way to do trash identification and detection in optimum amount of time with minimal training is by transfer learning.

Object detected was thus performed using Tiny-YOLO with good accuracy and made suitable for real-time implementation. With high-resolution images, the model shows good accuracy. The next step is to use these detections shown in Figure 4.29, to generate way-points, which is explained in the next Section 4.4. The detections are given in pixel coordinates, which need to be changed to absolute coordinates of latitude and longitude, for GPS control.

4.4 Coordinate Calculation

This section explains the details of the calculation of the absolute postion of the litter. The module flow chart is shown in Figure 4.30.

The drone camera records an HD video of the lake surface. The GPS sensor records

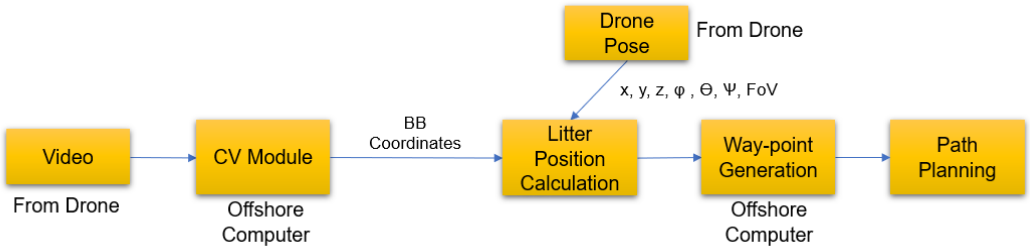


Figure 4.30: Flow Chart for the detection module (*Generated in MS PowerPoint*)

the GPS coordinates of the drone at each frame of the video. Orientation with respect to the magnetic north pole is recorded by the compass. The drone is flown at a constant altitude and is flat with respect to the horizontal (x-y plane). This ensures that the only rotation is around the yaw axis. This can be achieved easily with the altitude hold flight mode in the Pixhawk flight controller. The recorded video is then processed in the PC to extract frames at any rate required. Consider a frame extraction rate of one frame every two seconds. This can be done using many commercially available software, such as VLC Media Player. Once the images are extracted, they are sent to the object detection module, which then identifies the location of trash with respect to the top right corner of the image. It gives the size and position of the top right corner of each object detected, for every image as a text file. Cases of no object or no detection are left blank. This text file (Figure 4.29) is then processed through the algorithm and combined with the latitude, longitude and orientation data from the GPS and compass sensors. A python script was written, that takes this data as input and calculates the latitude and longitude of the trash (See Appendix). The size of the trash is not accounted for here. These coordinates are then made into way-points and provided as input to the navigation algorithm that determines the distance optimizing path to be followed, for the boat.

The algorithm also requires knowledge of camera installation angle and field of view values, that are determined from the gimbal positioning and camera specifications. The area of the lake the camera focuses on is assumed to be a rectangle as shown in Figure 4.31. The field of view in both horizontal and vertical directions can be adjusted in the script, according to camera specifications.

The first step would be to find the latitude and longitude of the top left corner of the box on the surface of the water. This is done in the following way.

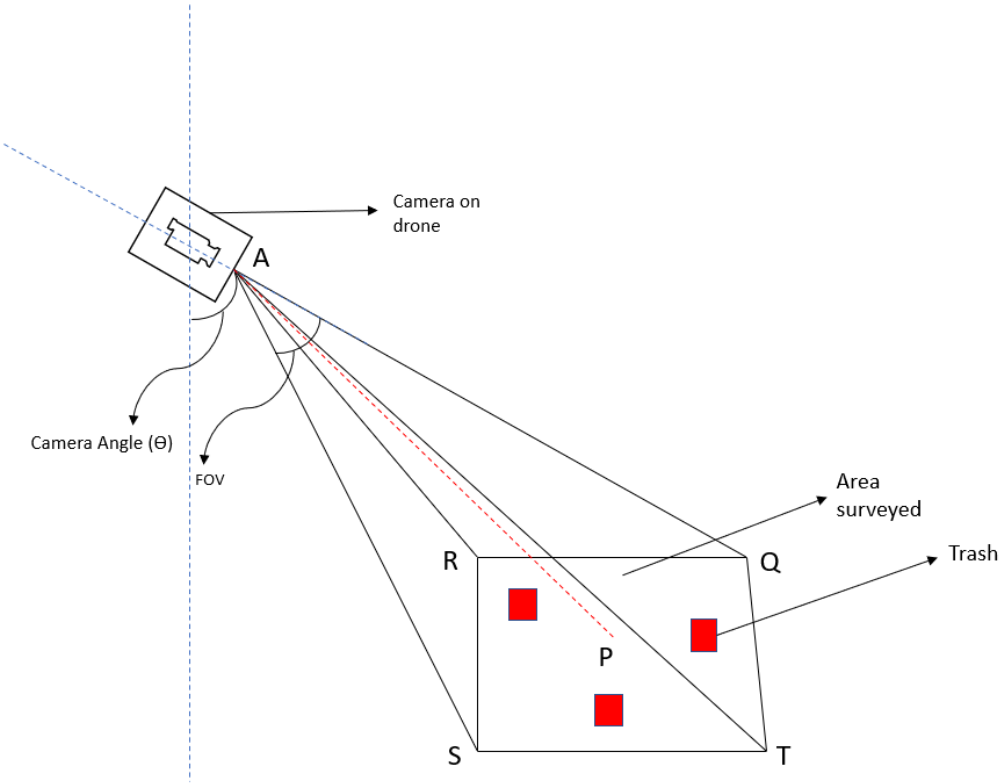


Figure 4.31: Illustration of camera's projection on the water surface (*Generated in MS PowerPoint*)

Let the orientation of the drone be θ with the magnetic north. Let the camera make an angle of ω with the horizontal plane. The angle made with the vertical will be $90-\omega$, which is denoted by β. Let the drone's altitude be h and the GPS coordinates of the drone be lat and $long$. The camera's horizontal and vertical fields of view are given by fov_v and fov_h Let the camera's line of sight hit the water surface at a point P. The

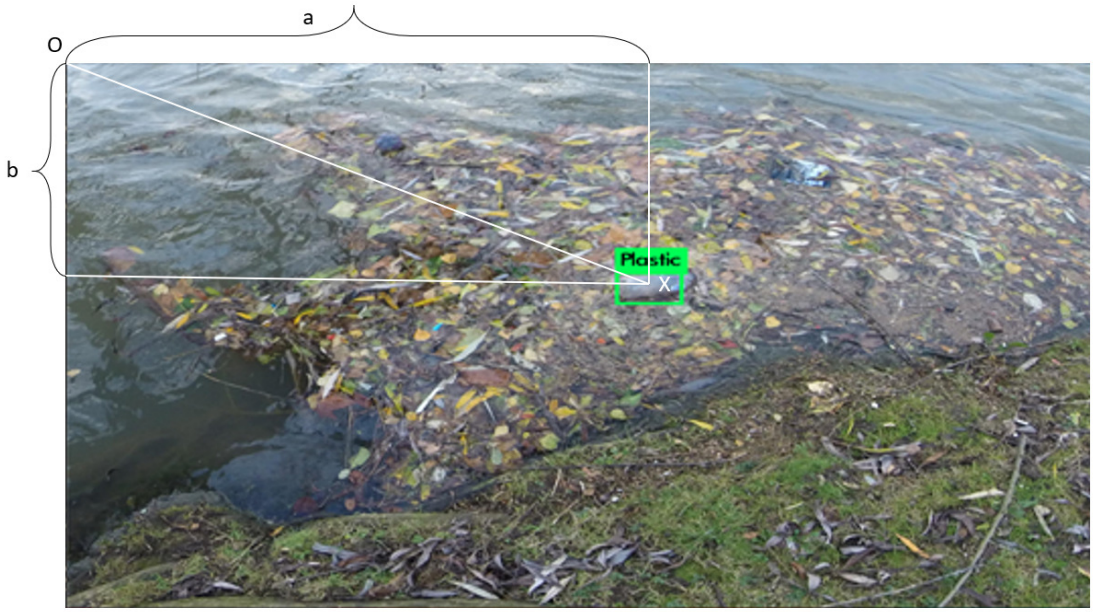


Figure 4.32: Parameters for the detected image (*Generated in MS PowerPoint*)

latitude and longitude of P are given by the equations :

$$P_{lat} = lat + h \cdot \tan\beta \cdot \cos\theta \tag{4.15}$$

$$P_{long} = long + h \cdot \tan\beta \cdot \cos\theta \tag{4.16}$$

The length of this line of sight (AP) is given by $\frac{h}{\sin\beta}$

$$Let\ \gamma = \frac{\pi}{2} - \beta - \frac{fov_v}{2} \tag{4.17}$$

The latitude and longitude of the point Q are required, as the image detections are performed by assuming the origin at the top left corner of the image. The coordinates of Q are given by :

$$Q_{Lat} = P_{Lat} + \left(\frac{AP}{\sin\gamma}\right) \cdot \sin\frac{fov_v}{2} \cdot \cos\theta + AP \cdot \tan\frac{fov_h}{2} \cdot \sin\theta \tag{4.18}$$

$$Q_{Long} = P_{Long} + \left(\frac{AP}{\sin\gamma}\right) \cdot \sin\frac{fov_v}{2} \cdot \sin\theta + AP \cdot \tan\frac{fov_h}{2} \cdot \cos\theta \tag{4.19}$$

