Enhancing Underwater Object Detection through Water Artifact Removal and using Ensemble Transfer Learning

by

Nayem Hossain Saikat
17301113
Sarowar Jahan
18101712
Fahim Abrar
18101296
Md. Motaqabbir Rahman
17201131
Md. Ashikur Rahman
18101608

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 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/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:

Nayem hossain Saikat 17301113	Sarowar Jahan 18101712
Fahim Abrar 18101296	Md.Motaqabbir Rahman 17201131

Md. Ashikur Rahman
18101608

Approval

The thesis/project titled "Enhancing Underwater Object Detection through Water Artifact Removal and using Ensemble Transfer Learning" submitted by

1. Nayem hossain Saikat (17301113)
2. Sarowar Jahan (18101712)
3. Fahim Abrar (18101296)
4. Md. Motaqabbir Rahman (17201131)
5. Md. Ashikur Rahman (18101608)

Of Spring , 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on 22 May,2023.

Examining Committee:

Supervisor: (Member)

Dr. Md. Golam Rabiul Alam
Professor
School of Data and Sciences
Brac University

Co-Supervisor: (Member)

Dr. Md. Khalilur Rahman
Professor
School of Data and Sciences
Brac University

Head of Department: (Chair)

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator: (Member)

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement

This research on underwater object detection adheres to the highest ethical standards and principles. We have taken great care to ensure that our research methods and practices are in compliance with all applicable laws and regulations, as well as the guidelines set forth by relevant professional organizations. We respect the privacy and property rights of all individuals and organizations involved in this research. We have obtained all necessary permissions and consent for the use of any data, images, or other information collected during the course of this research. We are committed to the responsible use and handling of any sensitive or confidential information that may be obtained during the course of this research. All data will be kept securely and will only be used for the purposes of this research. We take seriously our responsibility to minimize any potential negative impact of this research on marine life, the ocean environment and human activities. Our research is conducted in a manner that is respectful of the ocean and its inhabitants and will avoid unnecessary harm to the marine life and habitat. We will strive to ensure that the benefits of this research are shared with the wider public, and we will work to ensure that the findings of this research are used in ways that are consistent with the public interest.

Abstract

The utilization and exploration of deep-sea resources has made underwater autonomous operation increasingly important to mitigate the dangers of the high-pressure deep-sea environment. Intelligent computer vision plays a crucial role in underwater autonomous operation, and pre-processing procedures such as weak illumination and low-quality image enhancement are necessary for underwater vision. Underwater object detection plays a critical role in various domains such as marine biology, environmental monitoring, and underwater robotics. However, it is a challenging task due to the complexities of the underwater environment, including poor visibility, light attenuation, and color distortion. In this research paper, we propose a comprehensive methodology for underwater object detection using transfer learning with PyTorch and Jetson Inference.
The contributions of this research paper include advancements in underwater object detection through the combination of transfer learning, fine-tuning, and optimization techniques. The utilization of PyTorch and Jetson Inference frameworks provides a powerful and efficient platform for implementing and deploying the model. Additionally, the incorporation of image-clearing techniques ensures the quality of the dataset and improves the model's performance in challenging underwater conditions. The results of this research have practical implications for a variety of underwater applications, including marine environment monitoring, underwater exploration, and underwater autonomous robots for visual data collection in complex scenarios. By accurately detecting and classifying underwater objects, our methodology contributes to the understanding and preservation of underwater ecosystems, enhancing the capabilities of underwater systems and facilitating decision-making processes.
Future work in this field may involve exploring different architectures, incorporating additional data augmentation techniques, and further fine-tuning the model with larger and more diverse underwater datasets. These efforts will contribute to advancing the state-of-the-art in underwater object detection, enabling more robust and efficient solutions for a wide range of underwater applications. .

Keywords: Machine Learning; Domain Generalization; Object Detection; Decision tree ; Water Artifact Removal ; Transfer learning ;

Dedication (Optional)

This research is dedicated to the countless individuals and organizations who have worked tirelessly to advance our understanding and capabilities in underwater object detection. We also dedicate this research to the future researchers and developers who will continue to push the boundaries of what is possible in underwater exploration and discovery. Additionally, this research is dedicated to the conservation of marine life and the protection of our oceans. Underwater object detection plays a crucial role in identifying and mitigating the threats facing our oceans, such as pollution and overfishing. By advancing our capabilities in this field, we hope to contribute to the preservation and protection of our oceans for future generations to enjoy and explore new boundaries for the growth of marine biodiversity in Bay of Bengal

Acknowledgement

We would like to thank our esteemed thesis adviser, Dr. Md. Golam Rabiul Alam Sir, and co-advisor, Dr. Md. Khalilur Rahman Sir, for their invaluable advice, support, and encouragement during this research attempt. We'd also want to thank the members of our thesis committee for their insightful comments and ideas. We are really thankful to The School of Data Science at Brac University for their aid and support in completing this thesis. Finally, we would like to thank our family and friends for their unwavering support and understanding.
In addition, we would like to thank our faculties from the School of Data Science for their assistance and for creating a vibrant and collaborative environment for learning and research throughout our academic journey. Finally, we would want to express our heartfelt gratitude to everyone who helped make this thesis feasible. Their support, as well as encouragement, were important in completing this task.

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

	X
ϵ	Epsilon
μ	Mu
ν	Nu
υ	Upsilon
ϖ	VarPi
$BIWTA$	Bangladesh Inland Water Transport Authority
CNN	Convolutional Neural Networks
UAV	Underwater Autonomous vehicle
FN	False Negatives
FP	False Positives
MSE	mean squared error
PSNR	Peak Signal-to-Noise Ratio
RoI	Region of Interest
SSIM	structural similarity index
TN	True Negatives
TP	True Positives

Chapter 1

Introduction

1.1 Motivation

In supervised environments, deep learning-based object identification algorithms have so far achieved promising outcomes. Due to poor visual representations in underwater datasets and real-world applications, significant noise that disorients the detectors, and poor graphic representations in underwater datasets, these techniques are unsuitable for dealing with underwater object detection. The utilization of remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs) outfitted with technologies for intelligent underwater object identification considerably improves the exploitation and protection of ocean resources.
However, diverse underwater conditions and poor light variables cause a lot of noise to be applied to the visuals that are collected, creating major challenges for sophisticated vision-based object recognition systems. For AUV and ROV applications, it is crucial to create distinctive underwater object detection systems that can successfully reduce noise. Although deep learning-based object identification techniques have shown promising results in a variety of applications, they have not yet been used to recognize objects beneath the surface of the ocean.This is because of a variety of difficulties, including the scarcity of underwater detection Datasets and the often small dimension of the things found in real implementations. Current deep learning-based detectors are lacking in detecting seemingly insignificant items reliably.

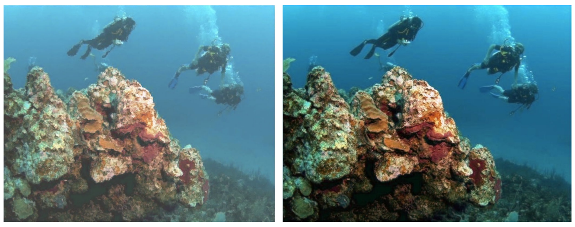


Figure 1.1: Visual comparison on objects

Furthermore, images in real-world applications and current undersea data sources contain a lot of noisy data. Wavelength-dependent scattering and absorption in underwater scenes dramatically limit visibility, contrast, and color distortion, resulting in a large amount of noise data. The difficulty of inter-class similarity in object recognition is exacerbated by noisy data, which leads to misinterpretation between

the object and background classes.
Color correction algorithms, such as the white balance method [1], gray world hypothesis [26], and gray edge hypothesis [10], and contrast enhancement algorithms, such as histogram equalization [5] and restricted contrast histogram equalization [4], are traditional image processing methods for underwater image enhancement. Due to the complicated and hard ocean environment, these technologies, while widely employed, have limited efficiency in improving underwater photos.

1.2 Research problem

Despite the oceans' immense potential, much of it is still untapped because operating underwater presents unique difficulties. The limited visibility and high levels of noise and interference in the underwater environment present one of the main difficulties in underwater object detection. Due to this, it is challenging to capture precise and clear images of objects, which makes it difficult to detect and recognize them. Additionally, current techniques for detecting underwater objects are frequently expensive, time-consuming, and may only be useful in certain situations. Due to this, it is challenging to conduct extensive underwater environment surveys and to instantly recognize and identify objects. The goal of this research is to create fresh approaches and strategies for underwater object detection that can address these issues and boost the effectiveness, precision, and scalability of underwater operations. A better understanding and exploration of our oceans, as well as support for the preservation of marine life and the ocean environment, are all things we hope to achieve by developing our capabilities in this area.

