Surveillance in Maritime Scenario Using Deep-Learning and Swarm Intelligence

by

Nazmul Islam
18101321
MD. Samiur Rahman Bhuiya
18101584
Ayesha Siddiqua Drishty
18101674
Snigdha Suparna Saha
18101385
Utsha Das Akash
18101322

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
May 2022

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

Nazmul Islam

18101321

[illegible]

MD. Samiur Rahman Bhuiya

18101584

[illegible]

Ayesha Siddiqua Drishty

18101674

[illegible]

Snigdha Suparna Saha

18101385

[illegible]

Utsha Das Akash

18101322

Approval

The thesis titled "Surveillance in Maritime Scenario Using Deep-Learning and Swarm Intelligence" submitted by

1. Nazmul Islam (18101321)
2. MD. Samiur Rahman Bhuiya (18101584)
3. Ayesha Siddiqua Drishty (18101674)
4. Snigdha Suparna Saha (18101385)
5. Utsha Das Akash (18101322)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 24, 2022.

Examining Committee:

Supervisor:
(Member)

[illegible]

Dr. Amitabha Chakrabarty

Associate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Unmanned Aerial Vehicles (UAVs) have played a crucial role in supporting Search and Rescue (SAR) Operations due to their fast movement capabilities and flexibility. During a search and rescue operation scenario, the time constraint is a crucial parameter, so the required time to detect humans in distress with precision is also a vital part. Modern Deep-learning algorithms like CNN also aid in these missions. However, most models and datasets available focus on search and rescue missions on the ground or land. UAV-based search and rescue operations in the Maritime Scenario remain a challenge. This study focused on using deep learning algorithms such as CNN to precisely detect a human in peril with a swarm of drones. At the same time, we emphasize using swarm intelligence algorithms such as Particle Swarm Algorithm (PSO) to effectively find a victim in the shortest time by exploring a massive area. The distinctiveness of this system is that it combines the model with the best Accuracy to detect and the best swarm intelligence algorithm for finding targets in the quickest time possible, thus enhancing the surveillance mission. In this research, among VGG16, ResNet50V2, InceptionV3, Xception and MobileNetv2 models, VGG16 produced IoU (Intersection over Union) score of 0.62 with Class Label accuracy of 99.15% and Bounding Box accuracy of 88.74% in CNN part. Along with that, among three different swarm intelligence algorithms, according to the simulation, Particle Swarm Optimization Algorithm took the minimum average time which is 20.4 units, whereas the Grey Wolf Optimization algorithm and Bat Optimization Algorithm, respectively took 65.6 and 73.8 unit of time.

Keywords: Object Detection; Marine Search and Rescue (SAR); Unmanned Aerial Vehicles (UAV); Convolutional Neural Network; Swarm Intelligence.

Dedication

We dedicate our work to our loving and supportive parents, without whom we would not have made it this far. A special thanks direct from our heart to our extremely helpful supervisor.

Acknowledgement

First and foremost, we wholeheartedly praise the Almighty Creator for allowing us to complete our thesis.
Next, we would love to express our gratitude to our respected advisor Dr. Amitabha Chakrabarty sir, for his unwavering support.
Finally, to our beloved parents for their constant love and support.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AFS	Artificial fish swarm
AI	Artificial Intelligence
ATSS	Adaptive Training Sample Selection
BA	Bat Optimization
CNN	Convolutional Neural Network
FC	Fully Connected Layer
GCN	Graph Convolutional Network
GPGPU	General-purpose Computing on Graphics Processing Units
GPS	Global Positioning System
GWO	Grey Wolf Optimization
ReLU	Rectified Linear Unit
ResNet	Residual Networks
ROI	Region of Interest
SAR	Search And Rescue
SI	Swarm intelligence
SSD	Single Shot Detector
UAV	Unmanned aerial vehicles
VGG	Visual Geometry Group

Chapter 1

Introduction

Aerial imaging is increasingly being used in mission-critical applications like traffic monitoring, agriculture, smart cities, disaster relief, and many more. Unmanned aerial vehicles (UAVs) with a high-resolution camera can be used for a variety of purposes, as detailed above [49]. UAVs, specifically, are capable of assisting in SAR (Search And Rescue) operations due to their quick and diverse use, as well as their capacity to offer an analysis of the situation [52].

The efficient deployment of autonomous UAVs is especially important in maritime scenarios, where large regions must be promptly surveyed and searched [21]. UAVs help rescue efforts by providing airborne imagery, topographic mapping, emergency delivery, and reducing unnecessary human hazards. One of the most difficult sectors in its application scenario is detecting, locating, and monitoring people in the marine environment [32] [42]. Deep neural networks, as well as other data-driven procedures, are currently being used to build these systems. These approaches are based on large data sets reflecting actual environments to generate effective visual statistics. Large-scale data sets, on the other hand, are scarce in maritime environments. Most UAV data sets, such as VisDrone [37] and UAVDT [31], are land-based and frequently focus on traffic conditions. Most of the limited data sets obtained in maritime areas are classified as remotely sensed data, and satellite-based radars are widely used. As they lack the resolution required for SAR operations, they are only suitable for spotting ships [7]. Moreover, satellite imaging is subject to cloud cover and provides only top-down images. For this study, we used SeaDronesSee [62], a large-scale data set of humans and other important objects in maritime scenarios.

This dataset includes videos and images of objects like humans and boats in open water taken with several drones and cameras. Moreover, this dataset contains many categories for items of importance such as swimmer, life jacket, floater, human on the boat, and boat, which will be useful for recognizing humans in marine contexts [57]. This will be helpful for rescuing humans by detecting the objects and class. Furthermore, we will use swarm intelligence to make the surveillance faster and more efficient. A swarm is a gathering of identical, simple agents, which are mainly the agents of nature like bees, ants, and birds, engaging locally with each other and their own surroundings, with no centralized control to permit interesting global behavior to arise. Swarm behavior and the procedures are a relatively new class of nature-inspired algorithms that can deliver cost-effective, quick, and accurate results to many kinds of challenges [11] [6]. Swarm intelligence is an emerging division of AI that aims to emulate the coordinated activity of swarms in nature, such as ants, bats, wolves, birds, and others. Our proposed system architecture will implement the concept of nature's swarm behavior to search and save human lives from maritime accidents.

1.1 Motivation

As a result of economic globalization, interest in air and sea travel has increased tremendously throughout all regions. Natural disasters and maritime accidents, on the other hand, are increasing year after year. After an accident, people's odds of survival are directly proportionate to how quickly they are rescued. Due to a lack of detection and rescue procedures in the water, many individuals died. Artificial vision and electronic radar surveillance are limited due to errors and resource limitations. So, detection of humans in maritime scenarios is a necessary process that can help and rescue many people in times of disaster as well as marine accidents. As mentioned earlier, with the help of our dataset using deep learning, we can detect whether the person is in danger or not by detecting ground-truth labels for items of interest, such as swimmer or floater. If it detects the swimmer, then we assume that the person is alive and need immediate help. On the other hand, if the person is a floater with a life jacket, then we will detect the person as someone who does not need emergency help. Now the detection process is done by data-driven methods like convolutional neural networks. Airborne cameras can now produce stable, high-resolution photographs owing to the rapid advancements in the Unmanned Aerial Vehicle (UAV) technology over the years.

SeaDronesSee is a huge data collection project that involves individuals in the maritime environment. Existing UAV data sets provide either very little or no metadata. We believe that it is a key roadblock in the creation of multi-model systems that utilize this further data in order to improve precision or efficiency. Besides, we are using the UAV instead of a helicopter and boat in case of detection because UAVs are cost-effective and it can easily detect the swimmer and floater. On the other hand, a helicopter cannot capture stable images and is also costly to operate. Besides, the boat requires more time in the detection process as the field of vision is limited. So, the optimal solution for detecting humans in maritime scenarios is to use UAVs. Moreover, we will use swarm intelligence for faster exploration of the disaster area, which will make our surveillance system faster. Swarm robotics can be defined as a system of simple robots capable of displaying collective intelligence behavior. Robustness, versatility, and scalability are all desirable features for swarms of robots [5]. As a result, we will have efficient surveillance for our system.

1.2 Problem Statement

In maritime scenarios, the effective utilization of Unmanned Aerial Vehicles (UAVs) is essential [57]. However, in this application scenario, different types of issues arise. Firstly, the weather is the main problem factor in detecting people from aerial images in open water [32]. Besides, people cannot be located and tracked smoothly. Secondly, in marine contexts, large-scale data sets are in short supply [57]. However, the dataset's images of the objects like boats, trees, and land vehicles can be mostly found. Furthermore, finding little items in high-resolution photos is a problem when recognizing objects in aerial photographs. If the image is obtained from a satellite, there is no clear image in the collection because the range of capturing the images is not adequate. The sea condition is another issue in real-life marine environments. In the open water, the body's visibility is quite changeable. As half of the body is submerged, the entire body cannot be seen. Due to the movement of waves in maritime environments, all of these factors make it more difficult to perform accurate body detection offshore [32]. The camera orientation and UAV motion play an essential role in recognizing the body's appearance. Additionally, when the individual is alive, their body keeps moving, which alters position and visibility. If the individual is unconscious, then the movement of the waves as well as the UAV may produce significant variations in observation [32]. As a result, a lot of irrelevant data captured by UAVs make the surveillance process slower as well as faulty. Hence, it is crucial to recognize the objects and prioritize humans correctly. In this research, we intend to detect objects using deep learning and enhance the detection speed with the help of swarm intelligence algorithms.

1.3 Research Objectives

The research aims to recognize humans from different classes like swimmers and floaters in maritime scenarios to make the surveillance operation more efficient. The objectives of this research are

1. Using deep learning models to accurately detect objects like swimmers, floaters, and boats.
2. Making the exploration process faster using swarm intelligence models.

Chapter 2

Literature Review

In this chapter, we analyzed different kinds of research papers similar to our research field. We learned about different methodologies and their implementation in different sectors, mostly object detection and swarm intelligence. Furthermore, we analyzed the advantages as well as flaws of those methodologies and also compared some of the methods described in the research papers with our selected models.

2.1 Human detection in Maritime Scenario

Machine learning is extensively used to classify items into their respective categories. Deep learning has been used in picture categorization and object detection in recent years. Deep learning uses a neural network with multiple hidden layers to improve image categorization ability. The Convolutional Neural Network (CNN) is one of the most commonly used deep learning neural networks for image categorization and object detection. There are two types of CNN-based object trackers. One is anchor-based, and another is anchor-free. The anchor-based detectors find objects using produced anchor candidates and suppress negative anchors with a fixed IoU. The items that are detected by the anchor-free detectors use point estimation.

2.2 UAV with Swarm Intelligence for Surveillance

2.2.1 Intro to UAV

Unmanned Aerial Vehicles (UAVs) have gained significant popularity. UAVs have been used to carry out complicated and sophisticated missions, including long-term and dangerous missions. Environmental monitoring, search activities, and surveillance are some examples of this type of application [12]. It happened primarily as a result of investments in embedded computers, telecommunications, sensor devices, and low-power technologies. Object detection has a lot of potential in the domain of UAVs. Due to its small weight, limited flight altitude, and unpredictable travel route, images from a UAV may be warped or jittered. As a result, in UAV object detection, algorithm efficiency, precision, and size of the model are all crucial [59]. Surveillance can be faster with UAVs with the help of swarm intelligence. The resolution of coverage issues, as well as efficient environment exploration and interaction between UAVs, are all central issues in this field [15]. Furthermore, compared to a single UAV system, which typically has a limited supply of energy, low computing potential, and poor durability, a multi-UAV system is intended to provide a much broader range, effective monitoring and comprehension of the area of interest, better and faster decision, and therefore better support of a wide range of applications [23]. To deliver these benefits, UAV Networks (UAVNs) collaborate [39].

2.2.2 Swarm Behavior

As previously stated, a swarm is a cluster of uniform agents that interact directly with one another and their surroundings, with no centralized control, allowing for the emergence of a variety of intriguing behaviors. To accomplish global emergent behavior, swarm intelligence promises to be able to control enormous networks of interconnected individuals using only local communication. They usually do not obey orders from a leader or a master plan. Swarm intelligence is helpful in a variety of technical applications, including multi-robot formation control, huge distributed sensing utilizing mobile sensor networks, fighting using cooperative UAVs, flocking, and so on [10]. Because of the exceptional efficacy of these biological swarm systems, biologists and environmentalists have been researching insect behavior for years. The amount of research articles demonstrating the successful implementation for Swarm Optimization algorithm in a variety of efficient tasks and research difficulties has steadily increased since the computer modeling of swarms was established. The principles of swarm intelligence are implemented to a wide range of issues, including optimum route discovery, planning, structural analysis, and photo and information processing. Swarms have been computationally modeled in a range of fields, including computer vision, bioinformatics, health informatics, stochastic processes, and operations research [8]. Swarm behavior occurs when robots exhibit collective activity through interaction with each individual robot in their vicinity. An assembly of robots demonstrating swarm behavior may be built and used for search and rescue operations, which would not only avoid putting human rescue and surveillance personnel in danger but would also boost the efficiency of the process [47]. When used for search and rescue missions, the concept of the aerial swarm, in which UAVs behave in a swarm above ground level, would be a big step forward [3].

2.3 Reviewed Topics

We analyzed these research projects' contemporary designs and solutions utilized during the research. Some of the most broadly adopted architectures are:

2.3.1 R-CNN Cascade

The Cascade R-CNN approach detects objects in multiple stages [30]. This method requires a large number of classifiers To accomplish the classification operation for each classifier's bounding box area. The next stage's classifier completes the classification work from the previous phase for the freshly built bounding box area. The bounding box created by repeating the previous procedure is predicted to become more precise. Each step's classifier only adjusts for samples with a higher IoU than the previous step's classifier. Thus, it was capable of learning high-performance object recognition. In its architecture, two parts are discussed and proposed.

1. **Cascaded Bounding Box Regression:** It is structured as a cascaded regression problem in the Cascade R-CNN. A series of customized regressors are used to do this,
$$f(x,\ \mathrm{b}) = f_T\mathrm{o}\ \ f_{T-1}\mathrm{o}\cdots\mathrm{o}f_1(x,\ \mathrm{b}), \tag{2.1}$$
Here in equation 2.1 T is the entire number of steps in the cascade. Each regressor f_T in cascade is enhanced with respect to the dispersion of samples b^t arriving at the appropriate stage, rather than the initial allocation of b^1. This cascade improves hypotheses progressively.

2. **Cascaded Detection:** The Cascade R-CNN uses cascade regression as a resampling approach for addressing the image resolution issue. At each stage t. The R-CNN includes a classifier h_t and a regressor f_t optimized for IoU threshold u^t, where $u^t > u^t 1$. This is learned by minimizing the loss
$$L(x^t,\ g) = L_{cls}(h_t(x^t),\ y^t) + \lambda[y^t \geq 1]L_{loc}(f_t(x^t,\ \mathrm{b}^t),\ \mathrm{g}) \tag{2.2}$$
where $b^t = f_{t-1}(x^{t-1}, b^{T-1})$,g is the x^t ground truth object, λ=1 is the trade-off parameter, [.] is the indicator function, cross-entropy loss $L_{cls}(h(x_i), y_i)$ and h^t is the label of x^t given u^t by the given equation in 2.3:
$$y = \begin{cases} g_y, & IoU(x,\ g) \geq u \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$
In the equation 2.3, if the IoU is above a threshold u, the patch is considered an example of the class. Thus, the class label of a hypothesis x is a function of u.

   In contrast to the entire loss of equation 2.2, this ensures a series of well-trained detectors of improved quality.

2.3.2 HRNet

The object detection problem, unlike object classification, is position-sensitive, necessitating high-resolution representation [58]. HRNet maintains a high-resolution illustration throughout the procedure, unlike other techniques such as AlexNet [14], VGGNet [20], and ResNet [24]. As a result, HRNet performed well in posture estimation and semantic segmentation, and it may also be used to recognize small objects. When objects of various sizes are integrated into a single image, HRNet's above features make it excellent for usage in marine images.
As seen in Figure 2.1, this network, dubbed HRNet (High Resolution Network), is made up of multiple simultaneous phases. A high-resolution sub-network is used in the first stage. Sub-networks of increasing resolution are added one after another to create new layers [53].
The dilemma of sample instability in the binary classification problem can be solved by merging two loss functions of DICE loss and BCE loss [45]. The formula for DICE loss is as follows in 2.4:
$$L_{dl} = dice\ loss = 1 - \frac{2|G\mathrm{I}\ P|}{|Y| + |P|} \tag{2.4}$$

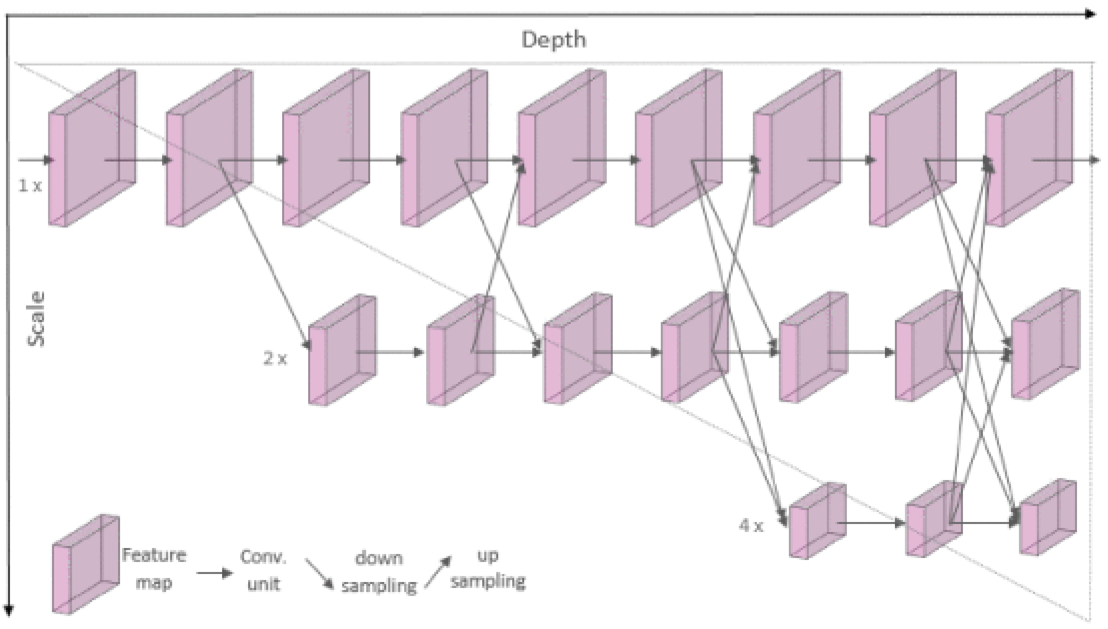


Figure 2.1: HRNet (High Resolution Network) [58].