Define $\alpha = \beta + \frac{\pi}{2} - \frac{fov_v}{2}$

The length QR and RS, which are required for scaling, are given by :

$$QR = AP \cdot \sin \frac{fov_v}{2} \cdot \left(\frac{1}{\sin \alpha} + \frac{1}{\sin \gamma} \right) \tag{4.20}$$

$$RS = 2AP \cdot \tan \frac{fov_h}{2} \tag{4.21}$$

Now, let the coordinates of point X, where the trash is identified in the image, be (a, b) with respect to the origin O, in image coordinates, as shown in Figure 4.32. These coordinates have to be scaled and changed into latitude and longitude, now that the latitude and longitude of the origin O are known (They are the same as the point Q). The distance OX is given by :

$$OX = \sqrt{\left(\frac{a \cdot RS}{512} \right)^2 + \left(\frac{b \cdot QR}{384} \right)^2} \tag{4.22}$$

Define

$$\eta = \arctan \left(\frac{\left(\frac{b \cdot QR}{384} \right)}{\left(\frac{a \cdot RS}{512} \right)} \right) \tag{4.23}$$

Simplifying further, let $\mu = \frac{\pi}{2} - \eta - \theta$

Finally, the object (trash, denoted here by X) coordinates (Latitude and Longitude) are given by :

$$X_{Lat} = Q_{Lat} - OX \cdot \cos \mu \tag{4.24}$$

$$X_{Long} = Q_{Long} + OX \cdot \sin \mu \tag{4.25}$$

These values give the latitude and longitude of the trash identified. These coordinates are exported as a CSV file to the navigation module, which then generates waypoints and finds the optimal path for trash collection. These detected coordinates for

20 points taken on the surface of the IITM lake are shown in Table 4.13 and plotted in Figure 4.33.

Table 4.13: Latitude and Longitude values of identified litter on IITM lake

Node	**Latitude**	**Longitude**	**Node**	**Latitude**	**Longitude**
A	12.9937	80.23937	K	12.99313	80.24
B	12.99396	80.23888	L	12.99324	80.2399
C	12.99392	80.23924	M	12.99326	80.2398
D	12.99386	80.23948	N	12.99324	80.23956
E	12.99375	80.23972	O	12.99342	80.23961
F	12.9936	80.2399	P	12.99348	80.23948
G	12.9935	80.23991	Q	12.99347	80.23935
H	12.99342	80.24009	R	12.99357	80.23912
I	12.99329	80.24026	S	12.99365	80.23901
J	12.99317	80.24014	T	12.99381	80.23896

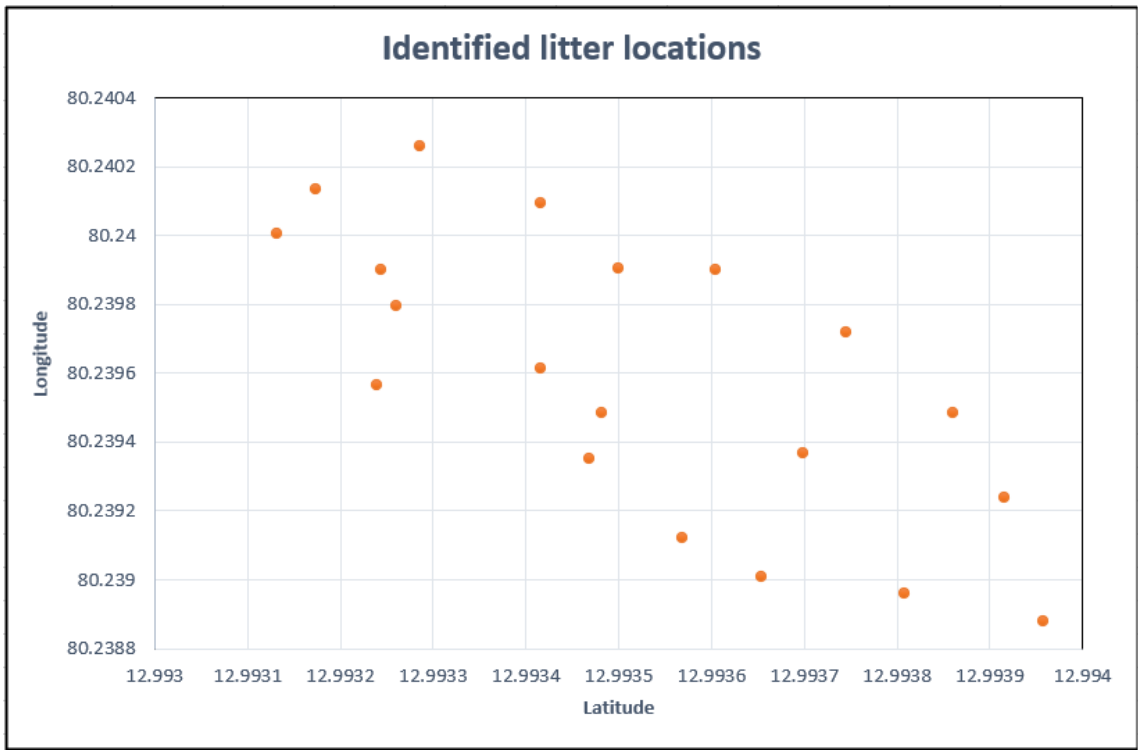


Figure 4.33: Locations of 20 points on the IITM lake (*Plotted in MS Excel*)

Chapter 5

AERIAL ROBOT MODULE

5.1 A Theoretical Approach to Quadcopters

A quadcopter is classified as a multirotor, as it derives its lift from rotors, rather than wings. The first quadcopters were developed in the 1920s, and were the first heavier than air vehicles capable of vertical take-off and landing. They however suffered from many problems, including reduced control and lack of stability. Advances in electronics led to the development of better flight controllers and more reliable sensors that increased the popularity of quadcopters for small indoor and outdoor flights. They were seen as convenient and easy to build and fly for small tasks. In recent years, quadcopters have seen increasing usage in a number of areas such as military, journalism, photography and surveillance, drone-delivery, agriculture, and also disaster relief. The versatility of quadcopters, due to their small size and ability to perform skilful manoeuvres has increased their usage in many of these areas. Current drones also incorporate high-level features such as object tracking, person following, and obstacle avoidance. In this chapter, a brief outline of the mathematics behind quadcopters is provided first, followed by an explanation of the components required to assemble and fly a drone.

5.1.1 A Mathematical Introduction

Figure 5.1 shows a schematic diagram of the quadcopter, with the forces and moments generated by the motors. The dynamic model for the quadcopter is given as follows.

Each motor provides an upward force proportional to the square of the angular velocity. Hence, the total upward force generated is given by :

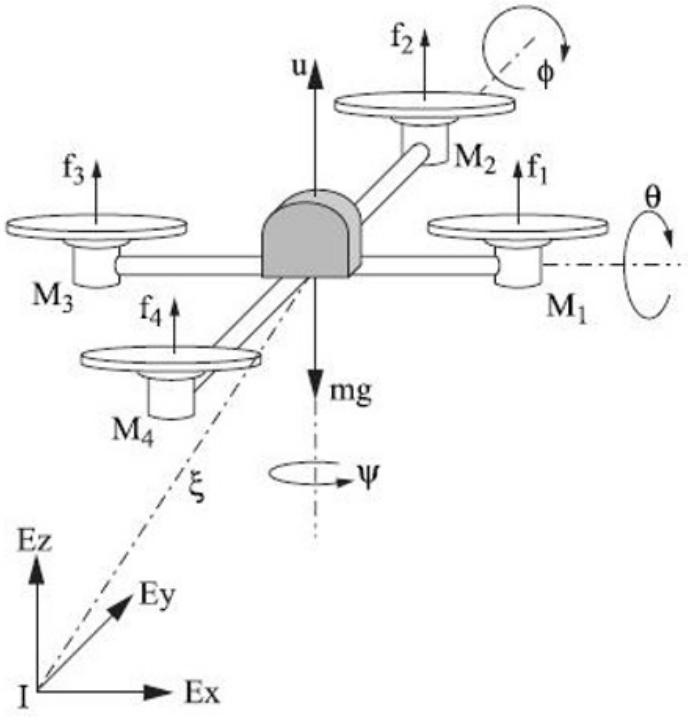


Figure 5.1: Forces on quadrotor

Source: *Aerial Robotics by Lozano [21]*

$$f = \Sigma f_i; \tag{5.1}$$

$$f_i = k_i + \omega_i^2 \tag{5.2}$$

Therefore, the force on the quadcopter is given by :

$$F = \begin{pmatrix} 0 \\ 0 \\ f \end{pmatrix}$$

Similarly, the torque matrix is given by :

$$\tau = \begin{pmatrix} \tau_\psi \\ \tau_\theta \\ \tau_\phi \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{4} \tau_{Mi} \\ (f_2 - f_4)l \\ (f_3 - f_1)l \end{pmatrix}$$

where l is the distance between the motors and the centre of gravity and τ_{Mi} is the

moment produced by motor M_i and ψ, θ and ϕ are roll, pitch and yaw respectively.