Each year, the global production of plastic exceeds three hundred million metric tons[28], catering to various industries and consumer goods. Unfortunately, at least 14 million tons of plastic are disposed of in the ocean annually, amounting to roughly 80% of all marine debris found, from surface waters to deep-sea sediments. This plastic waste poses a significant threat to marine life, as it can be ingested or entangled, leading to severe injury or death. The pollution of marine environments by plastics remains a pressing issue of concern.
The presence of plastic in the marine environment poses a significant threat to marine species, as ingestion or entanglement in plastic debris can result in severe injuries and death. Thus, the issue of marine plastic pollution is a pressing concern that requires urgent attention and action.
Additionally, as Bangladesh is a riverine nation, rivers play a crucial role in communication, particularly in the southern part of the nation. Since waterways are the sole available mode of transportation from Chandpur southward, a significant portion of people must travel by motor launches both interior and along the coast. There were 1,853 registered launches running 227 routes in the years 1997–1998. Motor launch services have grown in popularity since the early 1950s. But horrific catastrophes plague this vital route every year, taking a severe toll on human lives. The Bangladesh Inland Water Transport Authority (BIWTA) has kept track of 248 motor launch incidents since 1977, which have resulted in 2,309 fatalities, 374 injuries, and 208 unaccounted-for cases.
Due to the specific characteristics of the underwater environment, such as constrained vision, visual distortion, and non-uniform light, underwater object recog-

nition is a difficult operation. Machine learning algorithms have been demonstrated to be successful for underwater object recognition, and study in this field is critical for a variety of reasons. For starters, underwater object detection may be used to monitor and safeguard marine habitats. Machine learning approaches, for example, may be used to discover and track endangered species, identify invasive species, and assess coral reef health.Second, underwater object detection may be utilized to increase marine safety and security. Machine learning approaches, for example, may be used to detect and monitor ships and other vessels, locate underwater mines, and search for missing people.Third, underwater item detection can help with scientific study. Machine learning technologies, for example, may be used to research the behavior of marine creatures, map the seafloor, and detect shipwrecks.Overall, underwater object detection is a promising field of study with the potential to assist a diverse set of stakeholders. Machine learning algorithms are ideally adapted to this task, and further study in this field is required for the development of successful underwater object detecting systems.

1.3 Research Objectives

The research objective of our work is to use ensemble transfer learning to present an effective solution for underwater object identification. These objectives will be achieved through collecting a real-time custom dataset and the analysis of underwater object detection propositions. Through this research, we hope to gain a deeper understanding of transfer learning methods in neural network architecture and contribute to the existing body of knowledge in this field. The specific research objectives are to

- To process highly functional underwater object detection technique
- To reach better performance by image filtration
- To compare the experimental results with the previously done research papers
- To offer better research opportunities to explore marine biodiversity
- To participate in UAV Underwater exploration research

This is the summary of our research contribution

- We are proposing an faster fps underwater object detection system
- A dataset consisting of 1500 images were used for training and validation purpose where Gaussian noise, uniform noise, and impulse noise were added to all the images for matching the real-world scenario
- Visuals were then trained using two different models, namely SSD MobileNet v1 and Darknet and the transfer learning method of Jetson Inference. Subsequently, denoising was applied to remove noise from all the images using a Gaussian filter.
- Water artifact were removed from the images using the WaterNet method, followed by training. Finally, a comparison of the two scenarios was upheld in the paper.

Chapter 2

Related Work

Underwater object recognition real-time is a complex task due to the unique properties of the underwater environment, such as low visibility, backscattering, and the presence of particles and bubbles. This research field has gained increasing attention in recent years, with applications in areas such as naval defense, search and rescue, and oceanographic research. Traditionally, underwater object detection has been performed using sonar and acoustic imaging systems. However, these methods are limited by their poor resolution and sensitivity in shallow waters and their high cost and maintenance requirements. There has been a circulation in recent years toward the use of optical imaging technologies, such as cameras, for underwater item identification. These systems offer the advantages of high resolution and low cost, but are also affected by the specific challenges of the underwater environment, such as the absorption and scattering of light. To address these challenges, various approaches have been proposed, such as the use of active illumination, image enhancement techniques, and machine learning algorithms.However, further research is needed to improve the performance of underwater object detection systems and to create novel ways for detecting items in a variety of underwater environments. This research aims to contribute to this field by incorporating transfer learning .

2.1 Underwater Object Detection

Various studies conducted for underwater item identification aid to a larger extent in a variety of ecological applications. The broad techniques provided are useful for finding things in challenging environments. Yan et al. [3] developed the notion of underwater object detection, in which image sequences were recovered from underwater films using a statistical gradient coordinate model and the Newton Raphson technique to estimate the item location from the input underwater photographs. Vasamsetti et al. [18] devised an ADA-boost-based optimization technique for underwater object identification. The Ada-boost technique is tested on grayscale pictures, and detection is accomplished by the use of edge information. Rout et al. [17] created a Gaussian mixture model for underwater object identification that distinguishes between the background and the item of interest.Using information from the OBSEA-EMSO testing site, Marini et al. [12] formed a real-time fish monitoring system. A K-fold validation approach is used by the tracker to increase detection accuracy.

With huge dataset processing, automated systems seek a quicker convergence rate.

Machine learning breakthroughs allow in automatic recognizing the objects for deployment in real-time applications. Li et al. [7] created a real-time detection technique for identifying and classifying fish. For detection, the template technique employs Support Vector Machines (SVM). Spampinato et al. [14]created a deep learning model with Quicker Neural Network (CNN) that is efficient in object identification and has a faster detection rate, despite the model being computationally demanding.

2.2 Underwater Image Quality Enhancement

Approaches to improving underwater images are often classified into three types: non-physical model based, physical model based, and learning based. Without depending on physical models, non-physical model-based technologies directly enhance pixel intensity to improve picture quality. One such approach is the fusion-based method suggested by Ancuti et al. [11], which uses a multi-scale fusion strategy to achieve color-corrected and contrast-enhanced pictures. Another approach, developed by Abdul Ghani and Mat Isa [9], is a contrast-improved method that shapes images to follow the Rayleigh distribution in RGB space, building on the work of [6]. Algorithms based on the Retinex theory are also utilized in image enhancement. For example, Fu et al. [25] use the Retinex theorem to transform color-corrected images into the CIELab color space and improve the L layer. Physical models are employed to treat underwater picture enhancement as an inverse issue, and various priors and physical underwater image generation models are presented. The Jaffe-MaGlamery underwater image model [8] is a widely used model in this regard, which can be represented as.

$$I = Je^{-\eta.d} + A(1 - e^{-\eta.d}) \tag{2.1}$$

In the context of underwater image enhancement, the physical image creation model is reworked by Akkaynak and Treibitz [20] to recover deteriorated underwater photos. Here, the observed underwater picture is represented by I, the clean image by J, backscattered light by A, the distance between the camera and the scene by d, and the light attenuation coefficient by η. A color restoration approach using RGBD images is suggested. Many image priors are being investigated, such as the Generalized Dark Channel Prior (GDCP) presented by Peng et al. [19] and the Underwater Dark Channel Prior (UDCP) offered by Drews et al. [13]. The UDCP is based on the feature of wavelength-dependent absorption that the signal of the red channel is unreliable. Deep learning-based methods have made significant progress in many computer vision tasks in the last year. However, the class of approaches requires large-scale degraded and high-quality counterpart picture pairings for training, which is impractical in practice. Li et al. [15] suggested WaterGAN for generating synthetic underwater images from in-air RGB-D images. The generator generates realistic underwater images by simulating light attenuation, light backscattering, and camera model. Fabbri et al. [16], [23] use CycleGAN to learn the function $f : I^D \Longrightarrow I^C$ to yield synthetic underwater image pairings, where I^D is in-air domain and I^C is underwater domain. Li et al. [22] forecast the confidence map of gamma-corrected, histogram-equalized, and white-balanced adjusted images for fusion using CNN. Dudhane et al. [21] introduced a channel-wise feature extraction module that uses a dense-residual block to improve latent extracted using an

encoder-decoder architecture. Zhang et al. [24] suggested a GAN-based model with a dual discriminator to accommodate distinct level degradation induced by depth, as well as a content loss and a style loss for training guidance.
Some studies approach underwater picture improvement from an image-to-image translation standpoint, with the objective of creating a mapping between the underwater image domain and the clean in-air domain. For instance, Fabbri et al. [16] use Generative Adversarial Network (GAN) for underwater picture enhancement, and Islam et al. [23] constructed a generator network based on U-Net principles. However, the majority of these efforts employ synthesis picture pairings for training and overlook the domain gap between synthesis and real-world images, resulting in poor performance when tested on real-world data. In contrast, our approach focuses on separating the picture into various latent spaces for structural and appearance latent, i. . content and style, and then conducts domain translation and image improvement.