In the equation, G and P are denoted by the ground truth and prediction result respectively. The number of components in common between G and P is expressed by GI P, which is obtained by adding the pixel-wise components of the two matrices. The sum of matrix elements is represented by this value | |. In semantic segmentation problems, the Softmax loss function is widely employed as a loss function. The formula for calculating softmax loss is as follow in equation 2.5:

$$L_{sl} = softmax\ loss = -\sum_{c=0}^{C} y_c \log(\frac{e^{p_k}}{\sum_j e^{p_j}}) \tag{2.5}$$

Here, p is the network output probability and y is the ground truth. To enhance the stability of system training, DICE loss is frequently combined with BCE loss in reality, so overall loss is specified as equation 2.6

$$Loss = L_{DICE} + L_{BCE} \tag{2.6}$$

2.3.3 Graph Convolutional Network (GCN)

Graph convolutional networks can transfer information between close graph data nodes. For this reason, GCNs are commonly used in text categorization, relationship retrieval, and image processing [61]. GCN is employed in various fields such as molecular science, traffic predicting, social networking, and other areas where data has a distinct graph structure. The data from radar signals do not have a strong schema; however, it contains graph data in time and geographical domains. When evaluating a portion of data, the signals gathered at the subsequent location and time can provide critical data. As a result, graph information is an ideal technique to represent radar signals so that the information contained in them may be used entirely.

Figure 2.2 shows the flowchart for the graph-based technique, which comprises two steps. Firstly, radar signal graph information is constructed using the resident radar

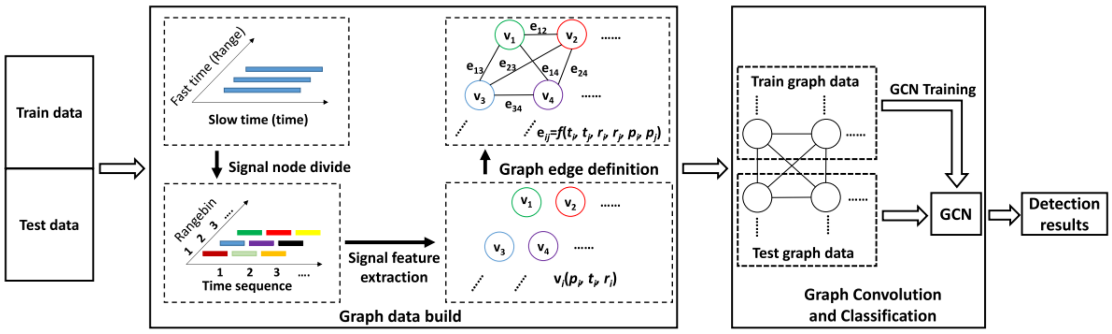


Figure 2.2: Detection of objects using GCN and radar graph data [61].

signal's derived time-range information. The nodes are one-dimensional signal portions inside a single rangebin, and the node characteristics are the matching signal's frequency distribution. Secondly, the edges are defined using timing, location, and power information. Thirdly, GCN is used to accomplish target and clutter binary categorization, which outputs the possibility of each node, detecting units, getting evaluated as clutter or target.
Each layer in GCNs aggregates the features of surrounding nodes to update the node feature embedding in the graph:

$$Z^{l+1} = concat(X^l, \tilde{A}_+X^l, \tilde{A}_-X^l)W, \tag{2.7}$$
$$X^{l+1} = \sigma(Z^{l+1}) \tag{2.8}$$

Here in equation 2.8 X^l is the embedding at the layer l for all the nodes, and W is the feature transformation matrix which will be learnt for the node classification. The activation function is usually set to be the element-wise ReLU Here W is the feature transformation matrix that will be learned for node categorization, and X^l is the embedding at layer l for every node. The component ReLU is commonly used as the activation function [48].

2.3.4 Long Short-Term Memory (LSTM)

A more advanced form of the RNN is the LSTM [44]. RNNs face difficulty with vanishing gradients, and LSTMs have been proposed as a remedy. LSTMs, unlike RNNs, can detect both short-term as well as long-term temporal correlations in information. The simplest LSTM has an input gate, an output gate, block input, and a forget gate. In figure 2.3 we can see the inner-mechanism of a LSTM block.
The equations in 2.13[35] can be used to describe how a memory block functions at each time step t.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.9}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.10}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_c x_t + U_c h_{t-1} + b_c) \tag{2.11}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_o) \tag{2.12}$$
$$h_t = 0_t \odot (c_t) \tag{2.13}$$

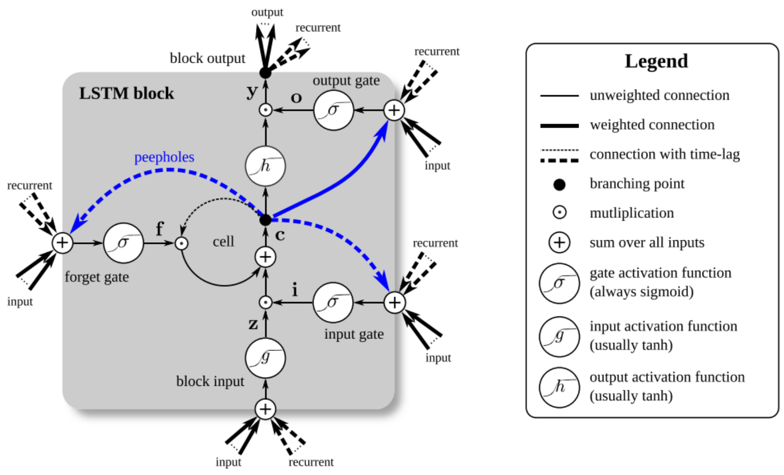


Figure 2.3: LSTM block [44].

In the five equations in 2.13, xt is the input of the model at time t, the weight matrices are denoted by Wi, Wf, Wc, Wo, Ui, Uf, Uc, Uo, Vo, and bias vectors are represented by bi, bf, bc, bo. Besides, at time t, ct is the state of the memory cell, ht is the memory block's outcome and the activations of the three gates are it, ft, 0t are respectively. The scalar product of two vectors is represented by .
The gate activation function (x) is a conventional logistic sigmoid function as described in 2.14, with gate activations ranging from 0 to 1. A value of 0 indicates that the gate is blocked, whereas a value of 1 indicates that the gate is open.
The tanh functions g(x) and h(x) for cell input and output activation have a static mean of 0. The equation is given in 2.15

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.14}$$

$$g(x) = h(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.15}$$

2.3.5 Faster R-CNN

Faster R-CNN is a standard object identification network that uses a convolutional neural network to extract visual features [51]. The region suggestions network module creates region proposals and tells the object classification module where to look. In order to generate an anchor, a small network is dragged over the convolutional feature map output within the last common convolution layers. Anchors have three scales and three aspect ratios by default, which fluctuate in box regression and play a key role in collecting region ideas. The area of interest pooling layer generates a fixed-size feature vector corresponding to the convolutional feature map's region proposal. The feature vector from ROI (region of interest) pooling and the object classification score determine object categorization. Figure 2.4 depicts the design of the faster R-CNN,
The loss function of Faster R-CNN network [50] is shown in equation 2.16.

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i,p_i^*\right) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i,t_i^*\right) \tag{2.16}$$

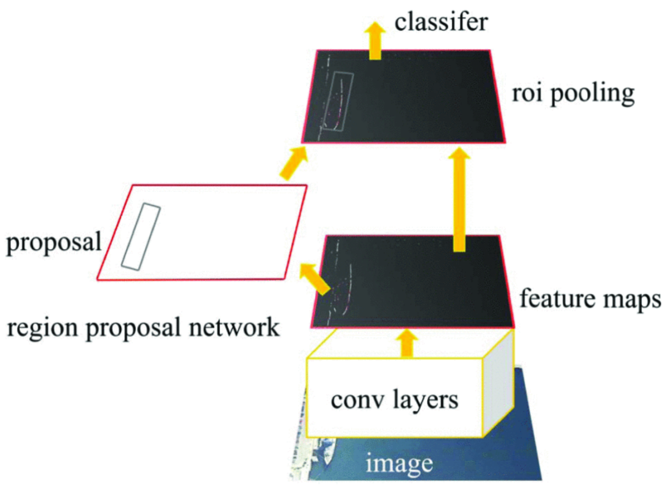


Figure 2.4: The Faster R-CNN design [51].

In the above formula, i reflects the number of different anchors when learning in a mini-batch. p_i is a probability that indicates if the i anchor is a foreground possibility. The value of an anchor is 1 if it is a positive sample; else, it is 0. The anchor's location coordinate is represented by the data in the array. The border coordinates of affirmative sample indicators in the target region are represented by this variable. L_{cls} is used to denote the value of picture block classification loss and the modulation variable t_i^* reflects the value of the acquired mini-batch.

2.3.6 Multi-box Single Shot Detector (SSD)

Multi-box Single Shot Detector (SSD) is a widely used object detection technique. It is faster in general than Faster RCNN [40]. Instead of predicting with the last layer as in traditional networks, it leverages some middle layers in the network called feature maps to recognize objects of various sizes. A single deep learning network can reduce computation time and enhance inference accuracy dramatically.

Moreover, we looked up "Swarm Intelligence" and found a variety of papers. We analyzed these research projects' contemporary designs and solutions utilized during the research.

2.3.7 Swarm Based Algorithms

In the late 80s, computer scientists suggested the tool to identify biological swarm systems in the area of Ai. G. Beni and J. Wang coined the phrase "swarm intelligence" to characterize a set of tactics for operating artificial swarms in the global optimization approach in 1989 [1]. Individual characteristics and their relationships with groups have been used to build algorithms for the analogous mechanism known as "swarm intelligence" [56]. Swarm intelligence ideas nowadays are implemented in a variety of problem domains, including optimization algorithms, route planning, scheduling, structural optimization, and picture analysis [13]. Swarm intelligence approaches such as Particle Swarm Optimization (PSO) [25], Bat Optimization, Grey Wolf Optimization, Ant Colony Optimization (ACO) [4], and Artificial Bee

Colony (ABC) Optimization [54] have all been developed in recent decades.

Some of the most broadly adopted models are:

2.3.8 Ant Colony Optimization (ACO)

M. Dorigo introduced Ant Colony Optimization (ACO) in the late 1980s, and it was initially used to handle discrete optimization problems. ACO is inspired by the social behavior of ant colonies. Without any visual data, a swarm of "nearly blind" ants can work out the shortest route between their food and colony [4]. The basic search strategy used by ACO is as follows [53]:

1. At the beginning, a large number of ants are placed in the area at random. The pheromones on each route have the same initial values, therefore τ_{ij} (0) is equal to τ_0.

2. Furthermore, using the arbitrary proportion principle, the $k-th$ (k=1,2,...,m) ant selects the next place to be relocated to, and its choosing probability is calculated by the equation 2.17

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum\limits_{s \in allowed_k} [\tau_{is}(t)]^{\alpha}[\eta_{is}(t)]^{\beta}}, & j \in allowed_k \\ 0, & \text{Otherwise.} \end{cases} \tag{2.17}$$

2.3.9 The Artificial Bee Colony (ABC)

The Artificial Bee Colony (ABC) algorithm is motivated by honey bees' biological function. In a bee colony, three groups are responsible for deciding the profitability of a food supply: employed, observers, and explorer bees. Information on closeness, quality, food quantity, and food extraction efficiency is used to calculate food profitability. The bee colony algorithm contains phases for food source selection. Throughout the working bee stage, each bee saves its position with its assigned food supply and searches for a better nectar source in a neighboring area. They also return to their colonies and perform a waggle dance to communicate critical information about their food supply. Onlooker honeybees examine the buzz movement of scout bees and pick engaged bees with adequate nectar supply during the Onlooker bee stage. Once data is obtained, spectator bees become engaged bees to continue activities. The hired bees are transformed into observer bees. Finally, scout bees become hired bees for their given food supply once they have discovered it [38]. A parameter known as a limit controls the food source selection. After a certain number of measurements, there is no change in fitness. The recruited bee abandons the food and becomes a scout bee in search of a new alternative [54].

After the beginning phase, ABC goes through a cycle of employed bee, spectator bee, and scout bee phases until the closure target is achieved. The ABC algorithm approach's primary framework for solving an optimization problem is as follows [54]: In the Initial phase food sources are being set using the following equation in 2.18

$$x_{ij} = x_j^{min} + rand(0,1)(x_j^{max} - x_j^{min}) \tag{2.18}$$

Here, the dimension of the vector x is denoted by vector j, rand(0,1) is a random factor that has been predefined and the lower and upper bound of the j^{th} parameter are the x_j^{min} and x_j^{max} . After this phase, the given equation 2.19 is used to assess fitness, where the value of the cost function at solution i is f_i.

$$Fitness_i = \frac{1}{1 + f_i} \tag{2.19}$$

During the employed bee phase, a new solution is formed by selecting a partner at random to identify a better food supply. The current solution and the value of the partner should be different and it is specified as follows in equation 2.20

$$x_{new}^j = x_j + rand(-1,1)(x_j - x_p^j) \tag{2.20}$$

If the value obtained is not within the predefined lower and upper bounds using the following formula, a bounding method is required.

$$x_{new}^j = lb \quad \text{if } x_{new}^j < lb \tag{2.21}$$

$$x_{new}^j = ub \quad \text{if } x_{new}^j > ub. \tag{2.22}$$

In the equation 2.21 and 2.22, x_{new}^j is the newly developed solution along with the current solution x_j. Besides, the partner solution is the x_p^j. x_j^{new} is then used to calculate new fitness. After that, a biased selection is used to see if the new fitness $Fitness_{new}$ has a higher value than the prior fitness answer $Fitness_i$. If the answer is yes, the trial number will be reset to zero. If this is not the case, the trial counter will be raised by one.
Furthermore, onlooker bees fly towards sources of food to evaluate the quantity of nectar available. The observer bee does statistical computations in this phase to choose a preferable food source using the formula in 2.23

$$p_i = \frac{Fitness_i}{\sum_{j=1}^{n} Fitness_j}. \tag{2.23}$$

Finally, the optimum outcome will be remembered before the scout bee phase, and solutions with a trial counter higher than the threshold will be deleted. Only one alternative will go through this phase if its attempt is bigger than its threshold, while the other solution will not.

2.3.10 Artificial Fish Swarm (AFS)

Artificial Fish Swarm is one kind of swarm intelligence model that was proposed in 2002 by Li XiaoLei. The fundamental idea of this algorithm is to replicate fish behaviors like foraging, swarming, and chasing by conducting a local search for individuals to come up with an optimum solution. It requires fewer parameter adjustments for which AFS can achieve a faster convergence speed. The movement to the central point of the "visual scope" characterizes swarm behavior in the AFS algorithm [56]. In terms of flexibility, data redundancy, faster speed, and concurrency, the artificial fish swarm technique offers various features that help it tackle the optimization issue clearly. Machine learning, data clustering, and picture segmentation are some of the challenges where AFS has been used to solve. This algorithm uses artificial intelligence to create artificial fish and look for the best value throughout issues.

$$X_j = X_i + rand() \times Step \times \frac{X_j - X_i}{||X_j - X_i||} \tag{2.24}$$

The above equation is used to update the position. For this update process, a point indicated by X_j, needs to be chosen in this artificial fish's visual perception. Here, the fishes are symbolized by points in optimization problems with the position of one artificial fish. X= (x_1, x_2,..., x_n) is used to express the position of one artificial fish, whereas X_i is the current position of this fish displayed [33].

2.4 Reviewed Findings

HRNet is the state-of-the-art neural network for detecting facial landmarks and pose estimation. As a result, HRNet is less useful for our research objective. Moreover, HRNet maintains a high-resolution procedure throughout the whole process, which can increase the time for our computational operations. GCN uses graph data. It can be implemented in fields such as molecular science, traffic predicting, social networking, and other areas where data has a distinct graph structure. So, we excluded GCN for our research purpose as our research does not provide any graph data. Furthermore, LSTMs take longer to train. Again having a memory block requires more memory to train, which is not time efficient. LSTM is usually used to process and make predictions given sequences of data. On the other hand, Convolution Neural Network is used to exploit spatial correlation in data, and it works effectively on images. One disadvantage of Faster R-CNN is that the RPN is trained using a single photo to retrieve all anchors in the mini-batch of size 256. The network might take a long time to attain convergence since all data from a single picture may be linked as their features are similar. In terms of swarm intelligence, ACO does not use any visual data. So we excluded ACO for our research purpose. AFS have many drawbacks namely a greater temporal complication, a shortage of balance between global and local search, and the inability to profit from group members' previous moves for future motions.