Solving the Lagrangian to get the equation of motion, the dynamical model of the quadcopter is obtained as :

$$\text{F} = \begin{bmatrix} m\ddot{x} \\ m\ddot{y} \\ m\ddot{z} - mg \end{bmatrix}$$

The torque equation is given by :

$$\tau = J\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} \tag{5.3}$$

where J is the inertia with respect to the fixed frame and η is the angular velocity with respect to the fixed frame.

5.2 Drone Building

Building and assembling drone requires a basic knowledge of aerodynamics, control theory, propulsion and sensors. This section explains the different components of the drone, their functions, their selection process and the final assembly[7]. A small introduction to drone basics is also provided. There are many types of drones, such as quadcopter, hexacopter, ready-to-fly (RTF) drones, etc. The quadcopter model was chosen for its simplicity and versatility. This section is divided into subsections detailing each function of the drone and the parts that are needed to achieve it.

5.2.1 Structure

The components constituting the structure of the drone are a frame, landing gear, shell, and propeller guards. Rubber props are also included to minimize vibration. The shell is an outer covering for the electronic components, which was not purchased, keeping in mind the budget and drone weight considerations. Also, propeller guards and rubber

mounts were add-ons that could be added later and were not really necessary for basic drone flight. Choosing a frame is one of the most crucial aspects of drone building and involves a lot of variables, discussed in Section 5.2.1.

Frame

The frame provides the base for the drone. It is the skeleton which supports and makes provisions for connecting the other parts of the drone. There are different types of frames for different drones, such as a quadcopter, hexacopter or octacopter. Every frame has consists of a central base and either 4, 6 or 8 arms depending on whether the multi-rotor is a quadcopter, a hexacopter, or an octacopter.

The frame has an upper plate and a lower plate which enclose a small space in between them to place the electronics. The battery is generally fixed on the underside of the lower plate of the frame, to reduce the height of the center of gravity. The motors and propellers are fixed on the ends of the arms, with the ESCs below the arms. The arms are placed in either a '+' or an 'X' configuration. The quadcopter frame is one of the simplest to assemble and manoeuvre. Most of the commercially available flight controllers are compatible with this type of frame, making it an ideal choice for the current work. Frame materials range from plastic or wood, which are cheap and easily available, to aluminium, Fiber-Glass and Carbon Fiber, which are more expensive. Plastic and wood have lesser strength and are liable to break during crashes, etc. Carbon fiber, on the other hand, is extremely sturdy and also lightweight, providing bulk and minimizing energy costs simultaneously. Some variants use aluminium to build the entire frame or as a supplement to carbon fiber.

A variety of frames are commercially available for different types of drones. For hexacopters, S550 and F550 are the most popular choices of 550mm frames. The F series frames are a product of DJI. For quadcopters, the F450 is a popular choice. The Q450 is a cheaper version of F450, which is recommended for beginners, building a quadcopter for the first time. These frames have an end-to-end distance of 450mm. Larger frames such as Tarot 680Pro that can support bigger propellers and also can fold

back for transportation are used in professional drone flying. They can carry bulky payloads. Small FPV racing frames supporting 5in propellers are available for drones used for acrobatic purposes, such as racing, stunt performances, and so on. These drones do not carry any payload. The general idea is that smaller frames have excellent flexibility and agility, while larger frames have better payload carrying capacity.

Figure 5.2: Drone Frame - F450

Source: *robu.in*

Frame weight and size are critical parameters influencing drone building. The size of the frame determines the size of the propellers that can be installed, avoiding overlap and interference. The propeller size determines the amount of thrust produced, which must be more than the take-off weight of the drone. Minimizing the weight of the frame is essential as the other components such as the battery and the motors are quite bulky. Increasing the frame size will increase the weight, requiring larger propellers that will then need lower KV motors to drive them. This increases the overall cost. Too small a frame will be unable to lift any payload. The payload, in the current scenario will only be a camera with its gimbal, weighing a maximum of 60g. Consequently, a very powerful or large frame was unnecessary. The F450 frame (Figure 5.2) was chosen

considering the optimal size of 450mm, lower cost, and excellent compatibility.

Landing Gear

The landing gear is a necessity to achieve a smooth landing and to protect crucial drone components from damage. The landing gear is also usually made of plastic, though other materials are also used. It is generally sold along with the frame.

There are many types of landing gear - fixed, retractable, shock-absorbing, non-shock absorbing, etc (Figure 5.3). The main issue with using the landing gear is the drag it generates. As the speed of an aircraft increases, the drag acting upon it also increases. Mechanisms to retract and stow the landing gear to eliminate drag add to the overall weight of the aircraft. On slow aircraft, the penalty of this added weight is not overcome by the reduction of drag, allowing the use of fixed landing gear. As the speed of the aircraft increases, the drag caused by the landing gear also increases and the retracting mechanism is necessary, despite the additional weight. Retractable landing gear stows into the fuselage while in flight. Other aircraft have separate doors that open and close, allowing the gear to enter or leave. The landing gear absorbs impacts in two ways - by transmitting the shock wave at a slower rate to the aircraft body or by absorbing it completely upon impact.

For the current work, a fixed type of plastic landing gear, was preferred, as the drone speed is not very high. The gear was supplied along with the frame.

5.2.2 Propulsion

This section explains the different components that make up the propulsion system of the drone. These are the motors, propellers, ESCs, the Power Distribution Board and the battery.

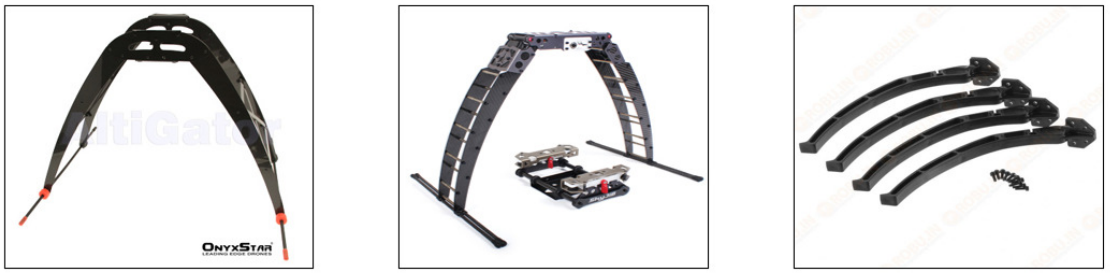


Figure 5.3: Types of Landing Gear - Fixed and Retractable

Source: *flight-mechanic.com*

Motors

The electric motor is one of the most crucial components of the drone. Along with the propellers, motors provide lift and thrust. Motor selection has an enormous influence on the flight characteristics of the multi-copter. Small differences in the specifications of a motor result in significant changes regarding the weight, responsiveness, and total power of the drone. Motors come in many types and sizes. To choose a motor, one must first know the approximate weight of the drone, as the thrust developed by all the motors combined must be larger than the drone weight by a good margin. This thrust is in part, dependent upon the propeller size too, which is determined by the frame size. The general rule for the Thrust-to-Weight ratio is that it should be at least 2, i.e., the thrust generated by the motors at full throttle should be at least twice the total take-off weight of the drone (this includes any payload carried). Too little thrust will result in poor control and difficulty during take-off. Having a higher Thrust-to-Weight ratio will also give additional space to include more payload.

Typically, electric motors are of two types - Brushed or Brushless. Brushed motors have lesser electronics, can operate in harsher environments and have a low cost of construction as compared to their counterparts. Brushed motors also do not need a controller, for fixed motor speeds. However, brushed motors suffer from issues such as low speed, over-heating due to poor heat dissipation, and brush wear out, causing motor damage. Brushless motors generate high speeds and more power. They are also more efficient and require less maintenance, increasing their life. They however are more expensive and require a speed controller for operation. Brushless DC motors were used

for the current work. Brushless motors are typically described by a 4-digit number, where the first two digits denote stator width and the last two digits denote stator height.