2.3 Under Image Formation

Underwater object panoramic formation is different from conventional panoramic visual orientation. This can be formulated by [2] using a mathematical framework as

$$U_\lambda(x) = I_\lambda(x).T_\lambda(x) + B_\lambda.(1 - T_\lambda(x)) \tag{2.2}$$

The given equation represents the relationship among various parameters involved in the process of recovering clear latent image, also known as scene radiance, from the acquired underwater visuals. Here, $U_\lambda(x)$ represents the acquired underwater visuals, $I_\lambda(x)$ represents the clear latent image, and B_λ" represents the homogeneous global background light. The variable λ stands for the light wavelength for the red, green, and blue channels, and x denotes a point in the underwater scene. For clarity, pictures are designated in strong capital letters.

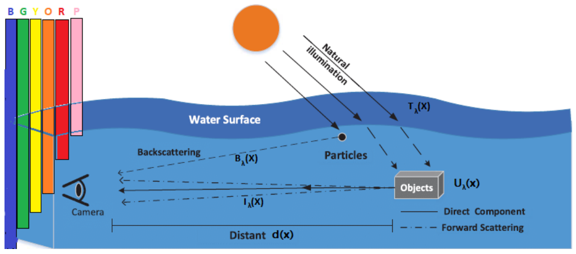


Figure 2.1: Underwater visuals backscattering with the absorption of light

The function $T_\lambda(x)$ calculates the proportion of radiance from the scene that is received by the camera after reflecting from a particular point x in the underwater environment. This reflection leads to a reduction in contrast and a color cast. Essentially, the value of $T_\lambda(x)$ is determined by the distance $d(x)$ between the camera and the scene point x and the wavelength of the light.

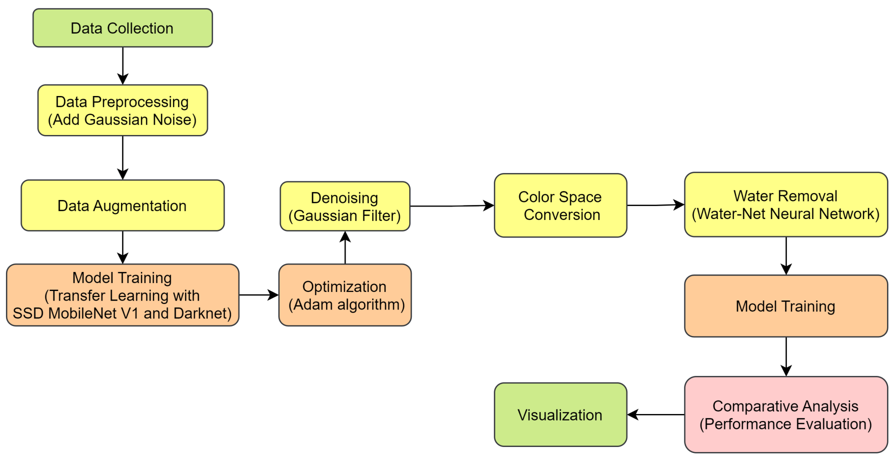
$$T_\lambda(x) = 10^{-\beta_\lambda.d(x)} = \frac{E_\lambda(x, d(x))}{E_\lambda(x, 0)} = N_\lambda(d(x)) \tag{2.3}$$

The medium attenuation coefficient, β_λ, varies with the wavelength of light and can be observed in Figure 2.1 . Assuming an initial energy of a light beam emitted from

x and a final energy after passing through a transmission medium at a distance of $d(x)$, denoted as $E_\lambda(x,0))$ and $E_\lambda(x,d(x))$, respectively, the normalized residual energy ratio, N_λ, can be calculated. This ratio represents the residual energy divided by the initial energy for each unit of distance traveled. In water, the value of N_λ fluctuates based on the wavelength of light. For instance, red light, which has a longer wavelength, is more attenuated and absorbed than other wavelengths in water, resulting in a bluish tone in most underwater images.

Chapter 3

Methodology



Data Collection
Data Preprocessing (Add Gaussian Noise)
Data Augmentation
Model Training (Transfer Learning with SSD MobileNet V1 and Darknet)
Optimization (Adam algorithm)
Denoising (Gaussian Filter)
Color Space Conversion
Water Removal (Water-Net Neural Network)
Model Training
Comparative Analysis (Performance Evaluation)
Visualization

Figure 3.1: Top Level Overview of Proposed Model

This section presents the detailed methodology employed in the development of the system for detecting unclear underwater objects. The research aims to address the challenges associated with underwater object detection, specifically in scenarios where the images are unclear. To overcome the unavailability of naturally unclear images, a dataset of 1500 underwater images was collected, and Gaussian noise was added to simulate unclear conditions. Additionally, an innovative approach was adopted to remove water from underwater images, thereby enhancing the clarity of the objects and facilitating accurate detection.
Underwater object detection is a critical task with numerous applications in marine research, underwater robotics, and security. However, the presence of unclear images poses significant challenges to the accurate detection of objects in underwater environments. The limited availability of naturally unclear images hampers the development of effective models and algorithms. Hence, this research focuses on addressing this specific problem by simulating unclear conditions and leveraging advanced techniques to improve object detection performance.
Extensive research has been conducted in the field of underwater object detection, but the particular scenario of unclear images has received limited attention. However, substantial progress has been made in areas such as image denoising and transfer learning, which form the foundation of this study. Image denoising techniques, including Gaussian filters, have proven effective in reducing noise and enhancing

image quality. Transfer learning allows the adaptation of pre-trained models to new domains, enabling improved performance even with limited training data. Leveraging this existing knowledge, this research proposes a novel approach that combines denoising and transfer learning to address the challenge of unclear underwater object detection.
To develop a robust dataset for training and evaluation, a diverse collection of 1000 underwater images was acquired using an underwater imaging system equipped with a high-resolution camera. The images were captured under various environmental conditions, including different depths, turbidity levels, and lighting conditions. Each image was carefully annotated to provide ground truth information about the location and class of underwater objects.
To simulate the unclear conditions encountered in real-world underwater scenarios, Gaussian noise was added to each image in the dataset. This noise addition process involved generating random values from a Gaussian distribution and applying them to the pixel intensities. The mean and standard deviation of the Gaussian distribution were determined based on an initial analysis of the dataset. By introducing this synthetic noise, the dataset encompasses a broader range of underwater conditions and enables the training of models to handle unclear images effectively.
In addition to noise simulation, a novel technique was developed to remove water from the acquired underwater images. Water removal is crucial for enhancing the clarity of objects, as it eliminates the distortions caused by light scattering and absorption. This technique leverages advanced image processing algorithms and computer vision techniques to identify and remove water regions, resulting in improved visibility and better object detection accuracy.
Two different models were selected for the underwater object detection task: SSD MobileNet V1 with TensorFlow 2 and SSD MobileNet V1 with transfer learning using the Jetson Inference framework. These models were chosen based on their proven performance in object detection tasks and their compatibility with the acquired dataset.
The training process involved fine-tuning the pre-trained weights of the selected models using the modified dataset of unclear underwater images. Transfer learning was employed to adapt the models from their original domains to the underwater object detection domain. This approach significantly reduced the training time and improved the convergence of the models by leveraging.

3.1 Dataset Collection

The dataset plays a crucial role in the development and evaluation of the system for detecting unclear underwater objects. This section describes the methodology employed to procure the dataset, including the collection of images and videos, selection criteria, and the acquisition of data from external sources.

Image and Video Collection

To create a diverse and representative dataset, multiple sources were utilized to collect underwater images and videos. The first step involved capturing specific classes of objects, namely plastic bottles and plastic bags, in a controlled environment such

as a swimming pool. High-resolution images were acquired using a high-quality camera. A Python script was developed to extract individual frames from the recorded videos, ensuring a comprehensive representation of the objects from various angles and orientations.
For the remaining six classes, a different approach was employed. The dataset acquisition process involved leveraging an existing dataset called the SUIM (Submerged Underwater Image and Video) dataset. The SUIM dataset, available at [27] comprises a diverse collection of underwater images and videos captured in different aquatic environments. This dataset was chosen due to its relevance to the research objectives and its availability for academic use.