In 2016, Bousetouane et al. developed a CNN-based detection pipeline for wide-area maritime surveillance using Annapolis Maritime Surveillance Dataset. The resulting pipeline provides excellent accuracy on a complicated maritime vessel dataset while also ten times faster than the central Fast-R-CNN pipeline [22]. The entire system based on the Fast-R-CNN architecture [18] is efficiently using a weaker HOG-based object sensor, the maritime fine-tuned VGG16 CNN model using ROIPooling layer built in the Fast-R-CNN architecture, and a simple confidence scheme in the three phases. It is also optimized for near-real-time performance, with multi-core concurrent region proposals and GPGPU processing for CNN evaluation. The pipeline readily replaces selective search using a HOG detector for area recommendation in the Fast-R-CNN model. Overall, Sliding window and Fast-R-CNN togetherly achieve a mean average precision (mAP) of 61.55% [22].

In 2018, Wang conducted an experiment in the MatLab 2017 (a) environment in a normal desktop machine where the utilized strategy is to discover the places using UAV-captured photos to implement maritime SAR using pre-processed (HG) and CNN imagery (HV). The training set for this experiment consists of 1500 images, 500 positive samples, and 1000 negative samples [36]. Hypothesis generation (HG) is used to capture the high-resolution color images of the sea by the UAV camera to choose the detecting region before the likely location of the victims can be determined. Then the rectangle finally frames the candidate region from the original image from the extracted and denoised binary images. After that, to verify the person in that area, hypothesis verification (HV) is done using the CNN. The findings imply that this technique outperforms artificial vision search in terms of overall performance, saving time in preparation, reducing rescue period, and being resource independent. The search efficiency of the UAV cluster operation, in particular, will improve considerably. CNN classification is a supervised learning method in which victims are identified on a map utilizing simple attributes from UAV photographs and then inspected in these areas [36].

In 2019, Moosbauer et al. established a new benchmark in object recognition with deep learning in case of marine contexts, and used Singapore Maritime Dataset (SMD) to test Faster and Mask R-CNN for object tracking [41]. Both DCNNs are pre-trained on COCO and ImageNet, which both contain a large number of maritime items and use ResNet-101 as their backbone. A completely Mask R-CNN with a deactivated sample categorization branch performed best in the VIS and NIR spectrums, having f-scores of 0.875 as well as 0.877 [41]. According to the results, the Mask R-CNN is a strong and well-suited DCNN at object recognition.

This leads to the conclusion that a basic but reliable architecture must be created solely from the domain of deep learning, especially from Convolutional Neural Network (CNN) allow us to assist in the early detection of humans on the open water.

Chapter 3

Methodology

This chapter is divided into two sections. The first part describes the deep learning model that has been used to detect humans in water bodies, and the second part briefly portrays Swarm Intelligence (SI) that has been used to search large areas effectively.

3.1 Deep Learning using CNN

This section briefly describes the idea of our selected dataset. All the information is separated into parts via sections of this chapter. At first, we dive into the various data collection and processing techniques. Then, we briefly describe the architecture of our selected model. Later, all the methods for enhancing the performance of the specific model during training are mentioned in detail. Our proposed methodology can be visualized in figure 3.1.

3.1.1 Data Collection

In the dataset selection process, we agreed to use the SeaDronesSee dataset [62]. SeaDronesSee is a large-scale dataset with a goal to help develop effective systems for SAR (Search and Rescue Operations) using UAVs (Unmanned Aerial Vehicles) in maritime scenarios. This dataset is a benchmark initiative from the Avalon (Aerial and vision-based assistance system for real-time object detection) project to fill the void of the proper dataset for deep-sea search and rescue operations using unmanned aerial vehicles and robust computer vision systems. This dataset consists of three tracks which are multi-object tracking, object detection, and single-object tracking. Here each track has its own dataset. From here, we chose the object detection dataset to achieve our expected goal. In our object detection dataset, there are 1796 testing images, 5630 train images, and 859 validation images.

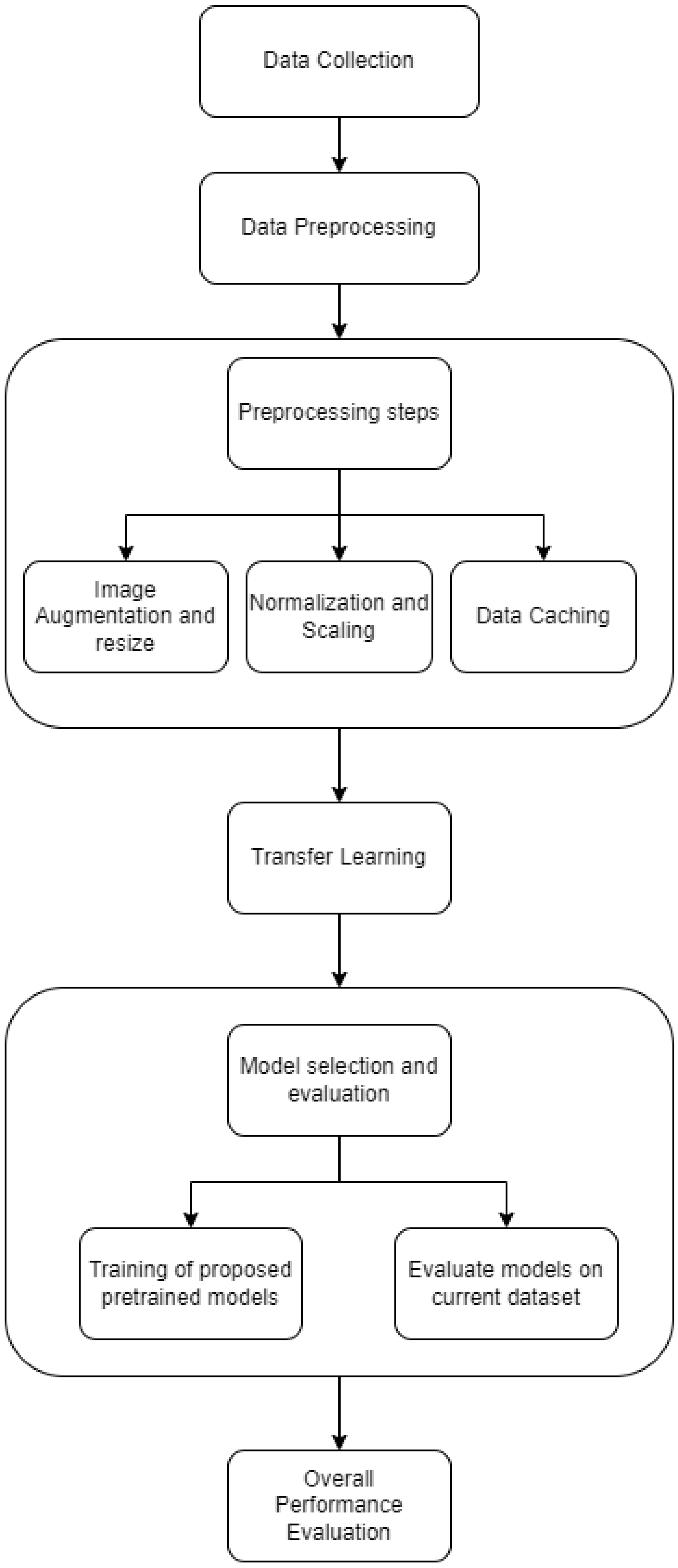

Data Collection
Data Preprocessing
Preprocessing steps
Image Augmentation and resize
Normalization and Scaling
Data Caching
Transfer Learning
Model selection and evaluation
Training of proposed pretrained models
Evaluate models on current dataset
Overall Performance Evaluation

Figure 3.1: Proposed Methodology Diagram

3.1.2 Data Preprocessing

To avoid unwanted results and correctly implement the dataset into our pre-trained model, we used the following preprocessing methods:

1. **Image Augmentation:** Data augmentation is a process by which we can insert more variation in our already existing dataset. This process helps us to increase the number of data. Image augmentation is one of the popular forms of data augmentation. We have introduced image augmentation techniques such as randomly zooming, increasing clarity, increasing brightness by a random amount, flipping or rotating images, etc. We hoped to introduce more variation in the dataset to make it robust.

2. **Image resize:** Our images in the dataset vary in size. There are a lot of high-resolution images; hence, it takes a lot of time to train. This process is responsible for establishing a base size for all images to feed our model. We expected to lessen the time of training through this process.

3. **Image Normalization:** Image normalization is a common image processing strategy to alter the intensity value of pixels. The fundamental goal of the normalization process is to convert an image into a range of pixel values between 0 and 1. This is accomplished by dividing all pixel values by 255, the maximum pixel value.

4. **Scaling Bounding Box:** We used the sigmoid activation function in order to achieve the predicted output values. The training and validation images' bounding box coordinates were scaled. The ground truth bounding box coordinates were scaled in relation to the original image's dimension.

5. **Data caching:** Usually, image preprocessing takes a lot of time, and we need to avoid running all preprocessing steps over and over again. So, after all the preprocessing was done, we saved the data in our device storage so that we could start training models much more quickly and efficiently.

3.1.3 Data Selection

In our dataset, there are a total of 6 categories such as swimmer, a life jacket-wearing swimmer, floater, person on the boat, passenger on a boat wearing a life jacket as well as boat were annotated in this dataset, which will be useful for recognizing humans in maritime scenarios [57]. We included three categories for our research purpose, which include boat, floater, and swimmer. We choose unique images for each category. In the validation dataset, from the image, floater included 194 images, swimmer included 111 images, and boat 118 images. From the training set, floaters included 678 images, swimmers included 546 images, and boats included 457 images. All the datasets are in COCO format [43]. We took annotations for bounding boxes from the COCO format dataset, along with image IDs and class categories from the annotation.

3.1.4 Model Architecture

This section discusses our proposed Convolutional Neural Networks [19] model that we used to detect humans on open water bodies. The Convolutional Neural Network (CNN) or ConvNets is a type of artificial neural network that specializes in identifying and understanding patterns. It has so far been most widely utilized for image analysis. The multiplication of two images produces matrices, which can be used to identify characteristics and patterns in the image. This capacity to recognize patterns is what makes CNN so useful for image analysis. CNN features hidden layers known as convolutional layers that distinguish it from a typical multi-layer perceptron or MLP that goes through all the steps of our model construction below. In order to construct our model, we have used different pre-trained models such as VGG16, ResNet50v2, Xception, InceptionV3, and MobileNetV2. We used these models as the first part of our architecture. Later, we added the bounding box architecture part and class label architecture part after going through the flattening of the layer. In the case of the bounding box network part, we used four dense layers and two dropout layers in order to control the amount of values that needed to be transferred to the next layer. On the other hand, in the class label part, we used four dense layers. Moreover, we used three dropout layers to control the flow of information from one layer to the next. A detailed description of each layer is given below:

1. **Convolutional Layer:** CNN's initial layer is the convolutional layer. It uses different filters to extract information from the input image during convolution processes. After receiving input, the convolutional layer modifies it and sends the transformed input to the next layer. A Convolutional operation is a name for this transformation. The convolutional layer contains various parameters and hyper-parameters such as filters, kernels, and K. Then, these filters compare images to distinguish the similarities and differences among them, which helps extract features. The primary goal of a convolutional layer is to extract and identify high-level features. In order to preserve the relationship between each pixel by learning the image aspect, we have performed this operation.

2. **Pooling Layer:** The pooling layer is employed to lessen computational costs by shrinking the dimension of the convolved feature map. This layer speeds up processing by preventing overfitting and lowering memory usage. It reduces layer connections and functions independently in each feature map. Depending on the methodologies, there are different pooling procedures.

3. **Flatten Layer:** The flatten layer alters the pooled feature map into a single column. It is an obligation to flatten the pooled outputs so that they can be sent to the next FC layer.

4. **Fully Connected Layer:** The weights and biases are stored in the Fully Connected (FC) layer, which joins the neurons from two separate layers. After flattening the input image from the previous layer , it is passed to the fully connected layer and then these layers pass on the data to other layers like a feed-forward neural network while assigning weights to each part of the data on the way.

5. **Activation Functions:** The activation function is a significant aspect of the Convolutional Neural Network model that specifies which model data should be forwarded and returned at the network's end. Nonlinearity is introduced to the network by this function. The Softmax, Sigmoid, ReLU, and tanH activation functions are the most commonly utilized. In our CNN, we have mainly used Three activation functions. These are given below:

   a **Sigmoid:** The exact function employed in the logistic regression classification process is the sigmoid [46] activation function, often known as the logistic function. The sigmoid function accepts any real value as input and returns a value between 0 and 1. When the input is larger, the output value will be nearer to 1.0. When the input is lesser, the output will be closer to 0.0. The following is how the sigmoid activation function is calculated:

   $$f(x) = \frac{1.0}{(1.0 + e^{-x})} \tag{3.1}$$

   In the equation 3.1, e is a mathematical constant that is the base of the natural logarithm.

   b **Softmax:** Softmax function provides a vector of given values adding up to 1.0 [34], which allows a winner-take-all function to produce a probability like output that can be read as class membership probabilities. It is a simplified version of the argmax function, with a vector of real values as input and a vector of the same length with values of 1.0 as output, comparable to probability. The softmax function is calculated as below:

   $$f(x) = \frac{e^x}{\Sigma(e^x)} \tag{3.2}$$

   In the equation 3.2, x is denoted as a vector of outputs, and the natural logarithm's base is denoted as a constant e.

   c **ReLU:** Rectifier Linear Unit (ReLU) [29] function is easy to implement and works by taking only the positive parts of its arguments. It is also useful to overcome limitations like clearing the gradients which prevents training deep models. Following equation is used to calculate ReLU function:

   $$f(x) = max(0, x) \tag{3.3}$$

   The equation 3.3 implies that if the input data (x) is negative, the value is returned as 0.0; else, it is returned as the value.

3.1.5 Learning Enhancements

This section describes the learning enhancement techniques used during the training of our model. Each of these processes is described below:

1. **Optimization:** Optimization is a mathematical method that allows algorithms to resolve well-structured optimization issues and analyze those algorithms, allowing the most effective answer to be depicted in a numerically well-defined manner. This algorithm's motive is to minimize the objective function, also referred to as the loss or cost function, which is used to distinguish between anticipated and expected data. Adam [16] is a first-order optimization algorithm in which the learning rate for each parameter is variable. Adam is a stochastic objective function algorithm based on adaptive estimations of lower-order moments, and its algorithm is based on first-order gradients. We utilized AdaMax for our model, which is an extension of Adam's gradient descent algorithm designed to speed up the optimization process. It's an augmentation of the Gradient Descent Optimization algorithm.

2. **Transfer Learning:** A technique known as transfer learning involves applying a previously learned model to a new situation. Instead of beginning from scratch, transfer learning is utilized when the knowledge of an already trained machine learning model is transferred to a new one, with the prior network's weights automatically changed to the new one. Transfer learning is vital because it saves training time, increases neural network performance, and requires little input. We employed VGG16 [20] , ResNet50V2 [60], InceptionV3 [26], Xception [27], and MobileNetV2 [55] in our model, all popular pre-trained machine learning models.

3.2 Swarm Intelligence

Swarm Intelligence is an artificial intelligence concept of science that consists of numerous independent individuals that are self-organized and accountable for their contributions to solving the challenge.

3.2.1 Algorithm Selection

Our research has explored three algorithms (Particle Swarm Optimization, Bat Algorithm, Grey Wolf Optimization) for their effective learning techniques: social, cognitive, and naive learning. These three representative swarm intelligence algorithms are briefly described below:

1. **PSO Algorithm:** Particle Swarm Optimization (PSO) is a popular swarm intelligence approach for solving non-linear continuous optimization problems. However, lately, it has been used in several practical, real-world applications. The PSO method starts with a random particle cluster and then modifies the iterations to find the best solution. At each sampling interval, the two values are adjusted for every particle. These two are called best values. Here, one is called the optimal position, which is the best-derived solution. The other is a global best position, representing the best value generated across the whole

swarm. The particle position and velocity can be updated after determining the above two best values. The position x_i^{t+1} of the particle is derived by multiplying its velocity v_i^{t+1} by its present position [25].

Algorithm 1: PSO pseudo-code

```
Set the initial parameters w, r1, r2, c1, and c2.
Generate the initial population of the particles ($x_i$).
Generate the initial velocity of the particles ($v_i$).
while not stop criterion do
    for each particle do
        Calculate the fitness.
        Calculate the best position up to now (P$_i$).
    end
    Determine the best particle (P$_g$)
    for each particle do
        Update the velocity.
        Update the position.
    end
    Find the best particle.
end
```

2. **Bat Optimization Algorithm :** The Bat Algorithm (BA) is a modern swarm intelligence optimization technique. The echolocation behavior of bats inspires this method. Microbats' echolocation skill is remarkable since they can find their prey and differentiate between different species of insects even in complete darkness. This method follows three idealized rules for simplicity based on behavior analogy like other optimization algorithms. Firstly, to perceive distance and environments, all bats use echolocation. Moreover, when bats search for their target by flying with a velocity using a fixed frequency fmin, they may automatically change the wavelength or frequency of their radiated pulses and the rate of pulse emission r   [0, 1] depending on how close they are to their target. Each bat's position x_i, velocity v_i, frequency f_i, loudness A_i, fluctuating wavelength λ, and emission pulse rate r_i are all described in BA. Lastly, the range of loudness is assumed to be from high (positive) A_0 to a low (constant) A_{min} as the loudness fluctuates in various ways.
This algorithm also takes into account some simplifications. For example, no ray tracing is not employed to estimate the time delay and 3-D topography as it is more computationally intensive in multidimensional scenarios. Besides, higher frequencies have shorter wavelengths and travel a shorter distance. So, the frequency is restricted to the range [0, f_{max}], and the rate of pulse emission is a threshold between [0, 1], where 0 implies no pulse emission, and 1 means the highest rate of pulse emission [9].