In general, the larger the motor, the more torque it produces. One important area to pay attention to is the KV rating of the motors. The KV rating theoretically describes how fast the motor spins when the voltage increases by 1V without load. For example, a motor rated 900KV, running with a supply voltage of 5V, will spin at 900 x 5 = 4500 rpm, approximately. Mounting the propellers decreases the rpm of the motors, and generates thrust. Higher KV motors turn the propellers faster, but generate lesser torque.Lower KV motors generate higher torque with slower motor speeds. Bigger propellers are matched with low KV motors, and smaller propellers with high KV motors. Matching high KV motors with large propellers will draw excessive current, causing over-heating and damage to the motor and other components of the drone. Hence, it is necessary to find an optimum size and rating for the motor. All motors also have a framework number like 12N14P, where 12 gives the number of electromagnets in the stator and 14 gives the number of perpetual magnets in the motor. This number is the same for most motors, except for extremely small motors that have more electromagnets to increase the torque generated. Table 5.1 shows the frame size to motor size relation that is generally suggested.

Table 5.1: Motor Size Chart (*Adapted from 'Multirotor Motor Guide' [29]*)

Frame Size (mm)	**Propeller Size (in)**	**Motor Size**	**KV**
<150	<3	<1306	>3000
180	4	1806	2600
210	5	2204-2206	2300-2600
250	6	2204-2208	2000-2300
350	7	2208	1600
450	8,9,10	>2212	<1000

For the drone, four EMAX MT2213 935KV Brushless DC Motors with 12N14P framework were purchased (Figure 5.4). They are compatible with Li-Po 3S and 4S

batteries and generate a maximum thrust of 860g/motor, drawing a maximum current of 18A. 10in propellers are suggested for this model. Higher thrust gives better speed, but also utilises a lot of power. The battery must have a sufficient discharge rate (C) to supply the given power. This affects the time of flight of the drone.

Figure 5.4: EMAX MT2213 935KV Brushless DC Motor

Source: *robu.in/product/emax-mt2213-935kv-bldc-motor-ccw-prop1045-combooriginal/*

The direction of rotation of the motors is also an important factor to take into consideration, as a combination of clockwise and anti-clockwise propeller spins is required to maintain drone orientation. Pixhawk recommends the configuration of motor spins for a quadcopter, as shown in Figure 5.5.

Electronic Speed Controller (ESC)

The Electronic Speed Controller is a key component of the propulsion system, facilitating the control of motor speeds by the flight controller. The ESC regulates and provides the correct voltage required, and also handles the maximum current the motors can draw. ESCs consist of a servo connector to accept the input signal from the flight controller, three bullet connectors to connect to the motor and a power input from the battery. Most ESCs commercially available are controlled by built-in 32-bit processors, running either the BLHeli-32 or the KISS firmware. The usage of 32-bit processors increases not only the speed of communication but also allows the drone to perform more acrobatic manoeuvres. These ESCs are also compatible with telemetry, where

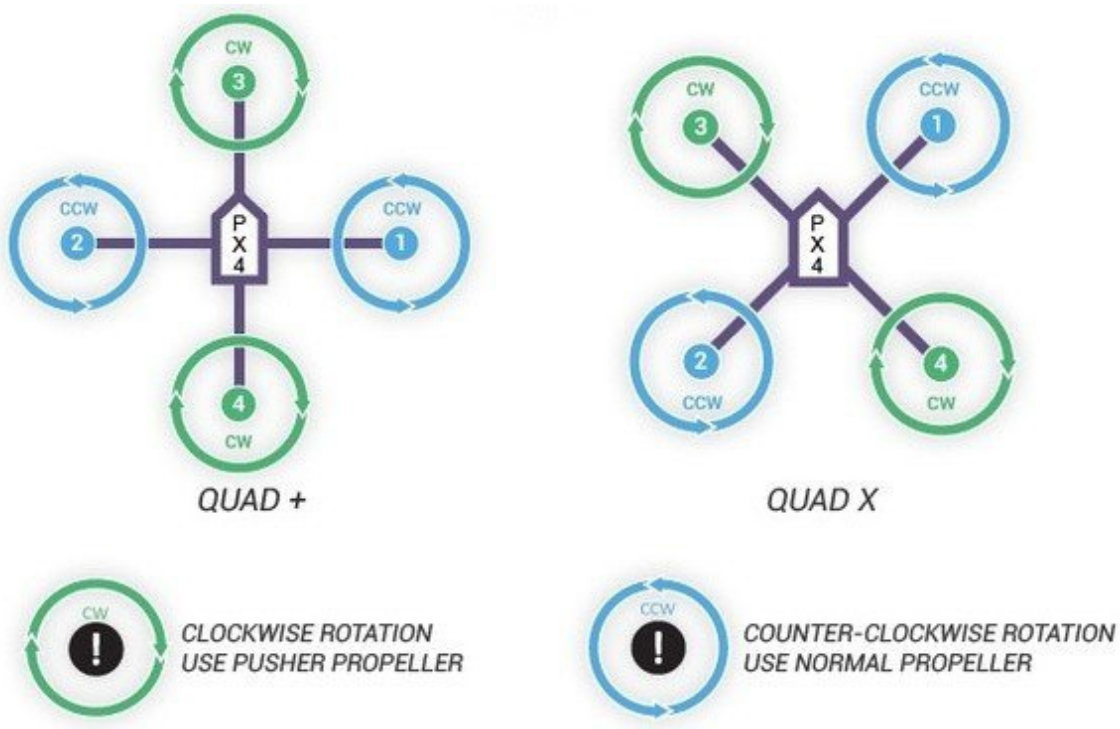


Figure 5.5: Motor Configuration for Quadcopter recommended by Pixhawk

Source: *PX4 Documentation*

signals such as temperature, rpm and current drawn are sent from the ESC to the flight controller, to monitor the health of the system. Each motor requires a separate ESC. Recently, 4-in-1 ESCs are becoming available, which include a PDB board.

ESCs are generally rated by the amount of steady current they can supply to the motors. The ESC provides power to the motor coils at incredibly fast switching rates. This switching is controlled by a microprocessor and carried out by transistors called MOSFETs. ESCs have current ratings such as 25A or 30A that represent the sustained steady current that the ESC can handle. For short periods of time (around 10s), ESCs can also handle a surge current, known as 'burst'. ESCs are also rated on their voltage handling capacity, as 3S or 4S. A higher voltage Li-Po battery will require lower current to operate the motors at the same power. Choosing the ESC again depends on motor and propeller size and battery capacity. The main consideration in selecting an ESC lies in ensuring that the maximum current passing through the circuit can be safely handled.

ESCs receive a PWM signal from the flight controller that dictates the speed of

the motor. Recent ESCs however use faster digital communication protocols such as D-Shot. D-Shot requires the ESC to incorporate a microprocessor and associated firmware, to translate the digital signal into motor rpm commands. The direct ESC-Motor connection facilitates the rapid change in motor speed.

Four Emax BLHeli Series 30A ESCs were purchased for the current project (Figure 5.6). This ESC can handle an input voltage of upto 16V, and a steady current of 30A. A 40A burst current is also handled for 10 seconds. It is suitable for Li-Po batteries ranging from 2S to 4S.

Figure 5.6: Emax BLHeli Series 30A ESC

Source: *https://robu.in/product/emax-blheli-series-30a-esc-oneshot-original/*

Propellers

Along with motors, propellers are the key components that keep the multi-rotor flying. They are also the most fragile of all the components and are damaged frequently. Choosing a propeller requires knowledge of size, pitch, blade configuration, material and efficiency. An optimal combination of all these factors needs to be found.

Propellers are usually described as *Size(inches) x Pitch(inches) x Number of Blades*. The propeller size is determined by the size of the frame. A bigger propeller sweeps through more air and thus requires more power to spin. Consequently, the responsiveness is decreased due to inertia. The larger propeller, however, generates more thrust and adds stability to the drone. These propellers are mostly used for large drones, with payload. Smaller propellers, on the other hand, have increased responsiveness and require lesser power to spin. However, deployed on the same frame, smaller propellers will generate lesser thrust and will also pose difficulties in control. These propellers are used on small, acrobatic drones. A mismatch between motors and propellers can be detrimental to both components.

The pitch is the angle of the propeller blades. It is the amount which the propeller moves in one rotation. A higher pitch gives more thrust and speed, but lesser torque. As a result, it uses more power and is slow to respond to inputs. A smaller pitch will give more torque, though with decreased thrust and low speed. It also reacts quickly to changing inputs and consumes less power. High and low pitch propellers are shown in Figure 5.7.

Figure 5.7: High Pitch and Low Pitch Propellers

Source: getfpv.com

To increase thrust, either the size or the number of blades is increased, at the cost of increasing the total power consumed due to the increased drag from the blades. In-

creasing the size is not always feasible due to frame restrictions and thus, multi-bladed propellers were developed. If the required thrust is not very high, propellers with 2 or 3 blades can be used as they also have increased responsiveness.