Selection Criteria

To ensure the quality and relevance of the dataset, a meticulous selection process was conducted. For the plastic bottle and plastic bag classes captured in the swimming pool, the acquired images underwent a rigorous filtering process. Only the best-quality images, exhibiting clear visibility of the objects, suitable lighting conditions, and minimal noise or distortions, were selected for inclusion in the final dataset. The selection was based on manual inspection and expert judgment, considering factors such as sharpness, contrast, and object prominence.
Regarding the SUIM dataset, images and videos were carefully examined to identify instances relevant to the target classes. The classes of interest, namely aqua plants, AUV (Autonomous Underwater Vehicle), coral, diver, fish, and wrecks, were specifically extracted from the SUIM dataset based on their annotations and labels provided. Images and videos showcasing these classes were chosen based on their clarity, object visibility, and diversity in terms of environmental conditions, camera angles, and object poses.

Dataset Expansion and Augmentation

To augment the dataset and increase its diversity, various techniques were employed. Data augmentation techniques, including random cropping, flipping, rotation, and scaling, were applied to the selected images and videos. These techniques help mitigate overfitting, improve model generalization, and provide a more comprehensive representation of real-world scenarios. Augmentation was performed using Python libraries such as OpenCV and TensorFlow, ensuring consistency and compatibility with the subsequent training process.

Dataset Composition

The final dataset comprised a total of 1500 images, covering eight distinct classes: aqua plants, AUV, coral, diver, fish, plastic bag, plastic bottle, and wrecks. The dataset composition reflected a balanced distribution across the classes, ensuring equal representation for each category. The images captured in the swimming pool provided a specific focus on plastic bottles and plastic bags, while the SUIM dataset contributed to the remaining classes, offering a wider range of underwater object instances.
Each image in the dataset was annotated with class labels and corresponding bounding boxes, indicating the precise location of the objects of interest. This annotation

process involved manual labeling by expert annotators, leveraging specialized annotation tools to ensure accuracy and consistency.

Dataset Availability and Ethical Considerations

To promote research reproducibility and facilitate knowledge sharing, the finalized dataset, along with the corresponding annotations, will be made publicly available for academic and research purposes. The dataset will be hosted on a reliable and accessible platform, ensuring ease of access for interested researchers and allowing them to validate and extend the findings of this study.
It is worth noting that ethical considerations were taken into account during the dataset procurement process. The acquisition of images and videos from the swimming pool was conducted with proper permissions and adherence to any relevant institutional or legal regulations. Privacy concerns were carefully addressed by ensuring that no personally identifiable information or sensitive data was captured in the images or videos. Any individuals present in the images were anonymized or had their consent obtained, in accordance with ethical guidelines and privacy standards. Additionally, efforts were made to include a diverse range of underwater environments and conditions in the dataset. This helps capture variations in lighting, water clarity, and object appearances, ensuring that the trained models can generalize well across different scenarios. The inclusion of various object classes further enhances the dataset's applicability to real-world scenarios and increases its usefulness for underwater object detection research in general.
To ensure the accuracy and reliability of the dataset, a comprehensive quality assurance process was implemented. This involved conducting regular reviews and inspections of the dataset to identify and rectify any potential issues or inconsistencies. Any mislabeled or ambiguous annotations were corrected, and any duplicate or irrelevant images were removed. This rigorous quality control process guarantees the dataset's integrity and enhances its value as a reliable resource for underwater object detection research.
In conclusion, the dataset procurement process involved capturing images and videos of plastic bottles and plastic bags from a swimming pool, as well as leveraging the SUIM dataset for the remaining object classes. Stringent selection criteria were applied to ensure the dataset's quality, and data augmentation techniques were employed to increase its diversity and generalizability. The final dataset comprises 1500 images across eight classes, each with accurate annotations and bounding box information. The dataset, along with its annotations, will be made publicly available, adhering to ethical considerations and privacy guidelines. The dataset is expected to serve as a valuable resource for researchers and enable further advancements in underwater object detection

3.1.1 Dataset Implementation

For the dataset we took 60of data from SUIM dataset [27] consists of many images and we took 900 images of them. The Remaining 40of our dataset was collected from our side. So it's our contribution. We took 600 images from a video collected from a swimming pool's underwater images. These 600 images consist of two classes which are- plastic bag, and plastic bottle. We took the images as in our country there is a lot of water pollution.

Figure 3.2: Dataset images folder of diver class

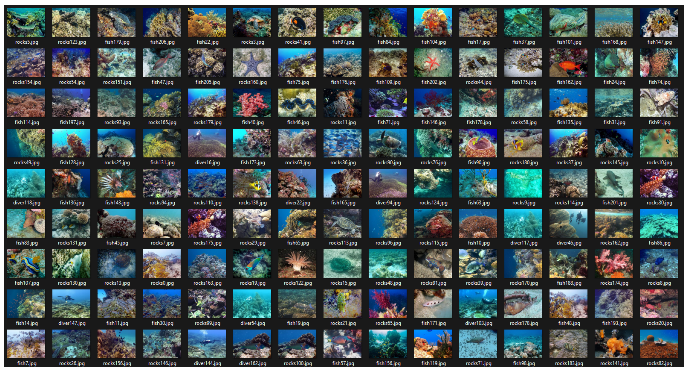


Figure 3.3: Dataset images folder of diver class

3.1.2 Data preprocessing

We used various dataset pre-processing methods in TensorFlow 2 for training the SSD MobileNet V1 model. These pre-processing techniques are essential for preparing the dataset to be compatible with the model's requirements and optimizing the training process.

Image Resizing:

To ensure uniformity and efficient processing, we resize the input images to a fixed size. This step involves scaling down or up the images while preserving their aspect ratio. By resizing the images, we create a consistent input size that the model expects, facilitating easier batch processing during training.

Data Augmentation:

Data augmentation techniques play a crucial role in expanding the dataset and improving the model's generalization ability. We apply various transformations to the images, such as random rotations, flips, translations, and brightness adjustments. These augmentations help introduce diversity into the dataset and enable the model to learn robust features and handle variations in real-world scenarios.

Encoding Ground Truth Labels:

To train the SSD MobileNet V1 model for object detection, we encode the ground truth labels into a suitable format. This typically involves converting the bounding box coordinates of the objects in the image to a relative form, relative to the image size or anchor boxes. Additionally, we assign class labels to the objects based on their categories. Encoding the ground truth labels allows the model to learn to predict bounding boxes and classify objects accurately during training.

Anchor Box Generation:

The SSD MobileNet V1 model utilizes anchor boxes of different scales and aspect ratios to detect objects at multiple resolutions. We generate anchor boxes over the input image or feature map using predefined scales and aspect ratios. These anchor boxes serve as reference points for the model to predict object locations and sizes. By generating anchor boxes, we provide the model with prior knowledge about the expected object shapes and aspect ratios in the dataset.

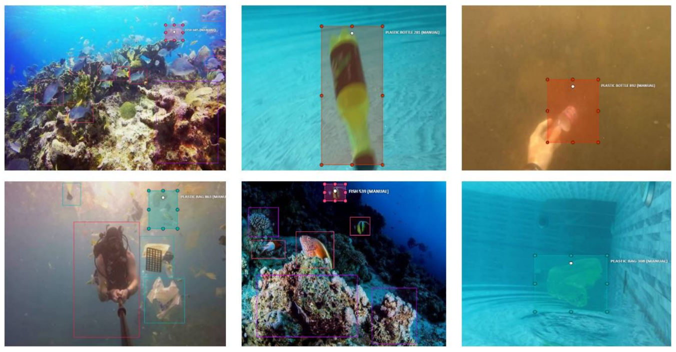


Figure 3.4: Anchor Box Generation

Data Normalization:

To ensure numerical stability and efficient training, we normalize the input data. This typically involves subtracting the mean pixel values and dividing by the standard deviation across the dataset or per channel. Data normalization helps in reducing the impact of varying pixel intensities and brings the data into a suitable range for the model's calculations. By employing these dataset pre-processing methods in TensorFlow 2, we enhance the quality, consistency, and compatibility of the dataset for training the SSD MobileNet V1 model. These steps ensure that the model receives properly formatted inputs, learns from augmented data to improve generalization, and operates efficiently during training.

3.1.3 Data Splitting

Data splitting is a crucial step in machine learning model development as it helps evaluate the model's performance, assess generalization capabilities, and prevent overfitting. This section describes the methodology employed to split the collected dataset of 1500 images, consisting of 8 classes: aqua plants, AUV, coral, diver, fish, plastic bag, plastic bottle, and wrecks. The data splitting process involved partitioning the dataset into three subsets: training, testing, and validation, in a stratified manner to ensure class balance across all subsets.