The pseudo-code below represents the basic steps of the Bat Algorithm (BA) based on these approximations and idealizations:

Algorithm 2: BA pseudo-code

```
Objective function f(x), $x = (x1, ..., xd)^T$.
Initialize the bat population $x_i (i = 1, 2, ..., n)$ and $v_i$.
Define pulse frequency $f_i$ at $x_i$.
Initialize pulse rates $r_i$ and the loudness $A_i$.
while $t <$ Max number of iterations do
    Generate new solutions by adjusting frequency
    And update velocities and locations/solutions
    for each particle do
        Calculate the fitness.
        Calculate the best position up to now (P$_i$).
    end
    if rand $> r_i$ then
        Select a solution among the best solutions
        Generate a local solution around the selected best solution
    end
    Generate a new solution by flying randomly
    if rand $< A_i$ and $f(x_i) < f(x)$ then
        Accept the new solutions
        Increase $r_i$ and reduce $A_i$
    end
    Rank the bats and find the current best $x*$
    end
```

In the pseudo-code, x∗ is the global best information acquired so far, and rand is a random number uniformly distributed on [0, 1]. The pulse emission rate (r_i) and loudness (A_i) are used in theory to vary the probability of the converging operation. A naive learning method is implemented when the condition rand $> r_i$ is satisfied [28].

3. **Grey Wolf Optimization (GWO):** The GWO algorithm is a unique swarm intelligence approach modeled after the natural leadership structure and hunting mechanism of grey wolves. Four sorts of grey wolves such as alpha, beta, delta, and omega, are used for replicating the leadership structure. Furthermore, the three basic hunting processes are implemented: looking for prey, enclosing prey, and attacking prey. Grey wolves prefer to live in packs and always move together toward their prey. Besides, they have a fairly rigid social dominating structure. Due to the rigid social dominance hierarchy in place, most members operate as followers (ω) of the dominators (α, $\beta, and\ \delta$), who rank first, second, and third, respectively, in terms of performance. There is another category in wolves named subordinates who are below the dominators and control the omega [17].

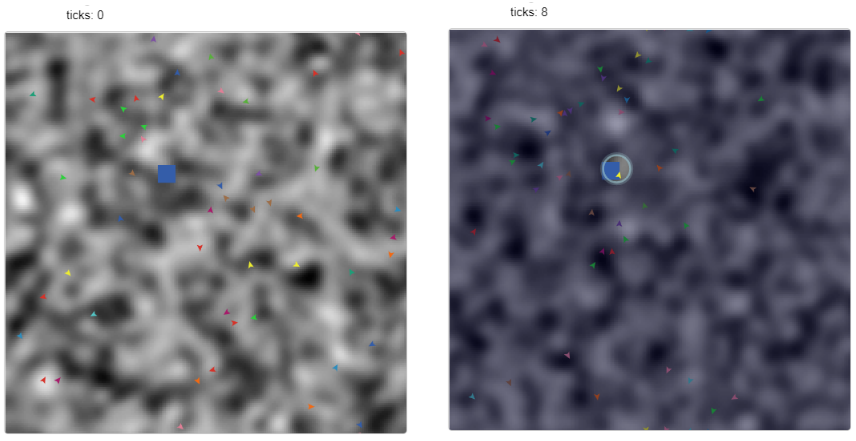
   The pseudo-code of the grey wolf optimization (GWO) algorithm is given below:

Algorithm 3: GWO pseudo-code

```
Initialize the grey wolf population x$_i (i = 1, 2, ..., n)$
Initialize a, A, and C
Calculate the fitness of each search agent
$x_\alpha$= the best search agent
$x_\beta$ =the second-best search agent
$x_\delta$ =the third best search agent
while (t < Max number of iterations) do
    for each search agent do
        Update the position of the current search agent
    end
    Update a, A, and C
    Calculate the fitness of all search agents
    Update $x_\alpha$, $x_\beta$, and $x_\delta$
    t=t+1
end
return $x_\alpha$
```

3.2.2 Swarm Intelligence Simulation Model

In this Research, We used Netlogo [2] to simulate the behavior of the Particle Swarm Optimization Algorithm, Bat Optimization Algorithm, and the Grey Wolf Optimization Algorithm. All of these algorithms are Metaheuristic Algorithms. Metaheuristic Algorithms aim to solve optimization problems. In our research, our target is to optimally explore a large area and to find a target in that area.
As we know, all the optimization algorithms like PSO, BA, and GWO have two sections in their working procedure which are Exploration and Exploitation. Our goal is to use the exploration behavior of these algorithms to find a target in a huge area. In our simulation model, we took fifty drones in a square area where they moved around in random patterns, which are derived from the noted swarm behavior algorithms.



Simulation Initialized Target Detected

Figure 3.2: Simulation in Netlogo

We designed the simulation by placing a fitness value in the range [0,1] in the patches, and we declared a certain area with a fitness 1 (Blue marked region in figure 3.2), which resembles our target (Swimmer and Floater) in the huge ocean.

We randomly generated the initial positions of the drones and their next position, and in some cases, the velocities were calculated according to each algorithm. We ran the simulation five times for each algorithm. We then keep track of the Ticks, which is a model-independent time unit in Netlogo Models. It is used instead of Hours, Minutes, and Seconds because it is standardized for all models and computers. This ensures that even if a processing unit runs slower than others, the value of the ticks is always constant.

In the simulations, we stopped the iteration only when one of the drones was able to find a target in figure 3.2. This allowed us to calculate the exact time unit needed to find a target using the drones in the algorithm.
We modified the algorithms to simulate the exploration process for a certain scenario which is finding a target in a huge arena, in our case, detecting a person in a maritime scenario.

Chapter 4

Implementation and Result Analysis of CNN Models

Previously we discussed the data collecting and preprocessing method, as well as a quick summary of our suggested model. In this segment, we explain how we constructed the network and compared our results to those from other models such as VGG16 [20], ResNet50v2 [60], InceptionV3 [26], MobileNetv2 [55] and Xception [27]. Figure 4.1 contains a brief overview of our fully obstructed model.

4.1 Implementation of the baseline model

This section describes our implementation process of five chosen models in python using libraries such as Keras and Tensorflow. In order to select the best model for our dataset, we selected five pre-trained models such as VGG16, ResNet50v2, MobileNetv2, Xception, and InceptionV3. Then, after flattening the part, we extended our model into two parts. Bounding Box Network and Class Label Network.

1. **Bounding Box Network :**
   This part is added to predict the bounding box coordinates of an object which is, in our case, floater, swimmer, and boat. We added three dense layers and two dropout functions to control the information from the previous layer. We used relu and sigmoid activation functions here.

2. **Class Label Network :**
   On the other hand, this network part is included in order to predict the class of detected objects. We added three dropout layers and four dense layers. For the activation function, we used softmax as it is a multiclass problem and ReLU function.

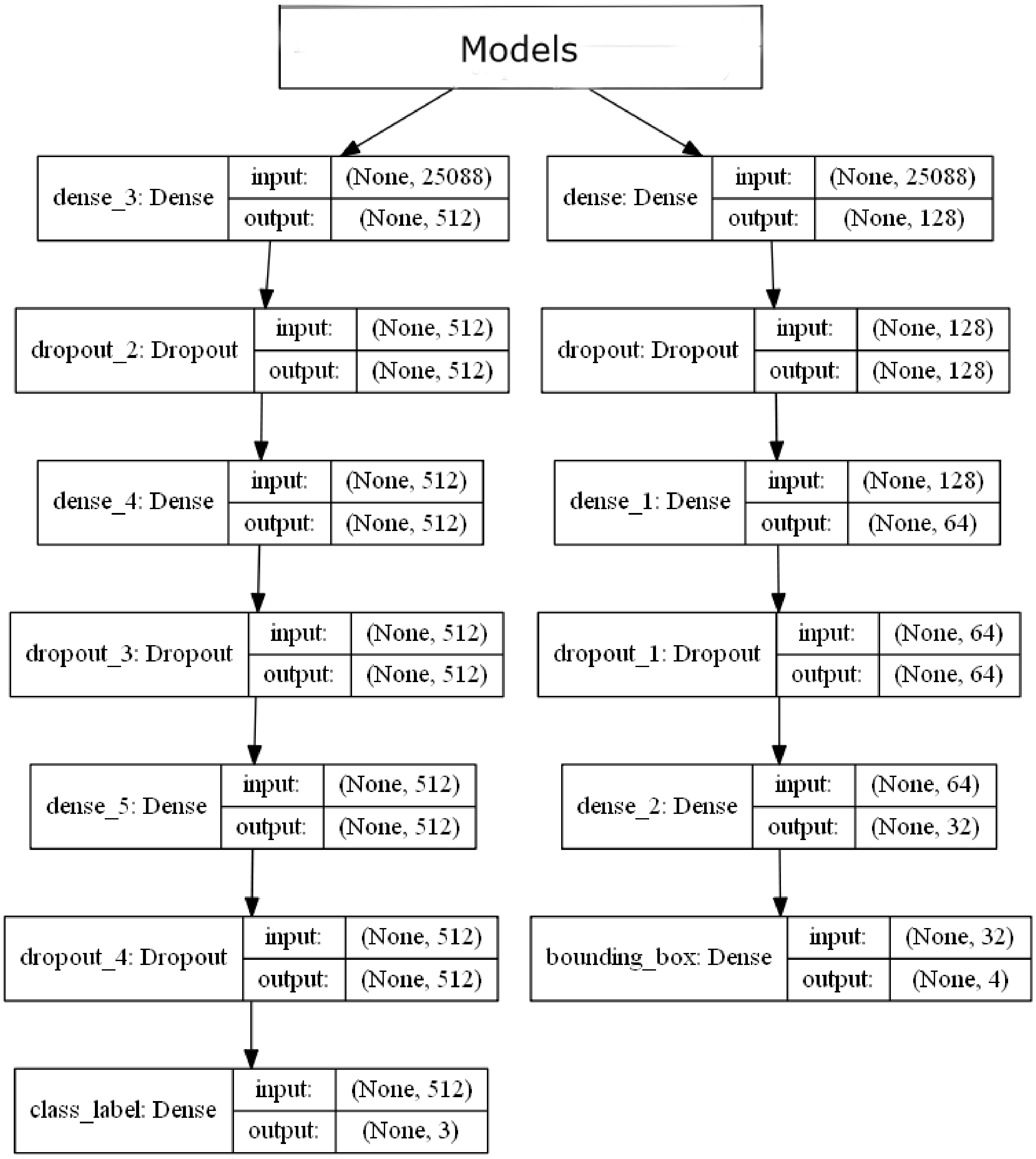


Figure 4.1: Model Architecture

4.2 Selected Models' Performance

This section contains the overall performance of all models. We have fitted our proposed pre-trained models to our dataset and calculated the accuracy and validation accuracy of both class label and bounding box parts to evaluate the models. Comparisons of results are given below in Table 4.1 and Table 4.2

After running all models on our dataset for 100 epochs, we managed to obtain our desired results for comparison. Now, for the class label part, surprisingly, all models

Model Name	Accuracy	Validation Accuracy
VGG16	99.15%	97.7%
ResNet50v2	99.01%	97.61%
InceptionV3	99.05%	97.15%
Xception	99.76%	97.16%
MobileNetv2	99.75%	98.11%

Table 4.1: Class Label Accuracy

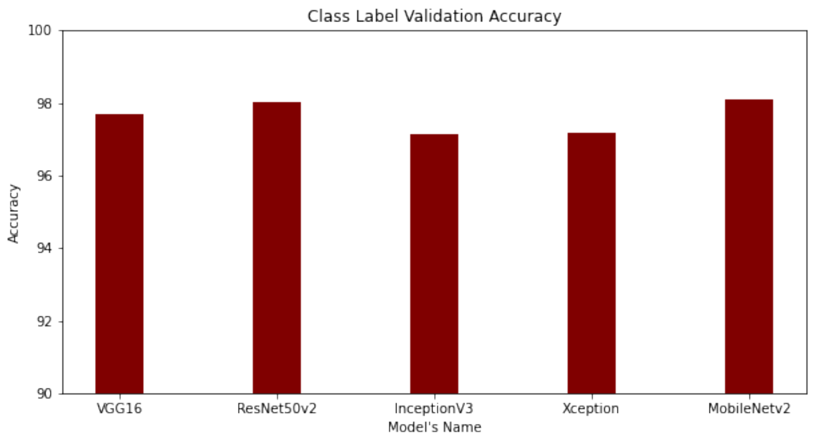


Figure 4.2: Class Label Validation Accuracy

showed promising results. After running the models, we managed to obtain the best result from the MobileNetV2 model, where accuracy is 99.75% and Validation accuracy is 98.11% and got the worst result from the ResNetv2 model, where accuracy is 99.01% and validation accuracy is 97.61%. From Xception, we got an accuracy result of 99.76% and validation accuracy of 97.16%. Lastly, from the VGG16 model, we got 99.15% accuracy and 97.7% validation accuracy, and from the InceptionV3 model, we got 99.05% accuracy and 97.15% validation accuracy. This “class label validation accuracy” data is further represented through the bar chart in Figure 4.2

Model Name	Accuracy	Validation Accuracy	Mean IoU score
VGG16	88.74%	86.05%	0.62
ResNet50v2	69.90%	72.10%	0.40
InceptionV3	78.25%	78.35%	0.48
Xception	81.23%	81.56%	0.51
MobileNetv2	72.05%	75.65%	0.45

Table 4.2: Bounding Box Results

On the other hand, in the bounding box part, we managed to get the best result from the VGG16 model with an accuracy of 88.74% and validation accuracy of 86.05%, while we got the worst result from ResNet50v2, which gave us an accuracy of 69.90% and validation accuracy of 72.10%. Moreover, we got a 78.25% accuracy result and 78.35% validation accuracy from InceptionV3 and obtained 81.23% accuracy from Xception and 81.56% validation accuracy, which is the second-best result. Lastly,

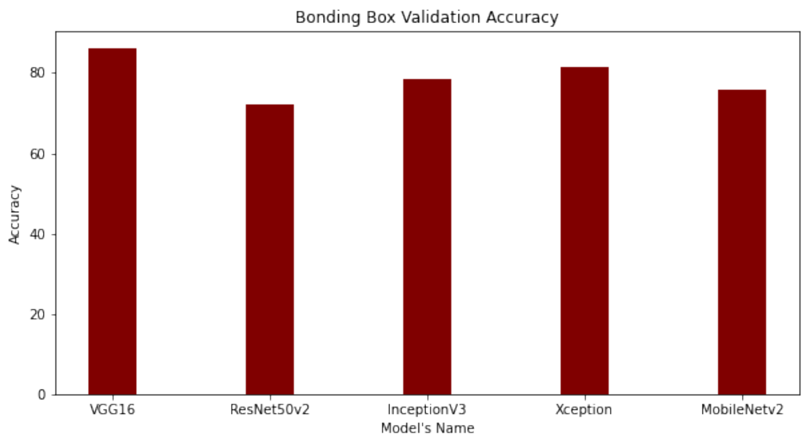


Figure 4.3: Bounding Box Validation Accuracy

we got 72.05% accuracy and 75.65% validation accuracy from MobileNetv2. This “bounding box validation accuracy” data is represented through the bar chart in Figure 4.3.

4.3 Model Comparison

During the class table part, although Xception showed promising results, we chose to implement VGG16 as the preferred model. The reason behind this decision is in the bounding box results, VGG16 showed the best result with a mean IoU score of 0.62 along with the best accuracy as shown in figure 4.4. While on the other hand, Xception showed the second-best result with a Mean IoU score of 0.51. Moreover, in the class label results, the performance difference between other models is very low.

As a result, we decided that VGG16 is the best performing model as it has good class label validation accuracy while not compromising on the bounding box scores as seen in other models.

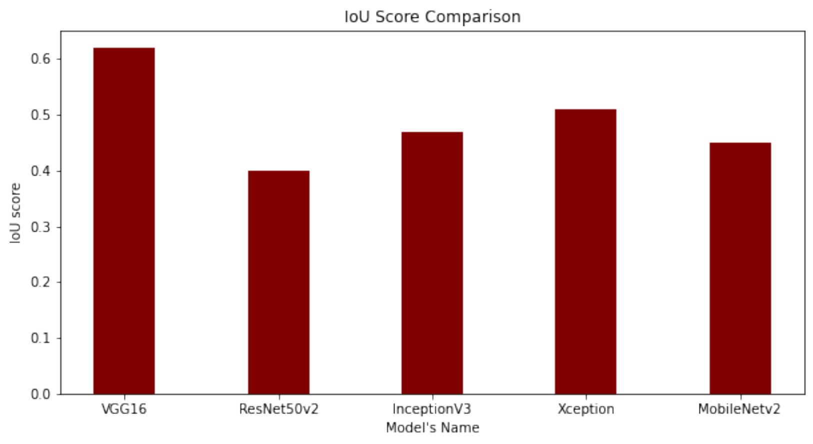


Figure 4.4: IoU Result Graph

4.4 Discussion

From Figure 4.5 and 4.6, we can observe that the predicted bounding box from VGG16 with the highest IoU score almost matches the actual bounding box region. On the other hand, the second-best model Xception, although it has better results than the other three models, the predicted bounding box is not as good as VGG16. The actual bounding box area is not covered as well as VGG16 by the predicted bounding box area using Xception. Generally, IoU scores less than 0.5 is undesirable, and the Xception model IoU is close to that threshold. On the other hand, VGG16 showed better results with 0.62, which is comparatively a better result. Overall, VGG16, although not the very best result in terms of class label portion, is, however, the best performing model in the bounding box portion.