Propellers are usually made of fibre-glass reinforced plastic, that has high stiffness. This helps it keep its shape while spinning. However, during a crash, the propeller can break easily. New propellers are being made using polycarbonate plastic that bends rather than breaks during crashes. Propellers are also affected by the climate of the flying area. More sophisticated propellers have advanced features such as small wing-tips, to reduce drag.

The direction of spin of a propeller is an important aspect. Propellers spin clockwise or anti-clockwise. As shown in the motor section, a quadcopter needs two clockwise propellers and two anti-clockwise propellers in the configuration shown (Figure 5.5), to negate the turning moment and prevent toppling. Propellers also come with adapter rings that help in attaching them to the motors if there is a mismatch of shaft diameter. Other add-ons include propeller guards and savers that help mitigate damage during light crashes. For the current work, eight 1045 (10 in length, 4.5 cm pitch) plastic two-bladed propellers were purchased, that are compatible with the motors and frame (Figure 5.8).

Battery

Quadcopters are generally powered by Lithium Polymer (Li-Po) batteries. A few small drones use Lithium Polymer High Voltage (LiHV) batteries. A Li-Po batteries is more reliable and is the better choice. A standard Li-Po cell has a storage voltage of 3.7V. To increase the voltage and power, many such cells are connected in series. The number of cells connected in this fashion is given by the battery configuration, i.e., a 4S battery has 4 Li-Po cells in series. The voltage of the battery pack is important as it affects the maximum speed of the motors.

Figure 5.8: 1045 Propellers (Plastic)

Source: *https://robu.in/product/orange-hd-propellers-114711x4-7-carbon-fiber/*

Increasing battery capacity increases the flight time, but will also increase the weight. The expected flight time for a 2000mAh battery is genrally around 8 minutes.

The amount of steady current that a battery can continuously supply for a given charge cycle is determined by its C-Rating. The higher the C-Rating, the more is the current supplied. The maximum safe current that can be drawn is given by the product of the C-Rating and the Battery Capacity (expressed in mAh). Forcibly drawing higher current can cause battery damage and even explosions in extreme cases. The maximum current drawn is important because the motors and ESCs need to be chosen accordingly and made compatible. The Li-Po battery connects to the PDB with an XT60 connector - a specially designed connector with a safety casing that can withstand high temperatures. For this project, an Orange 5200mAh 3S 40C/80C Lithium polymer battery, was purchased, that can give a flight time of around 20 minutes (Figure 5.9).

Figure 5.9: Orange 5200mAh 3S 40C/80C Li-Po Battery

Source: *https://robu.in/product/orange-5200mah-3s-40c80c-lithium-polymer-battery/*

Power Distribution Module

The power distribution board is a small component, consisting of a printed circuit board that splits the battery power equally to the four ESCs. A PDB simplifies the connections of the drone and reduces hassle while assembling the drone. PDBs also are equipped with voltage regulators. In a few cases, components such as cameras, video transmitters, etc. may require different voltages to function. Such requirements can be accommodated by a PDB. It is also important to select the PDB that can handle the maximum current the battery can supply. Recent developments in microprocessors have facilitated the integration of the PDB into the flight controller directly, to simplify the assembly process even further. A PDB - XT60 with BEC 5V and 12V was purchased for the current application (Figure 5.10). It can give a steady output current at 25A and a 30A burst current for 10 seconds, for each motor. The PDB also uses an XT60 connector to connect to the battery.

Figure 5.10: Power Distribution Module

Source: *robu.in/product/pdb-xt60-w-bec-5v-12v/*

5.2.3 Flight Controller

The flight controller is the 'brain' of the drone, consisting of the 'flight stack' software running on the 'vehicle controller' hardware. The flight stack is a collection of guidance, navigation and control algorithms for autonomous drone flight. It can also control other types of aerial vehicles such as fixed-wing, VTOL, etc. The basic flight-stack is shown in the flow chart in Figure 5.11.

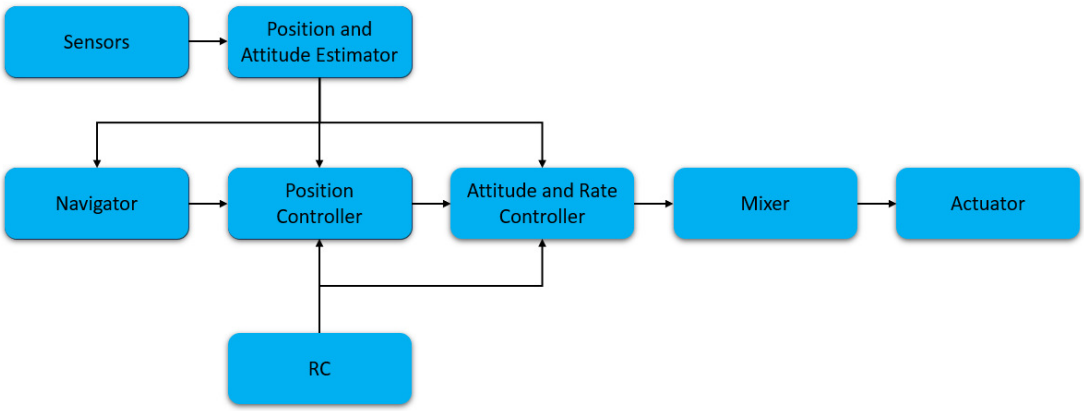


Figure 5.11: Flight Stack Block Diagram (*Adapted from PX4 Documentation*)

The estimator modules take sensor inputs, from single or multiple sensors and compute the vehicle state, i.e., values of all the controllable variables of the vehicle at that instant. For example, the attitude is calculated from the IMU sensor values. Sensor fusion, where data from multiple sensors is combined, is also used to improve accuracy of the computed state.

The controller takes the measured value and the set-point as input and adjusts the process variables such that the measured value matches the set-point. Eventually, the set-point is reached through small changes made to the process variables. For example, the position controller takes position set-points as inputs, the process variable is the currently estimated position, and the output is an attitude and thrust set-point that moves the vehicle towards the desired position.

The mixer translates commands from the controller into motor speed commands. This process is specific for a particular vehicle and depends on factors like motor compatibility, positioning, etc.

Choosing an autopilot is an important decision that significantly affects the working of the quadcopter. The selection of the autopilot hardware directly affects the configuration of the vehicle, the application potential, and the features required. Some of the factors that need to be taken into consideration are :

- The number of outputs required. This determines the number of actuators that can be fixed on the vehicle. It will directly affect the decision of the frame
- The type of communication protocol available, such as the number of UART terminals. This directly determines the number of devices that can be connected to the autopilot. These devices include telemetry modules, GPS systems, RC receiver modules, etc.
- The number of sensors included in the autopilot hardware. Some autopilots also support sensor redundancy to obtain accurate results. For example, many flight controllers contain multiple integrated IMUs

- Size. This generally is not a very significant variable in the case of a generic quadcopter, but matters a lot in the case of a racing drone or a customized vehicle. Such vehicles have limits on the amount of available space or the maximum weight allowed

- The number of analog input-output slots on the autopilot. These provide the means to incorporate analog sensors such as a receiver signal strength monitoring sensor

- The cost of the autopilot. The application area and functionality required determines the cost

The chosen autopilot hardware, Pixhawk 2.4.8 satisfies the requirement for the current project, in terms of the factors mentioned above[22]. It is one of the most frequently recommended autopilots and is compatible with many vehicles ranging from quadcopters to autonomous surface vehicles. Pixhawk has the potential to control and navigate any vehicle using RC or Futuba S-Bus servos.

Pixhawk is an independent open source project that is used in the development of many autonomous vehicles for various purposes ranging from racing and hobby drones to sophisticated industrial-use vehicles. Pixhawk boards are developed in parallel with PX4 and use the NuttX OS. Pixhawk is a well tested and used autopilot that has provides for flexible hardware attachments. Though it can be used with any ground station software, QGroundControl is recommended.

Pixhawk comes with built-in Wi-Fi ESP8266 external, and is compatible with GPS 7M. It has several redundant power inputs and an external safety switch to stop the device in the case of an emergency. Pixhawk also has some inbuilt sensors such as a 16-bit gyroscope, a 14-bit accelerometer, a magnetometer, a 3 axis accelerometer, and an MS5611 barometer sensor. The Pixhawk flight controller is mounted securely on the vehicle, preferably with a vibration damping mechanism to preserve connection integrity. The autopilot is mounted close to the centre of gravity of the system. The white arrow marks the heading direction. The buzzer and safety switch are installed for emer-

gencies. The board has clearly marked inputs named 'buzzer' and 'switch' that need to be connected to the buzzer and safety switch respectively.