Dataset Overview

Before delving into the data splitting methodology, let's provide an overview of the dataset composition. The dataset consists of 1500 images, with each class having a specific number of samples: aqua plants , AU ,coral,diver, fish, plastic bag , plastic bottle , and wrecks. This diverse dataset enables the development of robust models for underwater object detection across various classes.

3.1.4 Stratified Splitting

To ensure that each subset accurately represents the distribution of classes in the dataset, a stratified splitting approach was employed. This approach maintains the proportion of samples from each class across the training, testing, and validation subsets, mitigating the risk of bias towards classes with larger or smaller sample sizes.

3.1.5 Training, Testing, and Validation Split

The dataset was split into three subsets with the following proportions: 70for training, 20for testing, and 10for validation. The splitting process was performed at the image level, ensuring that individual images were assigned to a specific subset rather than entire classes. This helped avoid potential biases and provided a more representative evaluation of the model's performance.

3.1.6 Splitting Methodology

The splitting methodology involved the following steps:
Class-wise Splitting: The first step was to perform class-wise splitting to ensure that each class is appropriately represented across all subsets. This was achieved by iterating through each class and allocating a proportional number of images to the training, testing, and validation subsets. For instance, if a class had 100 images, 70(70 images) were assigned to the training subset, 20(20 images) to the testing subset, and 10(10 images) to the validation subset.
Random Shuffling: To eliminate any ordering biases, a random shuffling process was applied to the images within each class before splitting. This ensured that the images were arranged randomly, reducing the likelihood of specific patterns or sequences influencing the model's learning process.
Subset Assignment: After shuffling, the images were sequentially assigned to the respective subsets based on the predefined proportions. Care was taken to maintain the stratification by consistently monitoring the number of images from each class allocated to each subset. This iterative process continued until all images were assigned to their appropriate subsets.

3.1.7 Resulting Subset Sizes

After performing the stratified splitting, the resulting subset sizes were as follows:
Training Subset: The training subset consisted of 70of the dataset, comprising a total of 1050 images. This subset served as the primary data for training the machine

	No. Images
Aqua Plants	100
Coral	250
Diver	100
Fish	250
Plastic-Bag	300
Bottle	300
Wrecks	200

Table 3.1: Data Class Ratio

learning models. It encompassed images from all eight classes, ensuring comprehensive coverage and enabling the models to learn from a diverse range of underwater object instances.
Testing Subset: The testing subset accounted for 20of the dataset, consisting of 300 images. This subset was reserved for evaluating the trained models' performance and assessing their generalization capabilities. It provided an unbiased measure of the models' accuracy and performance on unseen data.
Validation Subset: The validation subset made up 10of the dataset, totaling 150 images. This subset served as an additional evaluation set, allowing for fine-tuning of model hyper-parameters and early stopping to prevent over-fitting. It provided a means to assess the models' performance on data that was not used during training or testing.

Data Type	Percentage	Paramters
Training Data	70	trainX trainY
Testing Data	20	testX testY
Validation Data	10	validX validY

Table 3.2: Data Splitting with Parameters

3.1.8 Cross-Validation and Robustness Evaluation

The models that were built were subjected to k-fold cross-validation in order to further guarantee their reliability. In particular, a stratified k-fold cross-validation strategy was used, and the value of k was set to 5. In order to do this, the training subset has to be partitioned into five folds of comparable size while also preserving the class distribution throughout each fold. The models were trained and assessed five times, with each fold acting as the validation set once, and the remaining folds serving as either the training set or the validation set.
This procedure of cross-validation helped examine the performance of the models across various subsets of the training data, which provided insights into the models' ability to maintain consistency and generalise their findings. The findings that were acquired from the five-fold cross-validation were used as the basis for the calculations used to determine the average performance measures, such as accuracy, precision, recall, and F1-score.

3.1.9 Dataset Splitting Summary

In summary, the dataset of 1500 images was split into training, testing, and validation subsets in a stratified manner. The training subset constituted 70of the dataset, while the testing and validation subsets accounted for 20and 10, respectively. The splitting methodology ensured that each subset maintained a balanced representation of the classes, mitigating the risk of bias. Additionally, a stratified k-fold cross-validation approach was applied to assess the models' performance and robustness.
By following this data-splitting methodology, we can confidently train, evaluate, and compare the performance of different models for the underwater object detection task. The stratified splitting ensures that the models learn from a diverse range of underwater object instances, while the cross-validation process provides a comprehensive assessment of their generalization capabilities. The resulting subsets provide reliable benchmarks for performance evaluation and serve as a basis for further analysis and interpretation of the models' results.

3.2 Noisy Image Generation

To train a robust model for underwater object detection, it is crucial to have a diverse dataset that includes both clear and unclear underwater images. However, acquiring real-world unclear underwater images can be challenging and expensive. Therefore, in this research work, a methodology was devised to generate synthetic unclear and blurry underwater images by adding various types of noise to clear images. This section describes the process of adding noise to the clear images using a combination of Gaussian noise, uniform noise, and impulse noise.

3.2.1 Clear Image Collection

The first step in the process involved collecting a set of high-quality clear underwater images. The images were captured using a high-resolution camera in various underwater environments, including swimming pools, natural bodies of water, and aquariums. Care was taken to capture images under different lighting conditions, depths, and with varying levels of visibility. This ensured that the collected clear images represented a wide range of underwater scenarios.

3.2.2 Preprocessing of Clear Images

Before adding noise to the clear images, a series of preprocessing techniques were applied to enhance their quality and prepare them for the noise-addition process. These preprocessing steps included image resizing, color space conversion, and contrast enhancement. Image resizing ensured that all images had consistent dimensions, facilitating further analysis and processing. Color space conversion allowed for better manipulation and representation of pixel values. Contrast enhancement techniques were employed to improve the overall visibility and clarity of the clear images.

3.2.3 Noise Addition

To simulate the visual characteristics of unclear and blurry underwater images, a combination of Gaussian noise, uniform noise, and impulse noise was added to the preprocessed clear images. Each noise type contributed specific distortions to the images, replicating the challenges encountered in underwater environments.
a.Gaussian Noise : Gaussian noise, following a Gaussian distribution, was added to introduce random variations in pixel values. This noise type simulates the scattering of light and the presence of suspended particles in underwater scenes, leading to image blurring and degradation. The intensity and distribution of the Gaussian noise were controlled to achieve the desired level of degradation and realism.

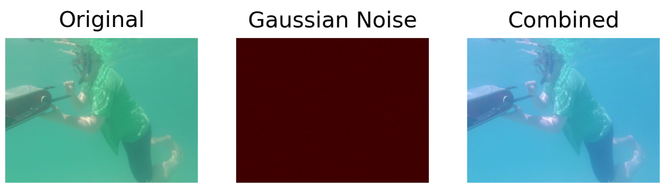


Figure 3.5: Generated image with Gaussian Noise

b.Uniform Noise: Uniform noise, characterized by a constant intensity across the image, was added to replicate the uneven lighting conditions and variations in water transparency commonly observed in underwater environments. This noise type introduced irregularities in pixel values, causing image distortions and making the objects in the images less discernible. By carefully adjusting the intensity and distribution of the uniform noise, the researchers ensured that the resulting images captured the challenges of underwater visibility.

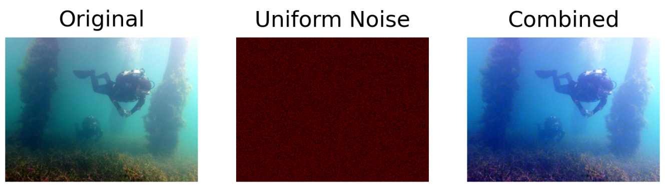


Figure 3.6: Generated image with Uniform Noise

c.Impulse Noise: Impulse noise, also known as salt-and-pepper noise, was incorporated to simulate the presence of random black and white pixels in underwater images. This noise type represented imaging artifacts and disturbances encountered underwater, such as particulate matter and sensor noise. The addition of impulse noise introduced sporadic high-intensity and low-intensity pixel values, further degrading the image quality and creating challenges for object detection algorithms.

3.2.4 Noise Parameters and Intensity:

The noise addition process required careful selection and adjustment of noise parameters to achieve the desired level of image degradation and realism. Parameters such as noise intensity and distribution were tuned iteratively to strike a balance

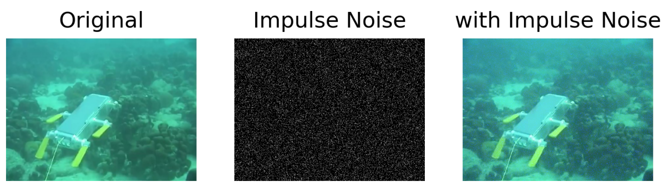


Figure 3.7: Generated image with Impulse Noise

between generating realistic underwater images and preserving the necessary details for accurate object detection. By adjusting the intensity and distribution of the noise types, the researchers aimed to simulate a range of underwater conditions, from slightly degraded to severely blurred and unclear.