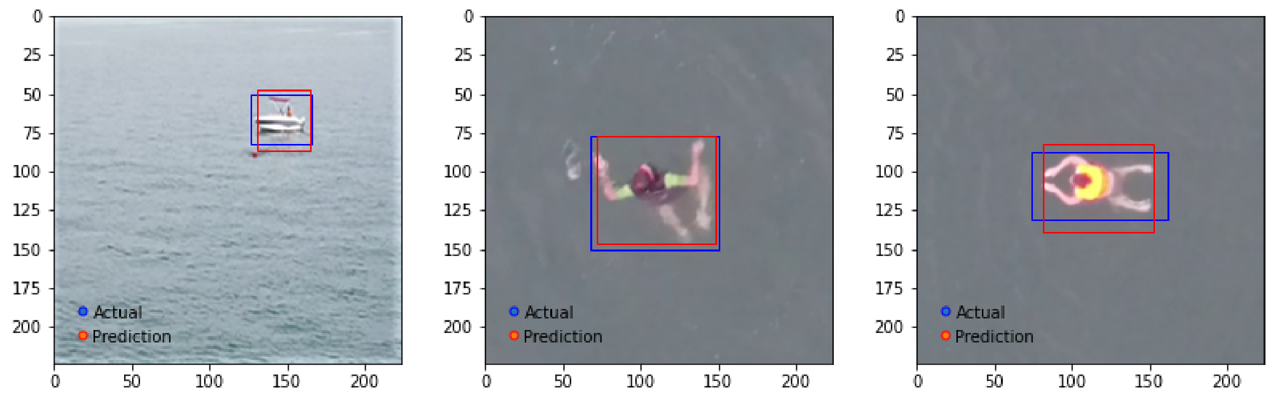


Figure 4.5: Difference between actual and predicted bounding box (VGG16)

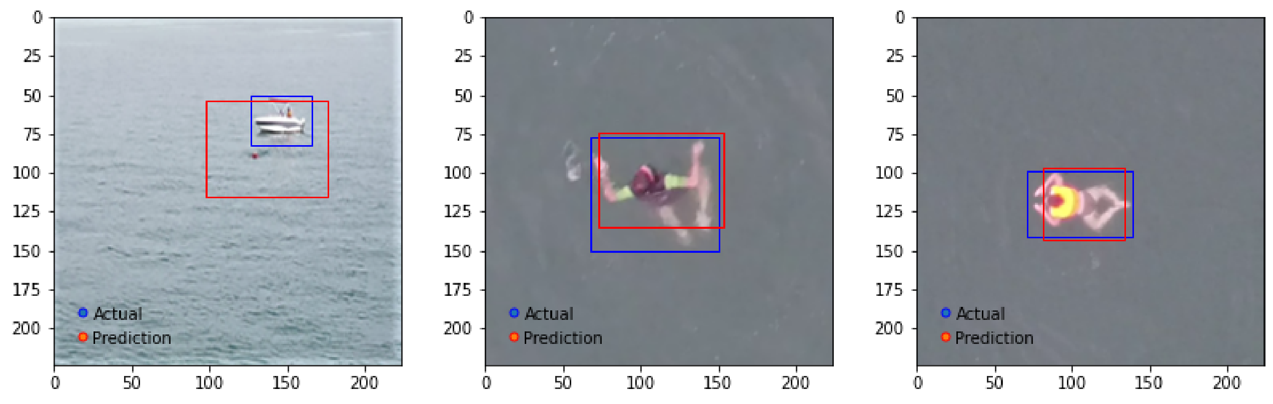


Figure 4.6: Difference between actual and predicted bounding box (Xception)

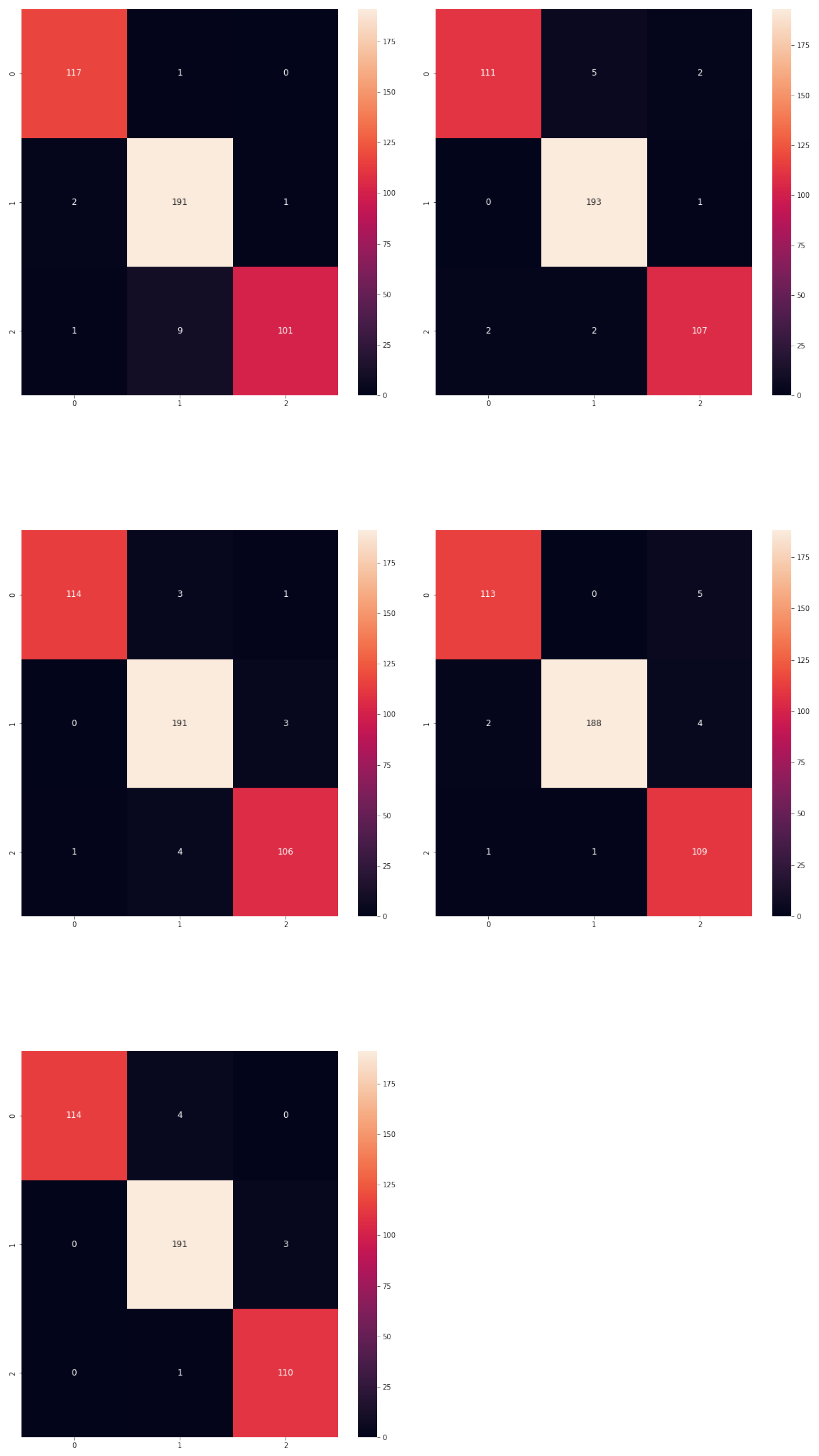
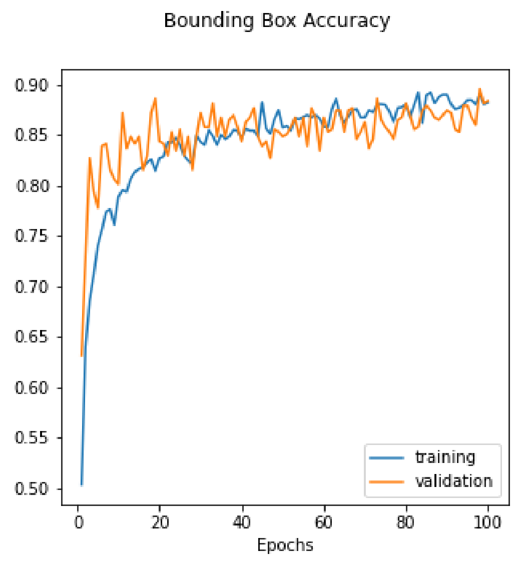
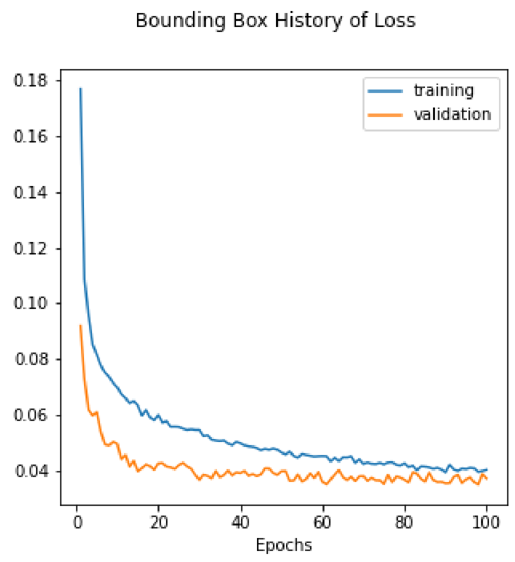


Figure 4.7: Confusion Matrix of VGG16, Xception, InceptionV3, ResNetv2, MobileNetV2 respectively

In Fig 4.7, we can observe the confusion matrix of all model's class label parts. In the confusion matrix, 0 indicates boat, 1 indicates floater, and 2 indicates swimmer. The first matrix of fig 4.7 contains the confusion matrix of the VGG16 model class label part; we can see that this model was able to identify 117 boats, 191 floaters, and 101 swimmers correctly from the validation dataset. However, this model was not able to correctly classify 14 images. The second figure shows the Xception model class label part's confusion matrix. This model was able to identify 111 boats, 193 floaters, and 107 swimmers correctly from the validation dataset. However, this model could not correctly identify the class of 12 images. The third figure bears the confusion matrix of the InceptionV3 model. While it could not detect classes in 12 images correctly, the InceptionV3 model was able to detect 114 boats, 191 floaters, and 106 swimmers. On the other hand, the fourth matrix of fig 4.7 contains the confusion matrix of ResNet50v2, and it was able to detect 113 boats, 188 swimmers, and 106 swimmers correctly. This model could not identify the class of objects correctly in 13 images. Lastly, the last confusion matrix contains the confusion matrix of MobileNet50v2; it detected 114 boats, 191 swimmers, and 110 swimmers correctly. However, this model could not correctly identify the class of objects in 8 images.



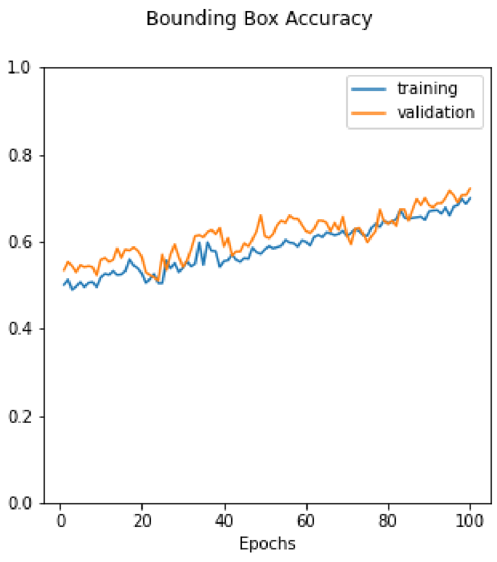
(a) VGG16 Accuracy



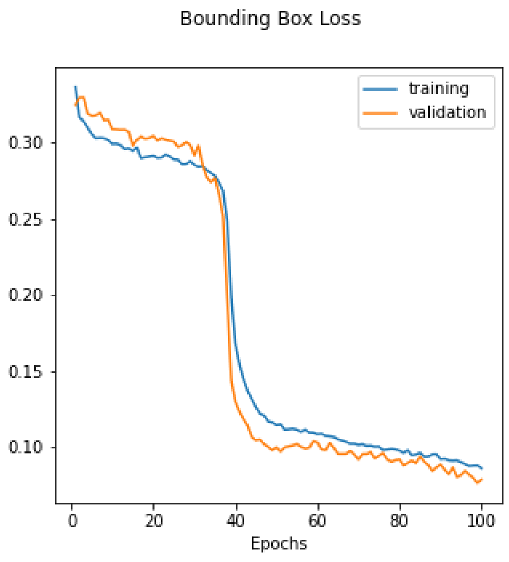
(b) VGG16 Loss

Figure 4.8: VGG16 Bouding Box Metrics

In figure 4.8, we can observe the VGG16 accuracy graph where the accuracy has reached 99.15%.On the other hand, the loss graph indicates how much data was lost which in this case almost reached 0.055.



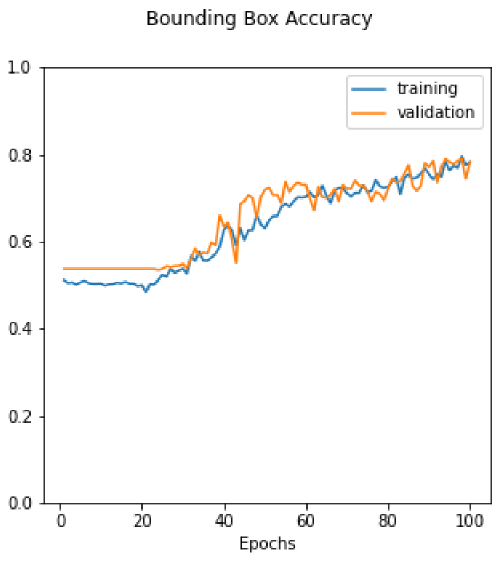
(a) ResNet50v2 Accuracy



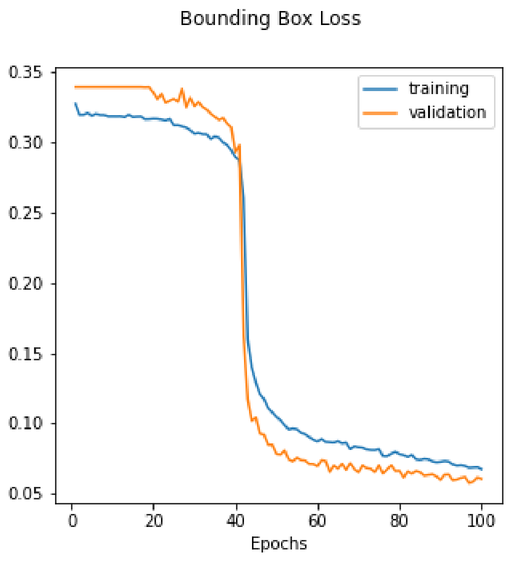
(b) ResNet50v2 Loss

Figure 4.9: ResNet50v2 Bouding Box Metrics

The ResNet50v2 accuracy graph (figure 4.9) shows that the accuracy has reached 69.90 and before 40 epochs the loss value was rather high. Later, the loss value gradually reached 0.08.



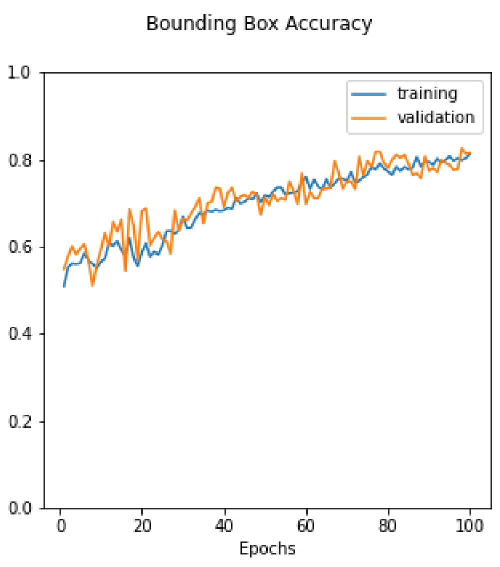
(a) InceptionV3 Accuracy



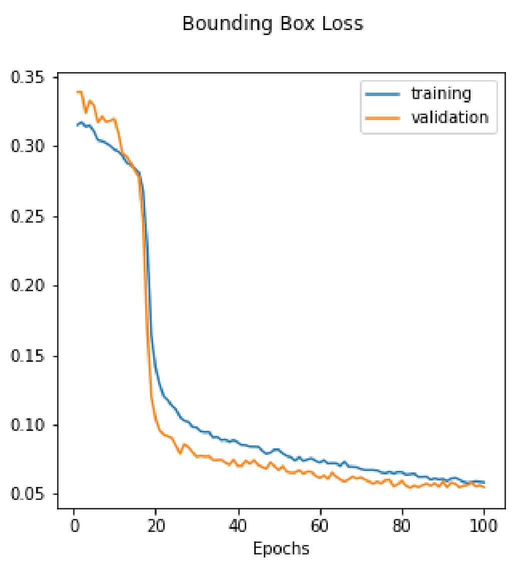
(b) InceptionV3 Loss

Figure 4.10: InceptionV3 Bouding Box Metrics

The accuracy graph for InceptionV3 (figure 4.10) reveals that the accuracy has reached 78.25%, while the loss value was quite high before 45 epochs. The loss value eventually increased to 0.07.



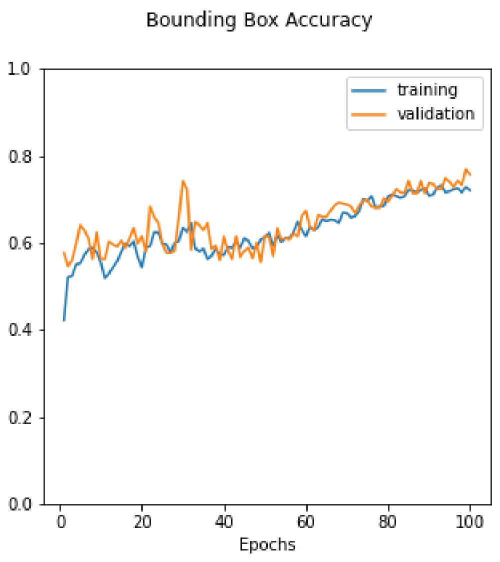
(a) Xception Accuracy



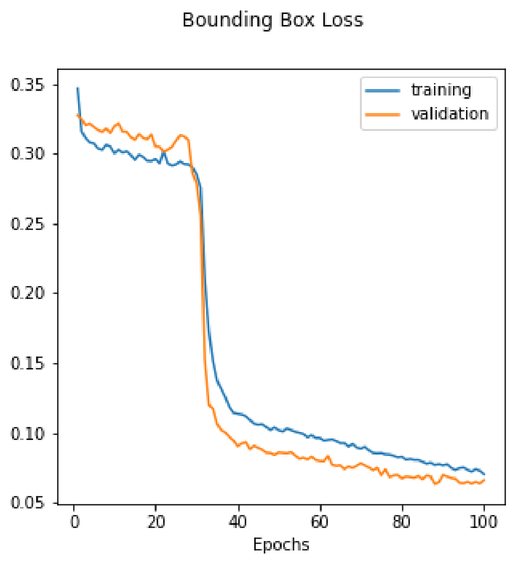
(b) Xception Loss

Figure 4.11: Xception Bouding Box Metrics

Xception's accuracy graph (figure 4.11) shows that the accuracy has reached 81.23%. After 20 epochs, the loss value decreased from high value to low value which is 0.06.