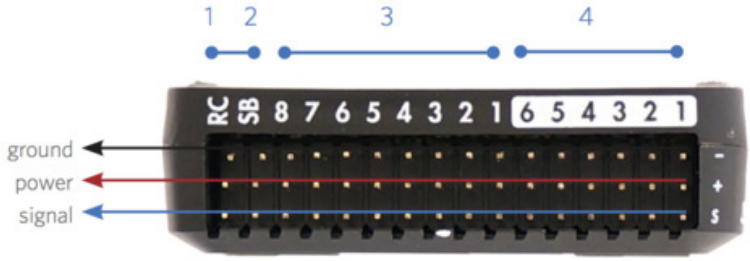
Though Pixhawk has an inbuilt compass, it is recommended to install another compass through the I2C port. An integrated GPS with compass was installed. Pixhawk draws power from the Li-Po battery. Pixhawk provides PWM control signals to the ESCs. The connection diagram is shown in Figure 5.12.

PX4 allows the user various levels of autopilot control. These include automation of take-off and landing, as well as maintaining attitude and following a given path. For the current application, the position hold, attitude hold and the path following modes are required. Attitude hold is essential for identifying the trash coordinates as detailed in Section 4.4. The path following mode allows the drone to survey the entire area, given a path via QGroundControl. This survey can either be performed initially, determining the boundaries or during the actual lake survey, when trash is identified.

QGroundControl is a ground control station that provides flight control and vehicle setup for PX4 and ArduPilot powered vehicles. It has a variety of facilities such as mission planning, way-point navigation with flight map display, video streaming and support for multiple vehicles. QGC is also compatible with a variety of Operating Systems, making it the ideal choice for autonomous flight control station for this project. A lake survey is first planned, where the entire lake is explored and boundaries are determined. A surveying path is then generated using QGC that instructs the drone the path to follow, while covering the lake surface to identify trash. Modes such as maintaining a constant altitude, speed, etc., are available in QGC, negating the need for a remote controller and allowing human-free flight.

Figure 5.12 shows the Pixhawk flight controller 2.4.8 which was purchased. This module is first configured using QGC. After all the connections are made, the drone is rotated in all angles and directions to calibrate the compass and gyroscope. Each motor is individually connected, powered and checked for clockwise or anticlockwise rotation. The user can either install clockwise and anticlockwise motors in the rec-

1 Input/output reset button
2 SD card
3 Flight management reset button
4 Micro-USB port



1 Radio control receiver input
2 S.Bus output
3 Main outputs
4 Auxiliary outputs

Figure 5.12: Pixhawk Autopilot wiring and connections

Source: *PX4 Documentation*

ommended configuration, or simply use clockwise motors and reverse the particular connections.

5.2.4 Sensors

In this section, an overview is provided, of the sensors the drone is equipped with. The camera is covered in a different section, as it is the main equipment here and is not functioning as a sensor for flying purposes. Drones fly in a variety of environments, and face a lot of disturbances. Ideally, a sensor should be robust enough to withstand vibrations and environmental factors, should be able to filter noise, should be able to survive shock impacts and should consume very less power.

Accelerometer

Accelerometers measure linear acceleration and force in three axes, X, Y and Z. They are vital sensors for preserving stability and orientation. Accelerometers come in many types and sizes, the most common ones being the piezo-electric and capacitive variants. Based on the force experienced in each direction, the tilt of the drone can be calculated. Accelerometers are also used to measure vibration and rate of change of altitude. This sensor can be installed additionally if one requires more precision. The current work uses the accelerometer that was inbuilt into Pixhawk.

Gyroscope

The gyroscope detects angular velocity in three axes. It detects the rate of change of roll, pitch and yaw. It works on the principle of conservation of angular momentum. The information about the change in roll pitch and yaw angles is used to provide stability and prevent drone wobble. Gyroscope information is fed to the motor control drivers dynamically to execute changes instantaneously. Camera stabilization is also performed using gyroscopic data in some advanced drones. Pixhawk also has an inbuilt

gyroscope that was sufficient for the current purpose.

The gyroscope and accelerometer are usually combined into a single sensor known as the Inertial Measurement Unit (IMU). For the drone, the inbuilt IMU was utilised.

Compass

The magnetic compass gives the data of the magnetic field in three axes, X, Y and Z. The data is then processed to identify the orientation with respect to the magnetic north. Combining the data with tilt information from the accelerometers gives more accuracy. The compass is highly sensitive to ferromagnetic materials and can also be used to detect such objects around the drone. Due to this sensitivity, a filter is also generally used to remove the noise, before calculating the orientation. This work uses a compass integrated into the GPS sensor module.

Barometer

A barometer converts the atmospheric pressure information into altitude data. This helps in drone navigation and achieving a set altitude. Estimation of speeds of ascent and descent, performed by the barometer, is of vital importance, especially during take-off and landing. The inbuilt Pixhawk barometer was used for the current work.

GPS

A GPS system determines the geographic coordinates of the drone by measuring the time taken for a signal to travel from a satellite. GPS modules also give an estimate of the drone's altitude. However, they are not very accurate and can only give position estimates to within 4m accuracy. But, by combining readings from other sensors the flight controller gets a more accurate value of its position. The GPS module can be used to identify way-points and is hence indispensable for autonomous flights. The GPS sensor

is usually integrated with a compass and is mounted on a small stand, high above the main drone body. This reduces interference both from external sources and from other drone components.

For the current objective, a Ublox NEO 7M GPS With Compass was purchased, that is compatible with Pixhawk 2.4.8. The 7M module was chosen over the 6M and 8M modules, optimising cost and sensor accuracy. The sensor and mounting method are shown in Figure 5.13.

Figure 5.13: The GPS Module, Stand and Mounting on the Drone

Source: *robu.in*

5.2.5 Video and Imaging

This section describes the necessary equipment for photography and videography, essential for the object detection algorithm to identify trash.

Camera

Advancing technology has enabled the miniaturization of HD cameras. This has facilitated their use in drones for a variety of applications like surveillance, aerial photography, etc. A standard camera is generally fixed on a drone, using a gimbal mount to preserve camera orientation during flight. The gimbal, based on its prescribed angle, also positions the camera at user-specified angles during flight. Drone cameras capture

at a variety of resolutions, the three most common being 720p, 1080p HD and 4K Ultra HD. Most cameras shoot at 30fps, while for advanced users, 60fps cameras are also available. Recent camera drones also have the flexibility to live stream. Some systems can transmit video from a distance of around 6 km too. This reduces the dependence upon memory card space and allows for first-person view (FPV) control. The camera also has a separate power source, like a small battery, in some drones. The images are captured in either RAW or JPG formats and are further processed.

A Campark Action Cam was purchased, that captures photos and records videos at 4K resolution and 30fps (Figure 5.14). It captures 16MP images with a 170°field of view. It can function with remote control and also has built-in Wi-Fi to transmit images live. This camera is also equipped with a waterproof case that allows photography at a depth of 98 feet underwater. It also has a long battery life, with two rechargeable 3.7V, 1050mAh batteries, that can last for 90 minutes. It is also equipped with a 32GB micro SD card for storage. This camera is suitable for the current purpose and also packs many additional features that are very useful for future work in this domain. The wide angle increases the area of coverage, and the high resolution is essential for object detection, as images with high resolution give better predictions.

5.2.6 Communication

This section explains the communication modules which consist of radio control and the telemetry module. These are vital for both manual and autonomous control.

Radio Control

To take control of vehicle movements in case of emergencies such as low battery situations or impending crashes, the remote control module is necessary. It is used to arm and disarm the vehicle and manually fly it, without the ground station. Autonomous flight does not require an RC system. The RC receiver is connected to Pixhawk through the 'RC input' port. A compatible RC transmitter remote is required, to send signals

Figure 5.14: Campark 4K Action Camera

Source: *https://www.amazon.in/Campark-Waterproof-Sports-Action-2Batteries/dp/B01MXXP4EC*

and control the vehicle's motion. The remote contains physical toggle keys to control parameters like throttle, pitch, and roll that affect the vehicle's heading and speed. Changes between different flight kmodes is also facilitated through the RC remote. A crucial parameter to choose in an RC remote is the number of channels it supports. This parameter decides how many different control commands can be sent to the vehicle. For ground vehicles and the ASV, two channels are sufficient to manipulate thrust and steering whereas for an aircraft, a minimum of 4 channels are required to control the thrust, yaw, pitch, and roll.

For this purpose, the drone is set up with FlySky FS-IA10B Radio Receiver, shown in Figure 5.15. This receiver has 10 channels and operates in the frequency range of 2.4055 to 2.475 GHz. It is compatible with many radio transmitters commercially available. A long range of around 1km can be achieved by fixing the antenna at an angle of 90°. This system is reliable and also incorporates interference removal modules, resulting in better communication.