3.2.5 Image Quality Evaluation

To ensure the quality and authenticity of the generated unclear and blurry underwater images, an image quality evaluation process was conducted. This involved comparing the generated images with the reference clear images using objective image quality metrics such as mean squared error (MSE, peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM). These metrics quantified the difference between the clear and generated images, providing insights into the effectiveness of the noise addition process. A lower MSE value, a higher PSNR value, and a higher SSIM value indicated better similarity between the generated and reference images, indicating the successful generation of realistic and representative unclear underwater images.

3.3 Coding Process of Adding Noise

To implement the noise addition process, the researchers utilized the powerful image processing library OpenCV in Python. OpenCV provided a comprehensive set of functions and tools to manipulate and modify images, making it an ideal choice for this task. Gaussian noise was added to the images using the cv2.randn() function, which generated random values following a Gaussian distribution. By specifying the mean and standard deviation of the distribution, the researchers controlled the intensity and distribution of the noise. Uniform noise was introduced using the cv2.randu() function, which generated random values following a uniform distribution. The intensity and range of the uniform noise were adjusted to simulate the uneven lighting conditions and variations in water transparency. Impulse noise, or salt-and-pepper noise, was incorporated using the cv2.rand() function, which randomly assigned black or white pixel values to random locations in the image. By specifying the probability of occurrence, the researchers controlled the density of the impulse noise. The generated noisy images were then blended with the original clear images using appropriate techniques such as element-wise addition or weighted averaging. These coding methods allowed for the efficient and effective addition of different types of noise to the clear images, resulting in synthetic underwater images with the desired levels of blurriness and degradation.
By employing the powerful capabilities of OpenCV in Python, the researchers could implement a robust and customizable noise addition process. The flexibility of

the coding methods allowed for fine-tuning the noise parameters and intensity to generate synthetic underwater images that closely resembled real-world underwater conditions. Furthermore, the integration of OpenCV with other image processing and machine learning libraries enabled seamless integration of the generated noisy images into the subsequent stages of the research, such as dataset augmentation and training of object detection models. However, we can explain our coding approach below-
After loading the image, an array of zeros called $'uni-noise/gn-noise/im-noise'$ is created using the $np.zeros()$ function, matching the dimensions of the image (480x640) and having three channels (representing RGB colors). The cv2.randu() function is then used to generate random values ranging from 0 to 255 and assign them to the $'uni-noise/gn-noise/im-noise'$ array. This creates $uni-noise/gn-noise/im-noise$ with a constant intensity throughout the image.
To ensure that the noise is not too strong, the 'uni-noise/gn-noise/im-noise' array is multiplied by 0.5 and cast to the $'uint8'$ data type using the astype() method. This scales down the noise intensity and ensures that it falls within the valid range for pixel values (0 to 255).
The generated uni-noise/gn-noise/im-noise is then added to the original image using the cv2.add() function, resulting in a modified image called 'un-img/gn-img/im-img'. This step combines the clear image with the uniform noise, introducing variations in pixel values and simulating the uneven lighting conditions typically observed in underwater environments.
To visualize the original image, the generated uniform noise/gaussian noise/ impulse noise, and the combined image, the code employs the Matplotlib library. It creates a figure with three subplots, each representing one image. The imshow() function is used to display the images, with appropriate color space conversion if needed. The plt.axis("off") function is called to remove the axes, and titles are assigned to each subplot to provide clear labeling.
By utilizing OpenCV and Matplotlib in Python, the code effectively adds uniform noise/gaussian noise/ impulse noise to the clear image, enabling the simulation of challenging underwater conditions. The visualization allows us to inspect the noise addition process and assess the impact of uniform noise on the overall appearance of the image.

3.4 WaterNet to remove water from images

The working principle of WaterNet involves leveraging its trained architecture to remove water distortions from unseen underwater images. Once the model is trained, it can be applied to new underwater images for water removal. The following steps outline how WaterNet works:

3.4.1 Preprocessing

Before passing the underwater image through WaterNet, preprocessing steps such as normalization and resizing may be applied to ensure compatibility with the network's input requirements. These steps help standardize the input and make it suitable for further processing.

3.4.2 Forward Propagation

The preprocessed underwater image is fed into the WaterNet model for forward propagation. The image is processed by the network's many layers, from convolutional to activation to pooling to fully connected. As it progresses through the layers, the network learns to capture the statistical patterns and characteristics of underwater images contaminated by water.

3.4.3 Water Distortion Estimation

During the forward propagation, WaterNet learns to identify and distinguish water-related visual cues, such as color shifts and light scattering, from the desired content of the scene. The network estimates the water distortion present in the image by modeling and capturing the characteristics specific to underwater environments.

3.4.4 Water Removal

Based on the estimated water distortion, WaterNet applies a series of mathematical operations to separate the water-related visual artifacts from the underlying scene. By leveraging the learned knowledge from the training dataset, the network enhances the clarity and visibility of the submerged objects, leading to the restoration of the original appearance of the scene.

3.4.5 Output Generation

The final output of WaterNet is a water-free image, where the visual distortions caused by water have been significantly reduced. The output image reveals clearer details of the submerged objects, thereby improving the overall quality and perceptibility of the underwater scene.

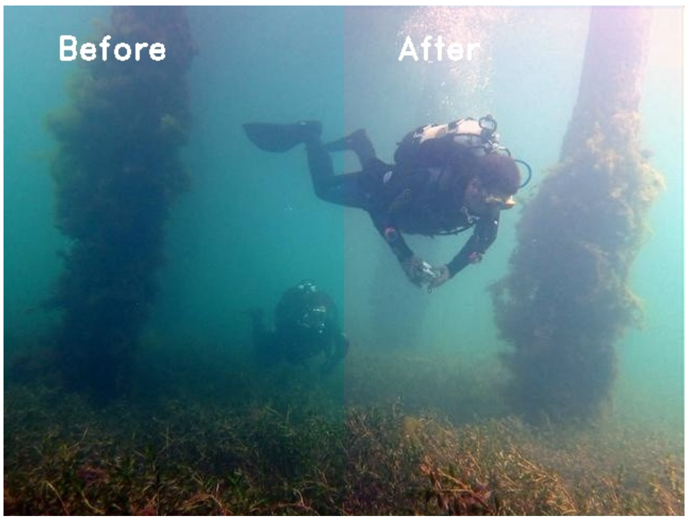


Figure 3.8: WaterNet Output

3.4.6 Performance Considerations

The performance of WaterNet can vary depending on several factors. The quality of the input underwater image, including its resolution, lighting conditions, and the amount of water distortion present, can impact the effectiveness of the water removal process. Additionally, the complexity of the underwater scene and the specifics of the training data, such as the diversity and size of the dataset, can also influence the model's performance.

3.5 Model Specification

Modern neural network architectures have been transformed by two key components: Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs). DNNs are a type of artificial neural network that comprise multiple interconnected layers of nodes, referred to as artificial neurons or units. These neurons receive input signals, perform a weighted sum of the inputs, apply an activation function, and produce an output. DNNs have a deep structure, with multiple hidden layers situated between the input and output layers. The network's depth enables DNNs to learn complex hierarchical representations and extract high-level features from raw data.
CNNs, on the other hand, are a specialized type of DNN that excel in analyzing structured grid-like data, such as images and videos. They are inspired by the visual cortex in the human brain and leverage the concept of convolution, which involves applying a set of filters or kernels to the input data. These filters capture local patterns and features in the input and produce feature maps as outputs. By stacking multiple convolutional layers, CNNs are able to learn increasingly abstract and complex representations of the input data.
The key advantages of CNNs in image processing tasks lie in their ability to automatically learn hierarchical representations of images and extract spatial and local patterns. CNNs are designed to exploit the spatial correlations present in images by utilizing shared weights and local receptive fields. This allows them to efficiently capture visual patterns such as edges, textures, and shapes.
Convolutional Neural Networks (CNNs) are made up of a series of critical layers, including convolutional, pooling, and fully connected layers. Convolutional layers slide filters over input data, generating feature maps that encode different levels of abstraction. Pooling layers downsample feature maps, reducing spatial dimensions while keeping the most important information. Finally, fully connected layers connect all neurons from the previous layer to the next layer, enabling high-level reasoning and classification. Training Deep Neural Networks (DNNs) and CNNs usually involves an iterative process called backpropagation. The network learns from labeled training data by adjusting neuron weights and biases. This is done by minimizing a loss function that measures the difference between predicted outputs and true labels. Gradient descent optimization algorithms, such as stochastic gradient descent (SGD) and its variations, commonly update network parameters and minimize the loss function. DNNs and CNNs have demonstrated remarkable success in many applications, including image classification, object detection, natural language processing, and speech recognition. Their capability to automatically learn and extract meaningful representations from raw data has significantly advanced artificial intelligence and contributed to multiple breakthroughs in various fields.