MobileNetV2 Accuracy



(b) MobileNetV2 Loss

Figure 4.12: MobileNetV2 Bouding Box Metrics

The accuracy graph in MobileNetv2 (figure 4.12) reveals that it has reached 72.05%. The loss value started reducing after 25 epochs and reached 0.07.

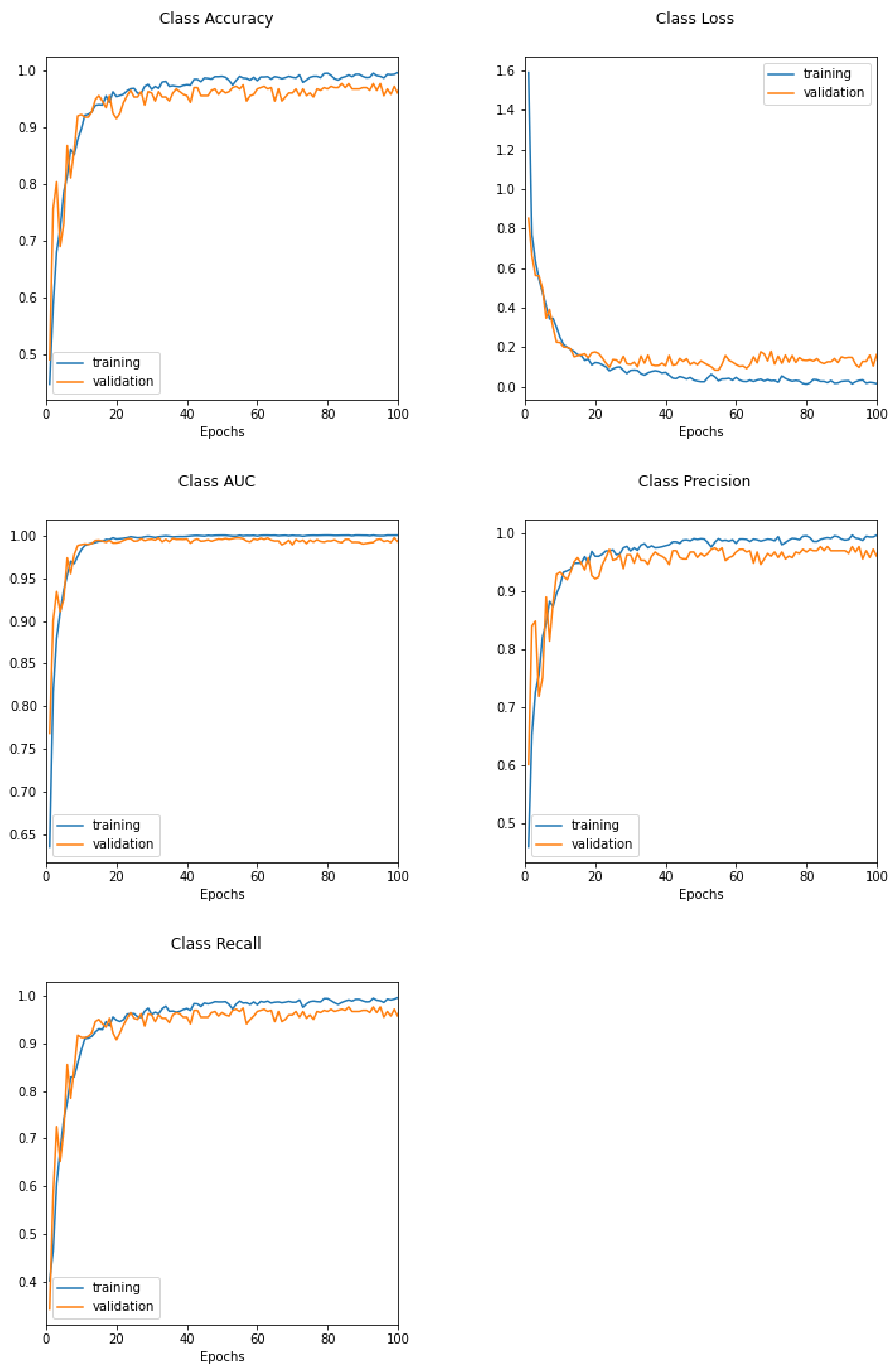


Figure 4.13: VGG16 class Label Accuracy, Loss, AUC, Precision and Recall respectively

In figure 4.13, the first graph indicates the class label accuracy graph of the VGG16 model. We can see that the accuracy reached 99.15%, and the second graph shows the loss value of the model, which almost got 0, and the validation loss value of 0.17. In the third graph, the AUC value almost reached 1, which indicates most of the predictions were true positives. Moreover, the fourth graph shows the precision value that almost got a value of 1, and the validation value is 0.95. This indicates maximum positive predictions do indeed belong to the positive class. Lastly, recall also gave us satisfactory results as it reached a value of 1, and a validation recall value is 0.94, indicating maximum positives were recalled.

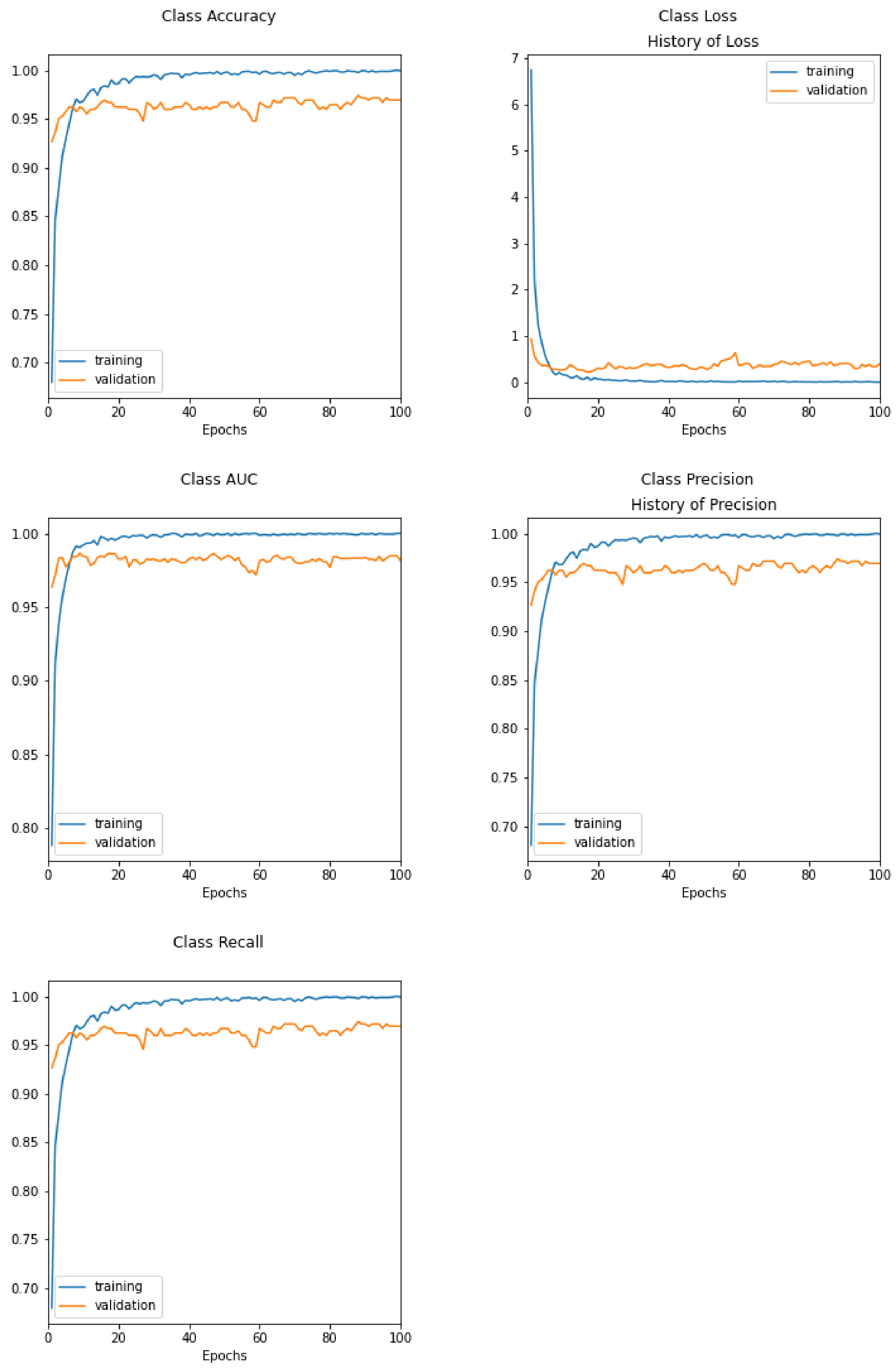


Figure 4.14: ResNet50v2 class Label Accuracy, Loss, AUC, Precision and Recall respectively

The first graph in figure 4.14 shows the ResNet50v2 model's class label accuracy graph. The accuracy was 99.01%, and the second graph displays the model's loss value, which was virtually zero, and the validation loss value of 0.5. The AUC value in the third graph is almost 1, and the validation AUC value is 0.975. Furthermore, the fourth graph depicts the precision value, which is approximately equal to 1, and the validation value, which is 0.96. This shows that maximal positive predictions are definitely in the positive category. Finally, recall yielded promising results, with a value of 1 and a validation recall value of 0.96, suggesting that maximum positives were recalled.

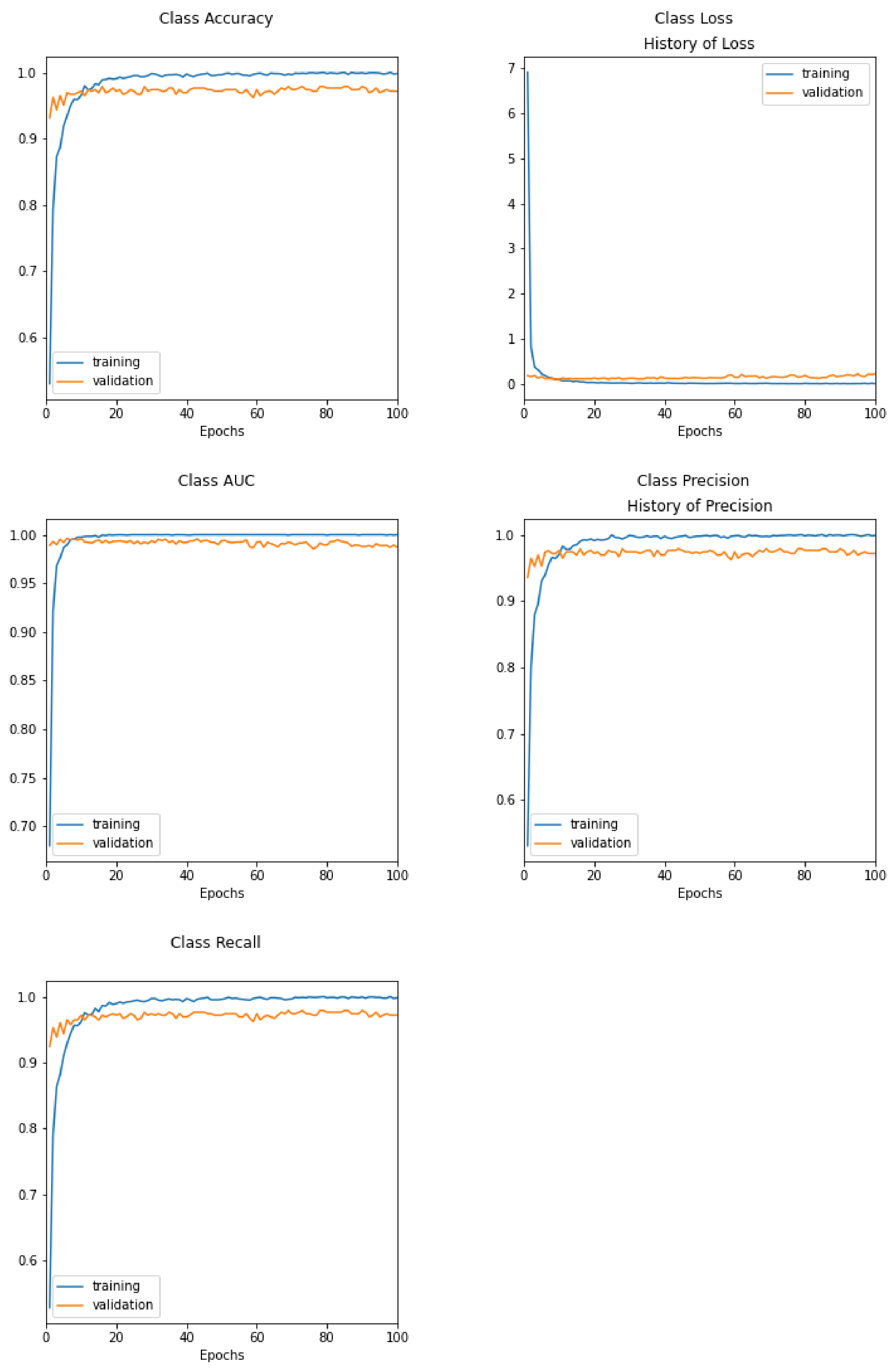


Figure 4.15: InceptionV3 class Label Accuracy, Loss, AUC, Precision and Recall respectively

The class label accuracy graph for the InceptionV3 model is shown in the first graph in figure 4.15. The accuracy was 99.05%, and the second graph shows the model's loss value, which is very good as the loss value almost reached 0. In the third graph, the AUC value is nearly 1, whereas the validation AUC value is 0.98. The fourth graph also shows the precision value, which is close to 1, as well as the validation value, which is 0.96. This demonstrates that maximally positive predictions are unquestionably positive. Finally, recall produced positive results, with a value of 1 and a validation recall value of 0.95, indicating that nearly all positives were recalled.

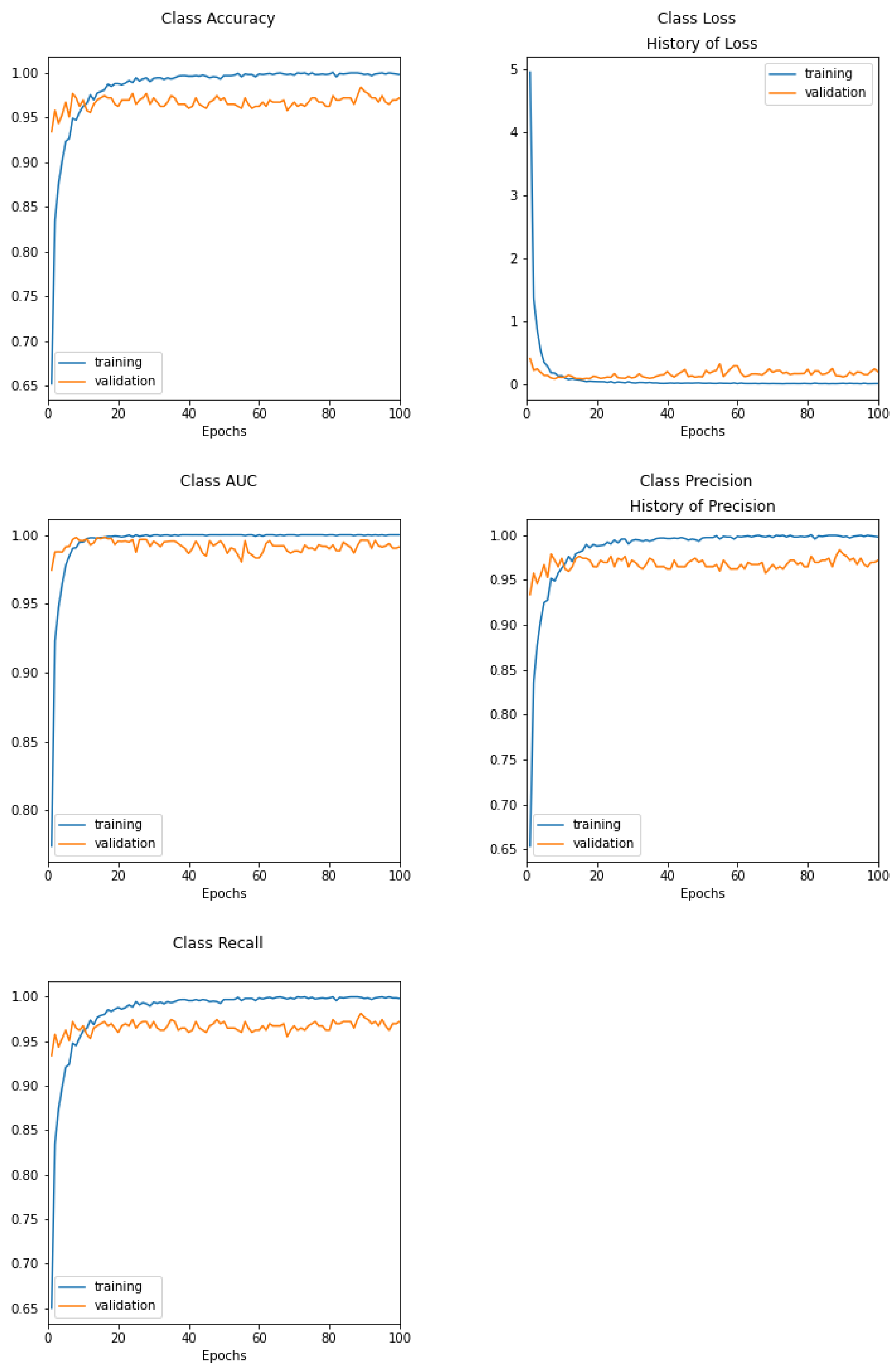


Figure 4.16: Xception class Label Accuracy, Loss, AUC, Precision and Recall respectively

The first graph in figure 4.16 shows the class label accuracy graph for the Xception model. The model's accuracy was 99.76%, and the second graph indicates the model's loss value, which was nearly zero, which is excellent. The AUC value in the third graph is almost 1, while the validation AUC value is 0.98. The precision value, which is close to 1, and the validation value, which is 0.96, are also shown in the fourth graph. This indicates that predictions that are maximum positives are undoubtedly positive. Finally, recall yielded promising results, suggesting that almost all positives were recalled, with a value of 1 and a validation recall value of 0.96.