Figure 5.15: The RC Receiver for manual control

Source: *https://robu.in/product/flysky-fs-ia10b-radio-receiver/*

Telemetry Module

The telemetry system facilitates communication between the drone and the Ground Control station. It is essential for to execute an autonomous mission. The telemetry receiver is connected to the Pixhawk flight controller through the Telemetry 1 port. The telemetry transmitter can be connected to a variety of devices, such as a mobile, a tablet, or a laptop, running the QGC software to create, load and monitor missions.

To communicate with ground stations, the autonomous vehicle is equipped with 3DR Single TTL MINI Radio Telemetry, shown in Figure 5.16. The module operates at a frequency of 433MHz and provides a range of 2.5km. It is compatible with QGC and also supports OTG to operate the vehicle using a mobile app. It works on the UART interface, with adaptive Time Division Multiplexing(TDM) being performed on the communication signals. Range extension can be achieved through manipulation of antenna length and angle.

Figure 5.16: The telemetry module

Source: *https://robu.in/product/3dr-singlr-ttl-radio-telemetry-433mhz-250mw-forpixhawk-and-apm*

5.2.7 Calculations

The total weight of the drone is 1269g. Adding any additional wiring, let the final take-off weight of the drone be 1300g. According to the design specifications, each motor can provide a thrust of 840g. For this motor, and a 3S Li-Po battery, the suggested propeller size is 1045 (10in length, 4.5in slope), while for a 4S battery, the suggested propeller is 8045. The total thrust provided by the 4 motors together is 840 x 4 = 3360g. This exceeds the drone weight by almost 2000g, providing a thrust-to-weight ratio of 2.58. This excess thrust is required for ease of control. The suggested ESC rating of the motor is 18A. The ESC must be able to handle the maximum current drawn by the motor and also the maximum current supplied by the PDB. The purchased PDB supplies a maximum steady current of 25A and a peak current of 30A for 10s/min. The ESCs purchased can support 30A of steady current and 40A of burst current for 10s. This ensures that the ESCs can handle the maximum current in the circuit, without damage. The Li-Po battery can supply a maximum of 208A (40C) continuously and 416A (80C) in burst, at a voltage of 11.1V. It contains 3 cells in series. Since this battery can supply the required current to all the four motors and has a large capacity of 5200mAh, suffi-

Table 5.2: List of Drone Components purchased

Component	**Quantity**	**Cost per item**	**Total Cost**	**Weight(g)**
Pixhawk 2.4.8	1	5950	5950	40
Q450 Frame + Landing Gear	1	1050	1050	405
Orange 5200mAh 3S 40C/80C Li-Po Battery	1	4149	4149	360
Ublox NEO 7M GPS With Compass	1	1469	1469	26
EMAX MT2213 935KV BLDC Motor CW + Propellers	2	1180	2360	106
EMAX MT2213 935KV BLDC Motor CCW + Propellers	2	1180	2360	106
Emax BLHeli Series 30A ESC	4	1039	4156	112
PDB XT60 with BEC 5V and 12V	1	399	399	11
FlySky FS-IA10B Radio Receiver	1	1782	1782	15
3DR Single TTL MINI Radio Telemetry 433MHz	1	2250	2250	15
Li-Po Battery Voltage Checker	1	125	125	-
Plastic GPS Antenna	1	199	199	8
XT60H Connector	1	59	59	6
Campark Action Camera	1	5090	5099	59
Miscellaneous	-	99	99	-
Total	-	-	**31506**	**1269**

cient for good flight time, it was chosen for the current application. The expected flight time using this battery is 15 minutes.

Chapter 6

CONCLUSIONS AND SCOPE FOR FUTURE WORK

6.1 Summary and Highlights

The aim of this work was to develop an autonomous system that employed different types of mobile robots collaborating to achieve the goal of clearing litter from the surface of lakes. This involves integrating and improving a large number of separate modules, both hardware and software, each requiring extensive background study. This work was conceived as a cheaper and more efficient alternative to the existing solutions, which were detailed in Chapter 3.

The use of deep learning methods for the identification and classification of floating litter has been explored and found to be satisfactory, automation-friendly and fast, addressing critical problems faced by earlier efforts in this area. Mounting this system on an aerial robot eliminates the need for a human observer for the identification of litter, making the system autonomous. Also, aerial robots are faster and more versatile than marine robots and can increase the speed of surveying. Their ability to be easily deployed in remote or tough to access areas is another positive. Shifting the task of litter identification to the aerial robots, reduces the load on the marine robot by eliminating the usage of advanced sensors. This also reduces the size of the marine robot, making it easier to deploy and improving litter collection. Having divided the task between different mobile robots, a communication system is required that can translate the data from one robot into information for the second. Algorithms were developed to generate global way-points from the detections given by the aerial robot. These are the main software systems that were developed for the current project.

For the physical testing of the developed algorithms and models, a quadcopter was assembled, using components purchased online, as detailed in Section 5.2. The way-point following and altitude hold functionalities were tested and found to be working

well. Object detection was tested using pictures and videos taken with mobile cameras. The model was found to generalise well to new images and hence was found to be satisfactory. The way-point generation algorithm and the navigation algorithm were tested using GPS coordinates from the IIT Madras lake and were found to be working well.

6.2 Future work

Future work in this project will be on the improvisations to each of the current modules. The object detection module can be improved to include more classes of objects and also refine the identification of obstacles. It could also be trained to identify lake boundaries. The drone can be modified to use the live streaming facility, which has not been tested in this work. Also, multiple drones could be deployed to increase the speed of aerial survey. Modifications to the boat could include trash segregation facilities, using the classification data from the drone. A slightly complicated modification could be to include a recharging platform on the boat, so that the drone can return to it for charging. As a starting step, this platform could first be arranged on the bank. Incorporating recharging facility would require the system to function simultaneously. Small changes can be made to the entire system to facilitate simultaneous data gathering, processing and communication. This will allow the drone and the boat to work together, real-time, instead of sequentially, cutting down the collection time even further.

Appendix A

APPENDIX

Code Listing A.1: Extraction of required parameters of detected box

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from csv import reader

# Code to extract the required paramters of the litter from detection
                                    results
with open('Detections.csv', 'r') as read_obj_r:
    # Pass the file object to reader() to get the reader object
    csv_reader_r = reader(read_obj_r)
    # Iterate over each row in the csv using reader object
    # Cardboard 0, Glass 1, Metal 2, Paper 3, Plastic 4, Trash 5
    # Legend for each class
    for row in csv_reader_r:
        if row[0][0]=='E':
            pass

        else:
            if "Cardboard" in row[0]:
                trashtype=0
            elif "Glass" in row[0]:
                trashtype=1
            elif "Metal" in row[0]:
                trashtype=2
            elif "Paper" in row[0]:
                trashtype=3
            elif "Plastic" in row[0]:
                trashtype=4
            elif "Trash" in row[0]:
                trashtype=5
```

```
# To extract required data fron the detection output
uu = row[0].find("left_x:")
left = row[0][uu+9:uu+12]
top= row[0][uu+23:uu+26]
width=row[0][uu+38:uu+41]
height=row[0][uu+53:uu+55]

# To calculate the coordinates of the centre of the image
left1 = str(int(left) + (int(width)/2))
top1 = str(int(top) + (int(height)/2))

# Export the results to a new .txt file
with open('trashtype.txt', 'a') as the_file:
    the_file.write(str(trashtype)+','+left+','+top+','+
                                        width+','+height+'
                                        \n')
    the_file.write(str(trashtype)+'\n')

# Saving top and left coordinate values of the detected
                                    box
with open('left.txt', 'a') as the_file:
    the_file.write(left1+'\n')
with open('top.txt', 'a') as the_file:
    the_file.write(top1+'\n')
```

Code Listing A.2: Calculation of GPS coordinates of the image origin

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from csv import reader

# Code to calculate latitude and longitude of the detections

fovvert = 60 # vertical field of view
fovhor = 120 # horizontal field of view
camera_angle = 60 # wrt the horizontal
camera_vert = 90 - camera_angle


with open('gps.csv', 'r') as read_obj:
    # Pass the file object to reader() to get the reader object
    csv_reader = reader(read_obj)

    # Code to extract latitude and longitude of the origin of the
                                        projection on the surface
    # gps.csv contains latitude, longitude, height and orientation
                                        values of the drone

    for row in csv_reader:
        imagelat = int(row[1]) + int(row[2])*np.tan(np.deg2rad(
                                            camera_vert))*np.cos(np.
                                            deg2rad(int(row[3])))
        imagelong = int(row[0]) + int(row[2])*np.tan(np.deg2rad(
                                            camera_vert))*np.sin(np.
                                            deg2rad(int(row[3])))
        hyp = int(row[2])/np.sin(np.deg2rad(camera_vert))
        ang_B = camera_vert + 90
        ang_C = 180 - ang_B - (fovvert/2)
        vert_dist = (hyp/np.sin(np.deg2rad(ang_C)))*np.sin(np.deg2rad
                                            (fovvert/2))
        hor_dist = hyp*np.tan(np.deg2rad(fovhor/2))
        lat1 = imagelat + vert_dist*np.cos(np.deg2rad(int(row[3])))
        long1 = imagelong + vert_dist*np.sin(np.deg2rad(int(row[3])))
        alpha = int(row[3])
```

```
        lat2 = lat1 + hor_dist*np.sin(np.deg2rad(alpha))
        long2 = long1 - hor_dist*np.cos(np.deg2rad(alpha))
        y = ang_B - (fovvert/2)
        vert_dist_down = (hyp/np.sin(np.deg2rad(y)))*np.sin(np.
                                            deg2rad(fovvert/2))
        vertical = vert_dist + vert_dist_down # Vertical side length
                                            of projected rectangle
        horizontal = hor_dist*2                 # Horizontal side
                                            length of projected
                                            rectangle
        print(vertical, horizontal)

        # Writing the data to new .txt files
        with open('lat.txt', 'a') as the_file:
                the_file.write(str(lat2)+'\n')
        with open('long.txt', 'a') as the_file:
                the_file.write(str(long2)+'\n')
        with open('theta.txt', 'a') as the_file:
                the_file.write(row[3]+'\n')
```