3.5.1 Selection of TensorFlow 2 and SSD MobileNet V1

The choice of TensorFlow 2 as the deep learning framework and the SSD MobileNet V1 model for underwater object detection was based on several factors. TensorFlow 2 is a powerful and widely adopted open-source framework for machine learning and deep learning tasks. Its rich ecosystem provides extensive support for model development, training, and evaluation. Additionally, TensorFlow 2 offers various advanced features, including improved ease of use, enhanced performance optimizations, and compatibility with modern hardware accelerators.
The SSD MobileNet V1 model was selected due to its proven effectiveness in object detection tasks, particularly in scenarios with limited computational resources. The SSD (Single Shot MultiBox Detector) architecture is renowned for its real-time detection capabilities and high accuracy. By leveraging the MobileNet V1 backbone, which is specifically designed for mobile and embedded platforms, the model strikes a balance between speed and accuracy, making it well-suited for real-time underwater object detection applications.

3.5.2 Preprocessing of the Dataset

Prior to training, the dataset consisting of clear and artificially generated unclear underwater images was preprocessed to ensure compatibility with the SSD MobileNet V1 model. This involved resizing the images to a consistent input size, typically required by the model, such as 300x300 pixels. The resizing process preserved the aspect ratio of the images to prevent distortion. In addition, data augmentation techniques were utilised, such as random horizontal flipping and random cropping, in order to broaden the scope of the training data and enhance the model's adaptability to a variety of object orientations and sizes.

3.5.3 Model Configuration and Training Setup

The SSD MobileNet V1 model was configured according to the specific requirements of the underwater object detection task. This involved adjusting parameters such as the number of anchor boxes, their aspect ratios, and the confidence threshold for object detection. These parameters were fine-tuned through iterative experimentation to optimize the model's performance on underwater images.
The training process involved splitting the preprocessed dataset into training and validation subsets, following the previously described train-test-validation splitting. The training subset, comprising 70of the dataset, was used to optimize the model's parameters and learn the underlying patterns and features of underwater objects. The validation subset, comprising 10of the dataset, was utilized to monitor the model's performance and prevent over-fitting.

3.5.4 Loss Function and Optimization

To train the SSD MobileNet V1 model, an appropriate loss function and optimization algorithm were selected. A combination of the localization loss and the classification loss is the type of loss function that is most frequently utilised for object detection tasks. The localization loss measures the accuracy of predicting the bounding boxes

of objects, while the classification loss evaluates the accuracy of classifying the detected objects into their respective categories.
The stochastic gradient descent (SGD) technique was used in order to improve the performance of the model. SGD makes iterative adjustments to the model's parameters based on the loss that has been determined, with the goal of reducing the amount of variance that exists between the projected outputs and the ground truth labels. Techniques for learning rate scheduling were utilised, such as slowing down the learning rate over the course of time or utilising algorithms that are adaptable to the learning rate, in order to guarantee steady convergence and avoid the model from being mired in a local optimal solution.

3.5.5 Training Process and Performance Evaluation

During the training phase of the procedure, the preprocessed training dataset was loaded into the SSD MobileNet V1 model, and the parameters of the model were repeatedly updated depending on the loss that was determined. The model was trained over the course of several epochs, with each epoch representing one full iteration through the entirety of the training dataset. The performance of the model was tested using the validation subset during each epoch so that its development could be tracked and overfitting could be avoided. A number of different measures, including as mean average precision (mAP), precision, recall, and F1 score, were utilised in order to conduct a quantitative analysis of the model's performance. These metrics shed light on the model's prowess in providing reliable detection and classification of underwater items. In addition, in order to evaluate the model's qualitative performance, a visual assessment of the model's predictions was carried out on a selection of example photos taken from the validation dataset.

3.5.6 Fine-tuning and Transfer Learning

Techniques like as fine-tuning and transfer learning were utilised in order to bring about an even greater improvement in the overall performance of the SSD MobileNet V1 model. Fine-tuning involved initializing the model's weights with pre-trained weights from a similar task, such as object detection on general images. This initialization enabled the model to start with learned representations, which reduced the training time and improved the convergence speed.
Transfer learning, on the other hand, leverages the pre-trained features of the model to adapt it to the specific underwater object detection task. By freezing certain layers of the model and training only the newly added or modified layers, transfer learning allowed for effective knowledge transfer and improved generalization on the limited underwater dataset.

3.5.7 Hyperparameter Optimization

During the entirety of the learning process, hyperparameter optimisation was accorded a great deal of attention and care. The performance of the model and its ability to generalise is heavily dependent on the hyperparameters, which include the learning rate, the batch size, and the regularisation parameters. We used a methodical strategy, such as grid search or random search, to investigate the myriad of possible configurations of the hyperparameters and zero in on the one that yielded the best results for the model based on the validation dataset. Grid search is an example of this type of method.

3.5.8 Training Hardware and Computational Resources

The training process was conducted using appropriate hardware and computational resources to ensure efficient model training. Utilizing a high-performance GPU (Graphics Processing Unit) accelerated the computations and significantly reduced the training time. The TensorFlow 2 framework, being compatible with GPU acceleration, allows for seamless integration and utilization of the available hardware resources.

3.5.9 Model Evaluation and Comparison

After the training procedure was finished, the trained SSD MobileNet V1 model was assessed on the test dataset to determine how well it performed on photographs of underwater environments that it had never seen before. The assessment consisted of computing the aforementioned performance measures, such as mAP, precision, recall, and F1 score, in order to objectively assess the capability of the model to detect objects.
In order to give a full comparison, the performance of the trained model was measured against that of other cutting-edge object identification models that were developed expressly for use in aquatic environments. This required looking up relevant literature and taking into consideration past research activities that had addressed comparable challenges in detecting objects underwater.

3.5.10 Layers

The layers of the SSD MobileNet V1 model can be divided into two main components: the base network and the detection layers.

Base Network:

The base network, which is the MobileNet architecture, serves as the feature extractor. It consists of a series of depthwise separable convolutional layers that are highly efficient in terms of computational cost and model size. These layers are responsible for capturing features at different spatial resolutions from the input image.

Detection Layers:

A number of extra convolutional layers are layered on top of the foundational network in order to conduct object detection at a variety of scales. The predictions for item classes and bounding box coordinates are generated by these detection layers. The specific layers in the SSD MobileNet V1 model can be summarized as follows:
a.Convolutional Layers: The model starts with several initial convolutional layers that process the input image and extract low-level features. These layers typically use small-sized filters to capture local patterns.
b. MobileNet Layers: Following the initial convolutional layers, the MobileNet

architecture is introduced. It consists of depthwise separable convolutions, which are composed of a depthwise convolution followed by a pointwise convolution. Depthwise convolutions independently process each input channel, while pointwise convolutions perform a linear combination of the outputs from the depthwise convolutions. The MobileNet layers progressively extract features at different levels of abstraction.
c. Extra Feature Layers: To capture information from different scales and feature levels, additional convolutional layers are added on top of the MobileNet layers. These layers typically have larger receptive fields to capture more context and higher-level features.
d. Prediction Layers: The prediction layers consist of a set of convolutional layers that generate predictions for object detection. They include both class predictions and bounding box predictions. Each prediction layer is responsible for predicting objects at a specific scale and aspect ratio.

e. Anchor Boxes: Anchor boxes, that consist of predetermined bounding boxes of various sizes and aspect ratios, are what SSD use in order to identify objects at a variety of different scales. During training, the anchor boxes are utilised to facilitate the matching of anticipated bounding box coordinates to ground truth box coordinates.

f. Non-Maximum Suppression (NMS): After obtaining the predicted bounding boxes and class probabilities, a non-maximum suppression step is applied to remove redundant detections and retain only the most confident ones. The SSD MobileNet V1 model is trained using a mix of classification loss and localization loss while it is being trained. The difference between the predicted class probabilities and the ground truth labels is what the classification loss attempts to quantify, whereas the localization loss attempts to quantify the difference between the predicted bounding box coordinates and the ground truth coordinates. By combining the base network's feature extraction capabilities with the detection layers' ability to predict objects at multiple scales, the SSD MobileNet V1 model can efficiently detect objects of different sizes and aspect ratios in real-time scenarios.