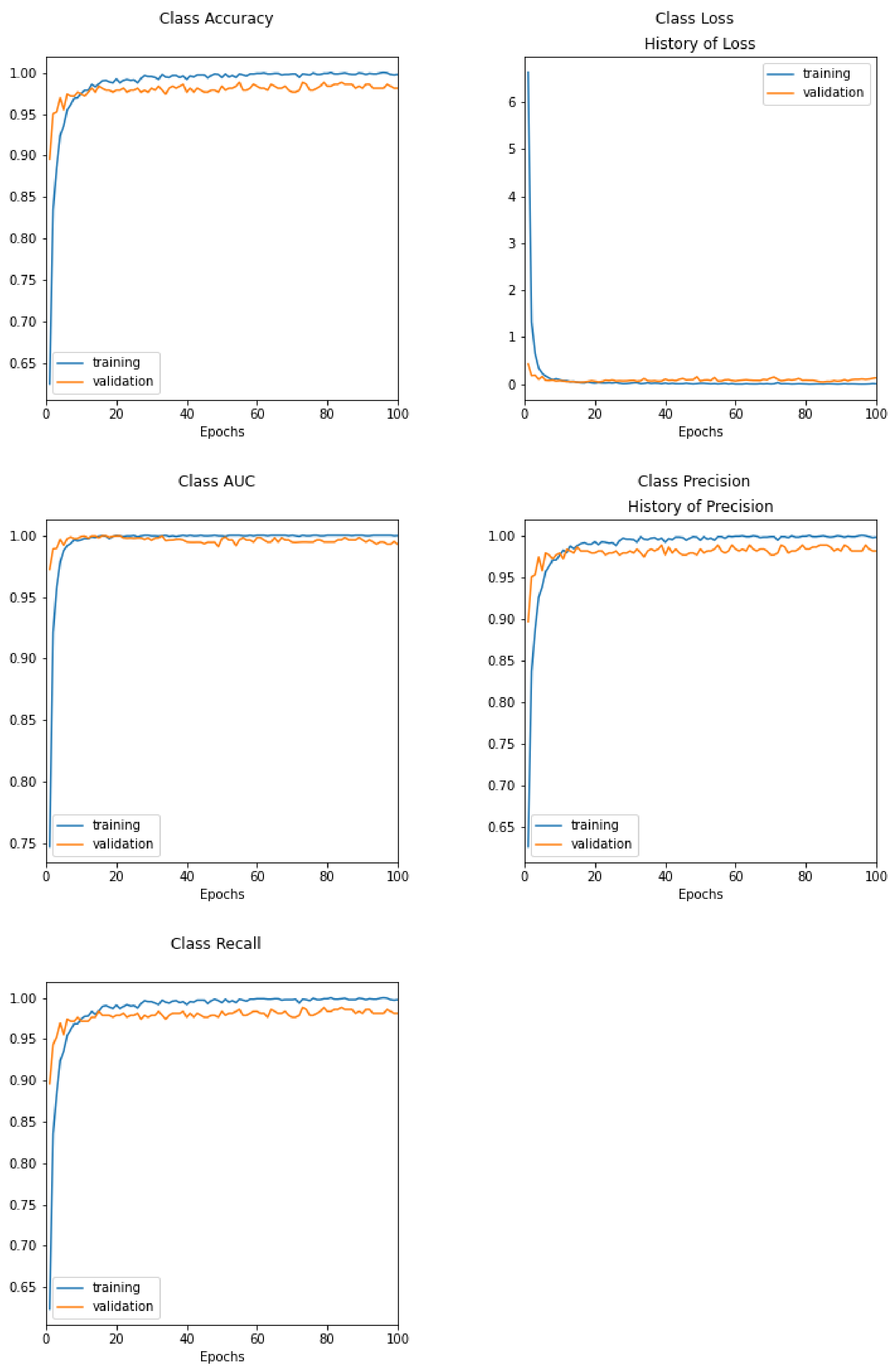


Figure 4.17: MobileNetv2 class Label Accuracy, Loss, AUC, Precision and Recall respectively

The class label accuracy graph for the MobileNet50v2 model is represented in the first graph in figure 4.17. The model's accuracy was 99.75%, and the model's loss value was virtually 0, which is great. The validation AUC value is 0.98, but the AUC value in the third graph is 0.99, and validity value is 0.97. This shows that maximal positive predictions are almost always correct. Finally, with a value of 0.99 and a validation recall value of 0.97, recall produced good results, indicating that all maximum positives were recalled.

Chapter 5

Implementation and Result Analysis of Swarm Intelligence Algorithm

As discussed earlier, we are using three swarm algorithms and comparing their results in order to find the best-performing algorithm. In this segment, we will also be discussing the work process and mathematical equations of the Particle Swarm Algorithm, Bat Algorithm, and Grey-Wolf algorithm.

5.1 Implementation of Swarm Intelligence Algorithms in Simulation

For the selected algorithms, we distributed random fitness values ranging from [0,1] across the arena. Then we normalized the values by using the minimum value, maximum value, and a float of 0.9599. To normalize, we used a function,

$$calculated_{fitness} = 0.9599 * \frac{(current_{fitness} - minmum_{fitness})}{(maximum_{fitness} - minimum_{fitness})} \tag{5.1}$$

We ran the iteration till we got the best-defined fitness value which is 1.

In PSO Algorithm:

The drones were initialized with variables including, velocity (v_i), position (x_i), fitness value (val), personal-best-position (p_{Best}). Initially, the position (x_i) was randomly selected within the range of the simulation area, and velocity (v_i) was initialized with random values distributed on [0,1]. Also, the $calculated_{fitness}$ value for that position was taken as the “val” of that particular drone, and the position is considered as the p_{Best}. The Global Best is selected by taking the best fitness value found among all the drones in each iteration, and we denote it as “g-val” and the position of that drone is considered as g_{Best}(Global Best Position)

In each iteration, for every particle, we updated the velocity according to 5.2

$$v_{i+1} = w \times v_i + c1 \times r1 \times (p_{Best} - x_i) + c2 \times r \times (g_{Best} - x_i) \tag{5.2}$$

Here, v_i is a two-dimensional vector where i represents each dimension. c1 is a constant which resembles the cognitive learning factor which initiates the movement of a particle towards its own personal best position (p_{Best}). c2 is another constant which resembles the social learning factor as it implies the movement of a particle towards the global best position (g_{Best}). x_i is the current position of that drone which is also a two-dimensional vector, and w is the particle inertia that helps us change the particle's motion, where the value of w=1 means moving in a straight line to always move toward the best position found.

Then according the equation no 5.3 we updated the positions-

$$x_{i+1} = x_i + v_{i+1} \tag{5.3}$$

This gives us the next position where our drone should take a random walk and in simulation our drone moves toward that position.

In Bat Optimization Algorithm:

The drones were initialized with variables including, velocity (v_i), position (x_i), fitness value (val), personal-best-p on (p_{Best}), loudness (A), pulse imitation rate (r) , frequency (f), initial pulse emission rate (initial-r). Initially, velocity (v_i) was initialized with random variables within [0,1], and the position (x_i) was randomly selected within the range of the simulation area.

We calculated the frequency (f) according to the following equation 5.4

$$f = f_{min} + (f_{max} - f_{min}) * rand \tag{5.4}$$

Here, here in our simulation, we considered f_{max} as 2 and f_{min} as 0, and rand is random variable at a range [0,1]

We initialized the pulse emission rate according to 5.5

$$r = rand * (r_{max} - r_{min}) \tag{5.5}$$

Here r_{max} is 1 and r_{min} is 0, and rand is variable with a random value within the range [0,1]

According to the equation 5.6 we also initiated loudness (A),

$$A = r * A_{max} - A_{min} + 1 \tag{5.6}$$

Here r is a random value in range [1,7], A_{max} is 2, and A_{min} is 1 and initially, we saved the pulse emission rate in the variable initial-r, which is required in the later part of the simulation process.

Also, for each drone, the $calculated_{fitness}$ value for that position was taken as the "val" and the position is considered as the p_{Best}. The Global Best is selected by taking the best value found among all the drones in each iteration and which is denoted as "g-val" and the position of that drone is considered as g_{Best}(Global Best Position)

In each iteration, for every particle, we adjusted the frequency according to 5.4.

We updated the velocity (v_i) accordingly,

$$v_i = (v_i + p_{Best} - g_{Best}) * f \tag{5.7}$$

Here v_i is a 2-dimensional vector where it represents each dimension, the current best position of the drone is denoted as p_{Best}, and the global best position vector is denoted as g_{Best}. And the f is the newly adjusted frequency (f) of that particular drone.

Then we checked if current loudness (A) is lesser than a random variable rand in the range of [0,1] or not and also if the fitness value (val) in the current position is lesser than the previous val or not. If yes, then we decrease the loudness (A) and increase the pulse emission rate (r) according to the following equations,

$$A = \alpha * A \quad ; \ \alpha \ = \ 0.5 \tag{5.8}$$

$$r = initial - r * (1 - e^{0.9} * n) \ \ ; \ n \ is \ iteration \ number. \tag{5.9}$$

Then according the equation no 5.3 we updated the positions-

$$x_{i+1} = x_i + v_{i+1} \tag{5.10}$$

This gives us the next position where our drone should take a random walk and enables it to explore within the arena of our simulation.

In Grey Wolf Optimization Algorithm:

At first The drones were initialized with variables including, position (x_i), fitness value (val), personal-best-position (p_{Best}). The position (x_i) was randomly selected within the range of the simulation area. Also for each drone, $calculated_{fitness}$ value for that position was taken as the "val" and the position is considered as the p_{Best}. The Global Best is selected by taking the best value found among all the drones in each iteration and which is denoted as "g-val" and the position of that drone is considered as g_{Best}(Global Best Position)

Here from the initial calculations we then update the position according to the following equation 5.3

$$x_{i+1} = \sum_{Q \ \epsilon \ \alpha,\beta,\delta} \frac{x_Q}{3} \tag{5.11}$$

$$x_{q+1} = x_q - (a * (2 * r1 - 1) * |2 * r2 * x_q - x_i|) \quad ; Q \ \epsilon \ \alpha, \beta, \delta \tag{5.12}$$

Here $x_\alpha, \ x_\beta \ and \ x_\delta$ denote the top 3 individual drones with the best fitness values (val). a is a weight linearly decreasing from 2 to 0 after each iteration. r1 and r2 are two distinct random variables distributed on [0,1] and x_i is the current position.

According to the updated positions with the fitness value of that position we again find the top three agents or drones with the best fitness values and denote them respectively as $\alpha \ , \beta \ and \ \delta$.
In the simulation, our drone moves to the next position according to the updated position in x_{i+1}.

In every algorithm, in each iteration we check if the fitness value (val) in the updated position is higher than the current val or not. If yes, then we update our fitness value of our drone with the higher val and update our (p_{Best}) to the currently updated position.

We also check if the global fitness value (g-val) is lower than the val found in the updated position or not. If yes, then we update our global fitness value (g-val) with the higher val and update our (g_{Best}) to the currently updated position.

Lastly, in every iteration we check whether the newly found g-val or global val is 1 or not. If it's 1, it means that our drone has found our target and we stop our simulation there. If not, we continue our iterations.

When our iterations stopped, we noted down the Ticks. And generated our results according to that.

5.2 Simulation Result Analysis

For each of the algorithms, we ran the simulation five times. We updated the target area in every simulation for all the algorithms. For example, in simulation 1, every algorithm had to find a specific target area, and in the next simulation, we moved our target and tasked every algorithm to find that specific target again. In this process, we noted down the time spent (ticks) by each algorithm in each simulation in a data table.
Here we can see that in each iteration, the amount of time each algorithm spends to find a target in the simulation area. Table 5.1 is represented in figure 5.1

	Time Spent (Ticks)		
Simulation	PSO	BA	GWO
1	4	55	47
2	15	33	71
3	15	15	96
4	48	103	61
5	20	163	53

Table 5.1: Simulation Results

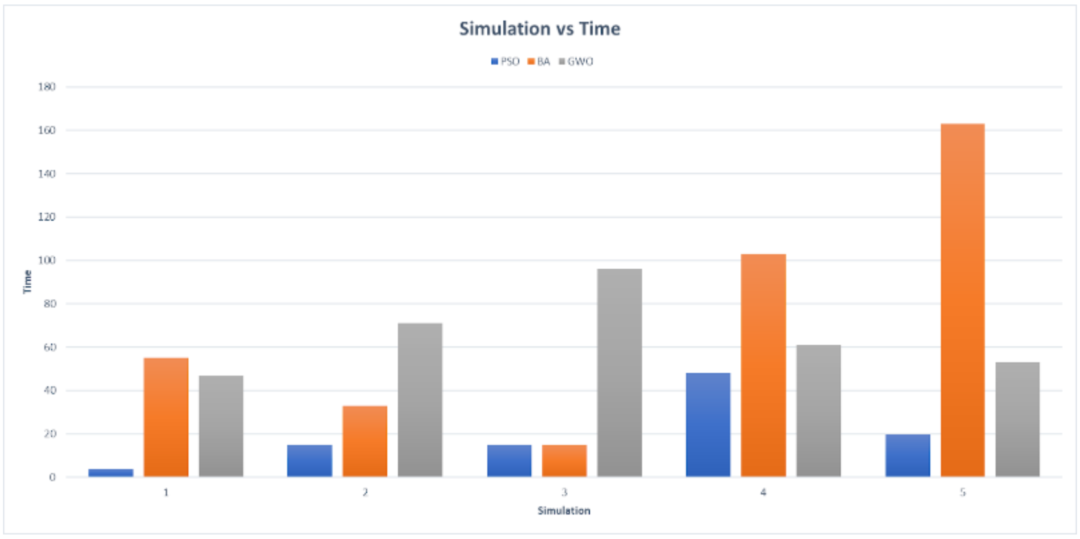


Figure 5.1: Simulation Vs Time

In the bar chart, we can clearly see that in each simulation, PSO takes the shortest time to find a target while GWO and BA take a much higher time in particular cases. Between GWO and BA, we can see that.
In the case of exploring and finding a target in simulations number 1, 4, and 5, BA takes higher time, whereas, in simulations 2 and 3, GWO takes higher time.

5.3 Result Comparison

Here, if we take a look at the bar chart of Average times spent by each algorithm over the course of 5 simulations, we can see in the figure 5.2

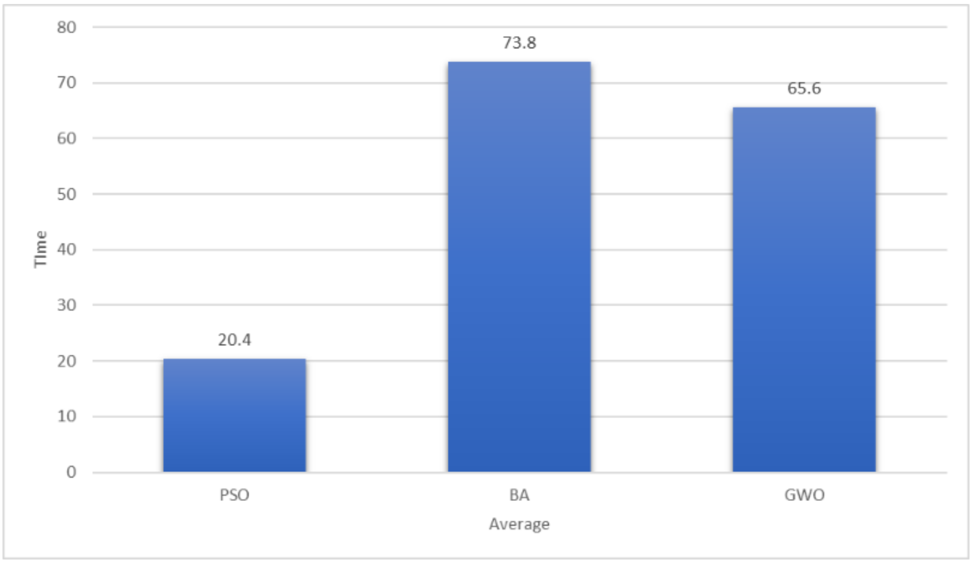


Figure 5.2: Average Time

PSO, on average, takes the lowest time to find a target, and the BA takes the highest time in the same scenario. Even though GWO spends a slightly lower time to detect a target than BA, it is much greater than the average time spent by PSO.

5.4 Discussion

According to the results, we can conclude that among the three different algorithms, PSO promises the best results. It can ensure that even having a unknown target in a huge area, the exploration procedure, which means the update of the positions of the drone in respect to both personal-best and global best allows it to optimally and effectively explore a large area. Thus allowing it to find an unknown target in the shortest time possible.