Code Listing A.3: Calculation of GPS coordinates of the detected litter

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from csv import reader

# Calculated values of vertical and horizontal side distances defined
vertical = 6.0
horizontal = 13.856

# image_final.csv is the csv file containing litter category and
                                    projected rectangle parameters
with open('image_final.csv', 'r') as read_obj2:
    # Pass the file object to reader() to get the reader object
    csv_reader2 = reader(read_obj2)

    # Code to calculate the latitude and longitude of the object from
                                        the projection origin
    for row in csv_reader2:
        # Distance from origin
        dist = np.sqrt(np.square(float(row[1])*horizontal/512.0) + np
                                            .square(float(row[2])*
                                            vertical/384.0))

        # Angle assuming top side as the x-axis
        beta = np.arctan2((float(row[2])*vertical/384.0), (float(row[
                                            1])*horizontal/512.0)) *
                                            180 / np.pi

        x = 90 - float(row[5]) - beta
        latobj = float(row[3]) - dist*np.cos(np.deg2rad(x))
        longobj = float(row[4]) + dist*np.sin(np.deg2rad(x))

        # Writing the latitude and longitude calculated to csv files
        with open('latobj.txt', 'a') as the_file:
                the_file.write(str(latobj)+'\n')
        with open('longobj.txt', 'a') as the_file:
                the_file.write(str(longobj)+'\n')
```

Bibliography

[1] **Abrams, M.** (2018). Remote robot cleans trash from water. URL https://www.asme.org/topics-resources/content/remote-robot-cleans-trash-water/.

[2] **Agrawal, P.** and **B. Bhattacharya**, Aquatic multi-robot system for lake cleaning. 2013. ISBN 978-981-4525-52-7.

[3] **Bai, J.**, **S. Lian**, **Z. Liu**, **K. Wang**, and **D. Liu** (2018). Deep learning based robot for automatically picking up garbage on the grass. *IEEE Transactions on Consumer Electronics*, **PP**, 1–1.

[4] **Barrow, H.**, Chapter 5 - connectionism and neural networks. *In* **M. A. Boden** (ed.), *Artificial Intelligence*, Handbook of Perception and Cognition. Academic Press, San Diego, 1996. ISBN 978-0-12-161964-0, 135 – 155. URL http://www.sciencedirect.com/science/article/pii/B9780121619640500078.

[5] **Bochkovskiy, A.**, **C.-Y. Wang**, and **H.-Y. M. Liao** (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

[6] **Duchi, J.**, **E. Hazan**, and **Y. Singer** (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, **12**, 2121–2159.

[7] **GetFPV** (2020). Quadcopters. URL https://www.getfpv.com/.

[8] **Girshick, R.**, **J. Donahue**, **T. Darrell**, and **J. Malik**, Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[9] **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun**, Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[10] **Hochreiter, S.** and **J. Schmidhuber** (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[11] **Jiannan Zhu, Y. C. Z. W., Guo Mengrui** and **Y. Qiao** (2019). Orca: China's river cleaning robot. URL https://medium.com/dyson-on/orca-chinas-river-cleaning-robot-929ce62269c4.

[12] **Kaushal Patwardhan, A. K., Shivraj Hagawane**, Aqua dredger river cleaning machine. 2020.

[13] **Kiefer, J.** and **J. Wolfowitz** (1952). Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, **23**(3), 462–466. URL https://doi.org/10.1214/aoms/1177729392.

[14] **Kingma, D. P.** and **J. Ba** (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[15] **Krizhevsky, A.**, **I. Sutskever**, and **G. E. Hinton**, Imagenet classification with deep convolutional neural networks. *In* **F. Pereira**, **C. J. C. Burges**, **L. Bottou**, and **K. Q. Weinberger** (eds.), *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, 1097–1105. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[16] **Lecun, Y.**, **L. Bottou**, **Y. Bengio**, and **P. Haffner** (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

[17] **Lecun, Y.**, **P. Haffner**, and **Y. Bengio** (2000). Object recognition with gradient-based learning.

[18] **Lecun, Y.**, **P. Haffner**, and **Y. Bengio** (2000). Object recognition with gradient-based learning.

[19] **Leshno, M.**, **V. Y. Lin**, **A. Pinkus**, and **S. Schocken** (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, **6**(6), 861 – 867. ISSN 0893-6080. URL http://www.sciencedirect.com/science/article/pii/S0893608005801315.

[20] **Liu, W.**, **D. Anguelov**, **D. Erhan**, **C. Szegedy**, **S. Reed**, **C. Fu**, and **A. Berg** (). Ssd: Single shot multibox detector. arxiv 2016. *arXiv preprint arXiv:1512.02325*.

[21] **of Military, D.** and **A. Terms** (2005). Uav. URL https://www.thefreedictionary.com/unmanned+aerial+vehicle.

[22] **Pixhawk** (2020). Pixhawk 1 flight controller. URL https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk.html.

[23] **Poudel, A.** (2019). Aerodynamics forces: How does an aircraft fly? URL https://www.geniuserc.com/how-aircraft-fly-and-aerodynamics-forces/.

[24] **Proença, P. F.** and **Mathieu** (2020). pedropro/TACO: Mask-RCNN weights trained on TACO-10. URL https://doi.org/10.5281/zenodo.3698158.

[25] **Redmon, J.** and **A. Farhadi** (2016). Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*.

[26] **Redmon, J.** and **A. Farhadi**, Yolo9000: Better, faster, stronger. *In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[27] **Redmon, J.** and **A. Farhadi** (2019). Yolov3: An incremental improvement. arxiv 2018. *arXiv preprint arXiv:1804.02767*.

[28] **Refugio, C.**, **D. Cristuta**, **S. Abjelina**, **G. Apoli**, **E. Anqui**, and **A. Valera** (2019). *Solar-Powered-Trash-Collector-on-Water (1)*. Ph.D. thesis.

[29] **Reid, J.** (2017). Multirotor motor guide. URL https://www.rotordronepro.com/guide-multirotor-motors/.

[30] **Rumelhart, D. E.** (1986). Parallel distributed processing: Explorations in the microstructure of cognition. *Learning internal representations by error propagation*, **1**, 318–362.

[31] **Simonyan, K.** and **A. Zisserman** (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[32] **Sinha, A.**, **P. Bhardwaj**, **B. Vaibhav**, and **N. Mohommad**, Research and development of ro-boat: an autonomous river cleaning robot. *In* **J. Röning** and **D. Casasent** (eds.), *Intelligent Robots and Computer Vision XXXI: Algorithms and Techniques*, volume 9025. International Society for Optics and Photonics, SPIE, 2014. URL https://doi.org/10.1117/12.2037898.

[33] **Sirsat, M.**, **D. Khan**, **M. Jadhav**, and **M. Date**, Design and fabrication of river waste cleaning machine. 2017.

[34] **Szegedy, C.**, **Wei Liu**, **Yangqing Jia**, **P. Sermanet**, **S. Reed**, **D. Anguelov**, **D. Erhan**, **V. Vanhoucke**, and **A. Rabinovich**, Going deeper with convolutions. *In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[35] **Thung, G.** (2017). Classification of trash for recyclability status. https://github.com/garythung/trashnet.

[36] **Vu, M.-T.**, **B.-A. Marie**, and **L. Van Linh**, Heritage image classification by convolution neural networks. 2018.

[37] **Zeiler, M. D.** (2012). Adadelta: An adaptive learning rate method. arxiv 2012. *arXiv preprint arXiv:1212.5701*, **1212**.