5.5 Proposed System Architecture

In an emergency situation in a maritime scenario, we will deploy several drones which will locally run our pre-trained model using VGG16 to detect swimmers, floaters, and boats and to work as swarms to efficiently cover a huge area and find

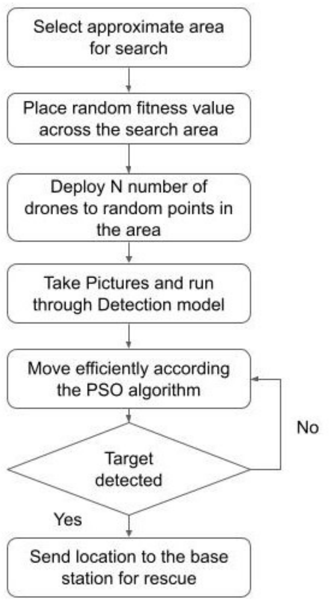

Select approximate area for search
Place random fitness value across the search area
Deploy N number of drones to random points in the area
Take Pictures and run through Detection model
Move efficiently according the PSO algorithm
Target detected
No
Yes
Send location to the base station for rescue

Figure 5.3: System Flow Diagram

a target using the Particle Swarm Optimization (PSO) algorithm. For example, if a ship-wreck occurs in the sea, the process will be as follows,

1. Base Station will select a wide area near the area where the situation occurred.
2. Base Station will randomly place fitness values across the selected area.
3. Then an “N” number of drones will be sent to random points in that area.
4. The drones will start detecting humans by taking pictures at a specific time interval and running through our selected best-performing detection model (VGG16).
5. They will also keep moving according to the modified PSO algorithm.
6. Once one of them finds out a target, the fitness value of that position will be set at 1, and the swarm will move accordingly.
7. As our drone finds a target, it will send the location to the base. As soon as the central unit gets to the specific location, they can start their rescue mission.

The whole system architecture can be further visualized by the flow diagram 5.3

Based on the best swarm intelligence algorithm that we found through simulation result analysis which is Particle Swarm Optimization Algorithm, and the CNN model VGG16, which is selected based on overall accuracy results, we propose the system architecture visualized in figure 5.4

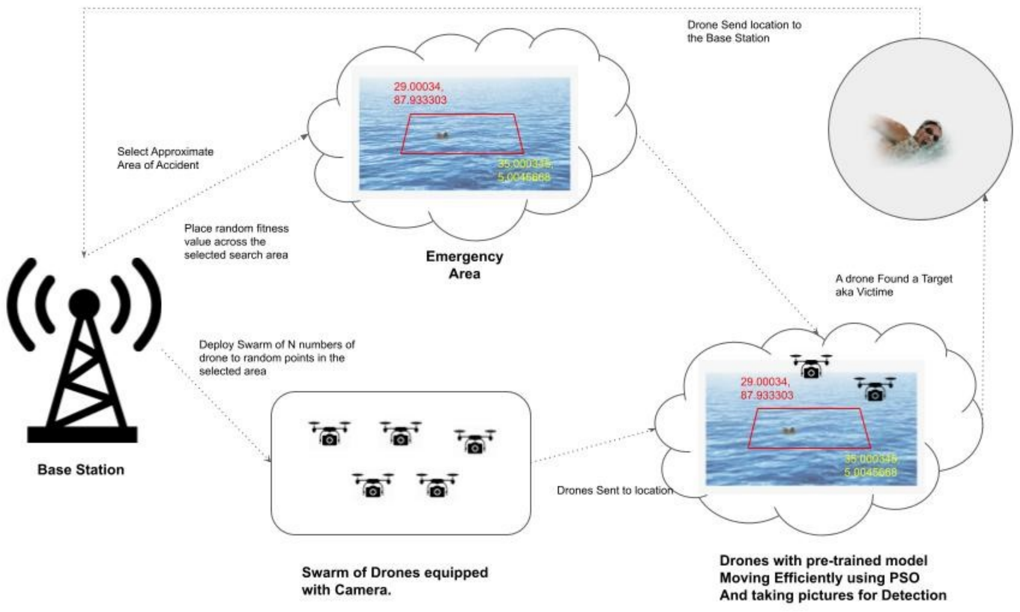

Drone Send location to the Base Station
29.00034, 87.933303
35.000345, 5.0045668
Select Approximate Area of Accident
Place random fitness value across the selected search area
Emergency Area
A drone Found a Target aka Victime
Deploy Swarm of N numbers of drone to random points in the selected area
Base Station
Drones Sent to location
Swarm of Drones equipped with Camera.
Drones with pre-trained model Moving Efficiently using PSO And taking pictures for Detection

Figure 5.4: Proposed System Architecture

Chapter 6

Conclusion

Surveillance in maritime scenarios using traditional means can either be slow or costly. In today's world, if it is required to search for someone in a short time who got lost due to a natural disaster in open water or due to maritime accidents, one has to choose a helicopter over a boat as boats are far slower and offer a lower range of visibility. However, the cost of using a helicopter is too much to opt for. In the case of maritime accidents, the chance of survival of a victim depends on the time required to rescue him and for that drones are the best option to use. In this research, our goal is to provide a proper solution that will make surveillance in maritime scenarios more effective, fast, and cost-effective by using CNN model VGG16 for accurately detecting victims of maritime accidents and using the swarm intelligence algorithm Particle Swarm Optimization Algorithm to find the position of our victim in the shortest time possible.

6.1 Future Works

Our research here opened a huge arena in front of us. In the future, we hope to work on a faster and more accurate neural network model. Though in this research, we used many state-of-art CNN models and trained them to yield the most accurate detection, we believe that with models specifically put together to detect and classify the humans emergency will ensure more effective surveillance. We also hope to use the exploitation part of the Particle Swarm Optimization Algorithm to use its convergence procedure to track a target so that in a moving scenario like seawater, the drone can keep track till the rescue reaches the victim. Further, we hope to modify these metaheuristic algorithms to ensure efficiency in the total surveillance process. Last but not least, our aim is to implement it at the hardware level with drones and simulate the complete system architecture in real-life cases.

Bibliography

[1] G. Beni and J. Wang, “Swarm intelligence in cellular robotic systems,” in *Robots and biological systems: towards a new bionics?* Springer, 1993, pp. 703–712.

[2] U. Wilensky, “Netlogo,” Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, http://ccl.northwestern.edu/netlogo/, 1999. [Online]. Available: http://ccl.northwestern.edu/netlogo/.

[3] M. Lovbjerg and T. Krink, “Extending particle swarm optimisers with self-organized criticality,” in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No.02TH8600)*, vol. 2, 2002, 1588–1593 vol.2. DOI: 10.1109/CEC.2002.1004479.

[4] D. Merkle and M. Middendorf, “Ant colony optimization, marco dorigo, thomas stützle. mit press (2004), isbn: 0-262-04219-3,” *European Journal of Operational Research*, vol. 168, pp. 269–271, Jan. 2006. DOI: 10.1016/j.ejor.2004.09.013.

[5] E. Sahin, W. Spears, and A. Winfield, *Swarm Robotics.* Jan. 2007.

[6] C. Blum and D. Merkle, *Merkle, D.: Swarm Intelligence: Introduction and Applications. Springer, Heidelberg.* Jan. 2008, ISBN: 978-3-540-74088-9. DOI: 10.1007/978-3-540-74089-6.

[7] C. Corbane, L. Najman, and E. Pecoul, “A complete processing chain for ship detection using optical satellite imagery,” *International Journal of Remote Sensing - INT J REMOTE SENS*, vol. 31, Dec. 2010. DOI: 10.1080/01431161.2010.512310.

[8] K. Parsopoulos and M. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications.* Jan. 2010, ISBN: [ISBN-10: 1615206663] [ISBN-13: 978-1-61520-666-7]. DOI: 10.13140/2.1.3681.1206.

[9] X.-S. Yang, “A new metaheuristic bat-inspired algorithm,” vol. 284, Apr. 2010. DOI: 10.1007/978-3-642-12538-6_6.

[10] Y.-f. Zhu and X.-m. Tang, “Overview of swarm intelligence,” in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 9, 2010, pp. V9-400-V9–403. DOI: 10.1109/ICCASM.2010.5623005.

[11] B. K. Panigrahi, Y. Shi, and M. H. Lim, “Handbook of swarm intelligence,” 2011.

[12] G. Varela, P. Caamaño, F. Orjales, Á. Deibe, F. López-Peña, and R. J. Duro, “Swarm intelligence based approach for real time uav team coordination in search operations,” in *2011 Third World Congress on Nature and Biologically Inspired Computing*, 2011, pp. 365–370. DOI: 10.1109/NaBIC.2011.6089619.

[13] H. Ahmed and J. Glasgow, "Swarm intelligence: Concepts, models and applications," Feb. 2012. DOI: 10.13140/2.1.1320.2568.

[14] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, Jan. 2012. DOI: 10.1145/3065386.

[15] D. M. P. F. Silva, L. F. F. de Oliveira, M. G. M. Macedo, and C. J. A. B. Filho, "On the analysis of a swarm intelligence based coordination model for multiple unmanned aerial vehicles," in *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*, 2012, pp. 208–213. DOI: 10.1109/SBR-LARS.2012.41.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, Dec. 2014.

[17] S. Mirjalili, S. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014. DOI: 10.1016/j.advengsoft.2013.12.007.

[18] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[19] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv e-prints*, Nov. 2015.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[21] S. P. Yeong, L. M. King, and S. S. Dol, *A Review on Marine Search and Rescue Operations Using Unmanned Aerial Vehicles*, version 10001953, Jul. 2015. DOI: 10.5281/zenodo.1107672. [Online]. Available: https://doi.org/10.5281/zenodo.1107672.

[22] F. Bousetouane and B. Morris, "Fast cnn surveillance pipeline for fine-grained vessel classification and detection in maritime scenarios," in *2016 13th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2016, pp. 242–248. DOI: 10.1109/AVSS.2016.7738076.

[23] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2016. DOI: 10.1109/COMST.2015.2495297.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[25] A. H. Al-Mter and S.-F. Lu, "A modified particle swarm optimization algorithm using uniform design," in *2016 International Conference on Information System and Artificial Intelligence (ISAI)*, 2016, pp. 432–435. DOI: 10.1109/ISAI.2016.0098.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[27] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

[28] J. Fan, M. Hu, X. Chu, and D. Yang, "A comparison analysis of swarm intelligence algorithms for robot swarm learning," in *2017 Winter Simulation Conference (WSC)*, 2017, pp. 3042–3053. DOI: 10.1109/WSC.2017.8248025.

[29] A. F. Agarap, "Deep learning using rectified linear units (relu)," Mar. 2018.

[30] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6154–6162. DOI: 10.1109/CVPR.2018.00644.

[31] D. Du, Y. Qi, H. Yu, *et al.*, "The unmanned aerial vehicle benchmark: Object detection and tracking," Mar. 2018.

[32] A. J. Gallego, A. Pertusa, P. Gil, and R. Fisher, "Detection of bodies in maritime rescue operations using unmanned aerial vehicles with multispectral cameras: Gallego et al.," *Journal of Field Robotics*, vol. 36, Dec. 2018. DOI: 10.1002/rob.21849.

[33] S. Gao and Y. Wen, "An improved artificial fish swarm algorithm and its application," in *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)*, 2018, pp. 649–652. DOI: 10.1109/ICIS.2018.8466458.

[34] R. Hu, B. Tian, S. Yin, and S. Wei, "Optimization of softmax layer in deep neural network using integral stochastic computation," *Journal of Low Power Electronics*, vol. 14, pp. 475–480, Dec. 2018. DOI: 10.1166/jolpe.2018.1579.

[35] Y. Liu, Y. Qin, J. Guo, C. Cai, Y. Wang, and L. Jia, "Short-term forecasting of rail transit passenger flow based on long short-term memory neural network," in *2018 International Conference on Intelligent Rail Transportation (ICIRT)*, 2018, pp. 1–5. DOI: 10.1109/ICIRT.2018.8641683.

[36] S. Wang, Y. Han, J. Chen, Z. Zhang, G. Wang, and N. Du, "A deep-learning-based sea search and rescue algorithm by uav remote sensing," in *2018 IEEE CSAA Guidance, Navigation and Control Conference (CGNCC)*, 2018, pp. 1–5. DOI: 10.1109/GNCC42960.2018.9019134.

[37] P. F. Zhu, L. Wen, X. Bian, H. Ling, and Q. Hu, "Vision meets drones: A challenge," *ArXiv*, vol. abs/1804.07437, 2018.

[38] S. Aslan and S. Demirci, "Solving uav localization problem with artificial bee colony (abc) algorithm," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 2019, pp. 735–738. DOI: 10.1109/UBMK.2019.8907034.

[39] F. Dai, M. Chen, X. Wei, and H. Wang, "Swarm intelligence-inspired autonomous flocking control in uav networks," *IEEE Access*, vol. 7, pp. 61 786–61 796, 2019. DOI: 10.1109/ACCESS.2019.2916004.

[40] W. Hammedi, M. Ramirez-Martinez, P. Brunet, S.-M. Senouci, and M. A. Messous, "Deep learning-based real-time object detection in inland navigation," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9013931.

[41] S. Moosbauer, D. König, J. Jäkel, and M. Teutsch, "A benchmark for deep learning based object detection in maritime environments," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 916–925. DOI: 10.1109/CVPRW.2019.00121.

[42] I. Nasr, M. Chekir, and H. Besbes, "Shipwrecked victims localization and tracking using uavs," in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, 2019, pp. 1344–1348. DOI: 10.1109/IWCMC.2019.8766534.

[43] D. Puri, "Coco dataset stuff segmentation challenge," in *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 2019, pp. 1–5. DOI: 10.1109/ICCUBEA47591.2019.9129255.

[44] Z. Baird, M. K. Mcdonald, S. Rajan, and S. J. Lee, "A cnn-lstm network for augmenting target detection in real maritime wide area surveillance radar data," *IEEE Access*, vol. 8, pp. 179 281–179 294, 2020. DOI: 10.1109/ACCESS.2020.3025144.

[45] Z. Cheng and D. Fu, "Remote sensing image segmentation method based on hrnet," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 6750–6753. DOI: 10.1109/IGARSS39084.2020.9324289.

[46] H. Pratiwi, A. Windarto, S. Susliansyah, *et al.*, "Sigmoid activation function in selecting the best model of artificial neural networks," *Journal of Physics: Conference Series*, vol. 1471, p. 012 010, Feb. 2020. DOI: 10.1088/1742-6596/1471/1/012010.

[47] S. Saxena, R. Jais, and D. R. Sona, "Search and rescue operations using robots demonstrating swarm behaviour," in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, 2020, pp. 115–119. DOI: 10.1109/ICCES48766.2020.9137864.

[48] Y. Sun, Z. Yang, and Y. Dai, "Trustgcn: Enabling graph convolutional network for robust sybil detection in osns," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2020, pp. 1–7. DOI: 10.1109/ASONAM49781.2020.9381325.

[49] Z. Tang, X. Liu, and B. Yang, "PENet: Object detection using points estimation in high definition aerial images," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA: IEEE, Dec. 2020.

[50] M. Wang, X. Luo, X. Wang, and X. Tian, "Research on vehicle detection based on faster r-cnn for uav images," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 1177–1180. DOI: 10.1109/IGARSS39084.2020.9323323.

[51] S. Wei, H. Chen, X. Zhu, and H. Zhang, "Ship detection in remote sensing image based on faster r-cnn with dilated convolution," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 7148–7153. DOI: 10.23919/CCC50068.2020.9189467.

[52] A. N. Albanese, V. Sciancalepore, and X. P. Costa, “Sardo: An automated search-and-rescue drone-based solution for victims localization,” *ArXiv*, vol. abs /2003.05819, 2021.

[53] M. B. Gamra, M. A. Akhloufi, C. Wang, and S. Liu, “Deep learning for body parts detection using hrnet and efficientnet,” in *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2021, pp. 1–8. DOI: 10.1109/AVSS52988.2021.9663785.

[54] G. Muntasha, N. Karna, and S. Y. Shin, “Performance analysis on artificial bee colony algorithm for path planning and collision avoidance in swarm unmanned aerial vehicle,” in *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, 2021, pp. 1–6. DOI: 10.1109/AIMS52415.2021.9466085.

[55] R. Rokhana, W. Herulambang, and R. Indraswari, “Multi-class image classification based on mobilenetv2 for detecting the proper use of face mask,” Sep. 2021, pp. 636–641. DOI: 10.1109/IES53407.2021.9594022.

[56] J. Tang, G. Liu, and Q. Pan, “A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends,” *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 10, pp. 1627–1643, 2021. DOI: 10.1109/JAS.2021.1004129.

[57] L. Varga, B. Kiefer, M. Meßmer, and A. Zell, *Seadronessee: A maritime benchmark for detecting humans in open water*, May 2021.

[58] J. Wang, K. Sun, T. Cheng, *et al.*, “Deep high-resolution representation learning for visual recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3349–3364, 2021. DOI: 10.1109/TPAMI.2020.2983686.

[59] X. Xie and G. Lu, “A research of object detection on uavs aerial images,” in *2021 2nd International Conference on Big Data Artificial Intelligence Software Engineering (ICBASE)*, 2021, pp. 342–345. DOI: 10.1109/ICBASE53849.2021.00070.

[60] E. Dupuis, D. Novo, I. O’Connor, and A. Bosio, “A heuristic exploration of retraining-free weight-sharing for cnn compression,” in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022, pp. 134–139. DOI: 10.1109/ASP-DAC52403.2022.9712487.

[61] N. Su, X. Chen, J. Guan, and Y. Huang, “Maritime target detection based on radar graph data and graph convolutional network,” *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022. DOI: 10.1109/LGRS.2021.3133473.

[62] L. A. Varga, B. Kiefer, M. Messmer, and A. Zell, “Seadronessee: A maritime benchmark for detecting humans in open water,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2260–2270.