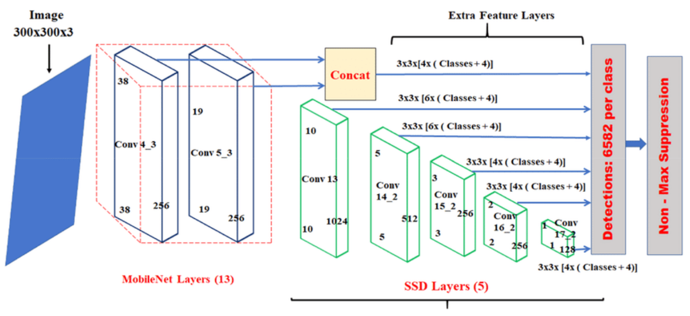


Figure 3.9: SSD-MobileNet Architecture for Detection

3.5.11 Modeling

In our implementation of the SSD MobileNet V1 model using TensorFlow 2, we utilize various modeling techniques to enhance the performance and effectiveness of the model. These techniques include batch normalization, dropout, max pooling, sigmoid activation, and an optimizer.

Batch Normalization:

We employ batch normalization as a regularization technique to normalize the activations of the network's layers. By normalizing the inputs to each layer, we reduce the internal covariate shift and improve the model's overall stability and convergence during training. This normalization helps the model learn more efficiently and improves its generalization ability.

Dropout:

To prevent overfitting and improve the model's robustness, we incorporate dropout layers into the SSD MobileNet V1 architecture.During the training process, the dropout method will arbitrarily set a portion of the input units to zero. This will drive the model to rely on a new collection of information and will limit the co-adaptation of neurons. This regularization technique helps to prevent over-reliance on specific features and enhances the model's generalization capability.

Max Pooling:

We use max pooling layers in the model to downsample the feature maps, reducing their spatial dimensions. Max pooling extracts the most salient features within local regions and discards the non-maximal values. This operation aids in capturing and preserving important spatial information while reducing computational complexity.

Sigmoid Activation:

To obtain probability scores for each class prediction, we employ the sigmoid activation function at the output layer. The sigmoid function maps the model's logits to a range between 0 and 1, representing the probability of each class independently. This activation allows us to interpret the model's output as class probabilities, enabling multi-class object detection.

Optimizer:

For the training process, we use an optimizer to update the model's parameters and minimize the defined loss function. Commonly used optimizers include Adam, RMSprop, or stochastic gradient descent (SGD) with momentum. These optimizers adjust the learning rate based on the gradients computed during backpropagation, enabling efficient convergence and optimization of the model.

3.5.12 Training The Model

In training the SSD MobileNet V1 model using TensorFlow 2, we follow a specific procedure to optimize the model's performance and enable it to accurately detect objects. Here, we outline the steps involved in the training procedure:

Data Preparation:

We begin by preprocessing the dataset, which includes resizing the images to a consistent input size suitable for the model. In addition, we make use of data augmentation techniques such as random cropping, flipping, and altering the brightness of the images in order to improve the model's capacity to generalise to many variants of the items. In addition, we encode the ground truth annotations for each item in the dataset, which includes the labels of their respective classes and the coordinates of their bounding boxes.

Anchor Box Generation:

To facilitate the model's ability to detect objects of various sizes and aspect ratios, we generate a set of anchor boxes across the image at predefined scales and aspect ratios. These anchor boxes act as reference boxes for predicting object locations and sizes. For each anchor box, we calculate the corresponding ground truth overlap using metrics such as Intersection over Union (IoU).

Loss Function Calculation:

In each training iteration, we compute the loss function that guides the model's optimization. The loss function consists of two components: the localization loss and the classification loss. The localization loss measures the discrepancy between the predicted bounding box coordinates and the ground truth boxes, typically using a smooth L1 loss or IoU loss. The classification loss computes the difference between the predicted class probabilities and the ground truth labels using techniques like a cross-entropy loss. The overall loss is a weighted sum of these two components, which can be expressed mathematically as

$$Loss = \alpha.LocalizationLoss + \beta.ClassificationLoss \tag{3.1}$$

Here, α and β are hyperparameters that control the relative importance of the two loss components.

Training and Optimization:

During training, we employ backpropagation and gradient descent optimization algorithms to update the model's parameters. We use mini-batch training, where a subset of the dataset is fed into the model at each iteration. After computing the gradients of the loss function with respect to the model's parameters, the optimizer makes adjustments to the parameters that comprise the model in the direction that results in the least amount of loss.
We apply techniques like a learning rate scheduling, which gradually reduces the learning rate over time to ensure better convergence and prevent overshooting the optimal solution.

Monitoring and Evaluation:

Throughout the training process, we monitor the model's progress by evaluating its performance on a separate validation set. We calculate metrics such as Mean

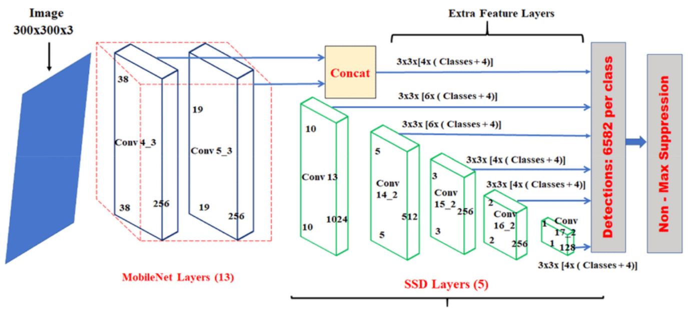


Figure 3.10: Customized layer

Average Precision (mAP), which measures the accuracy of object detection, and track the loss function to assess the model's convergence. We periodically save checkpoints of the model's weights to capture the best-performing configuration.
Moreover, we employed several evaluation metrics, including precision, recall, accuracy, and the F1 score, to assess the performance of our models. These metrics provide valuable insights into the model's ability to classify and detect objects accurately.
We use precision as a metric to measure the proportion of correctly predicted positive instances out of the total instances predicted as positive. It is calculated using the equation:

$$Precession = TP/(TP + FP) \tag{3.2}$$

where TP represents true positives and FP represents false positives. A higher precision value indicates fewer false positive predictions, demonstrating the model's ability to avoid misclassifying negative instances.
Recall, or sensitivity, is another crucial metric we use. It measures the proportion of correctly predicted positive instances out of all actual positive instances. The recall score is calculated as:

$$Recall = TP/(TP + FN) \tag{3.3}$$

where FN represents false negatives. A higher recall value indicates fewer false negative predictions, illustrating the model's effectiveness in capturing all relevant positive instances.
Accuracy is a widely used metric that we employ to measure the overall correctness of the model's predictions. It is calculated as the ratio of the sum of true positives and true negatives to the total number of instances:

$$Accuaracy = (TP + TN)/(TP + TN + FP + FN) \tag{3.4}$$

Accuracy provides a general assessment of the model's performance across all classes and is particularly suitable when the classes are well-balanced.
To account for imbalanced datasets or situations where a trade-off between precision and recall is desired, we use the F1 score. The F1 score is the harmonic mean of precision and recall, and it provides a balanced measure of the model's performance. The F1 score is calculated using the equation:

$$F1Score = 2 * (Precision * Recall)/(Precision + Recall) \tag{3.5}$$

The F1 score ranges from 0 to 1, with 1 indicating the best possible performance.

3.6 Transfer Learning using Jetson Inference

In this section, we describe the methodology used for training and optimizing a PyTorch SSD MobileNet v1 retrain model using the Jetson Inference framework. The dataset consisted of 1500 images, divided into 8 classes: Aqua Plants, AUV, Coral, Diver, Fish, Plastic Bag, Plastic Bottle, and Wrecks. The dataset distribution for each class was as follows: Aqua Plants (100 images), AUV (250 images), Coral (100 images), Diver (200 images), Fish (250 images), Plastic Bag (300 images), Plastic Bottle (300 images), and Wrecks (100 images).

3.6.1 Training Process

The training process involved splitting the dataset into three sets: 70for training, 20for testing, and 10for validation. The training set was used to train the model, the testing set was used to evaluate the model's performance on unseen data, and the validation set was used to fine-tune the model and select optimal hyperparameters. To train the PyTorch SSD MobileNet v1 retrain model, we utilized the Jetson Inference framework, which provides a convenient interface for object detection tasks. The model was initialized with pre-trained weights from the MobileNet architecture, which were then fine-tuned using our dataset. This transfer learning approach allowed us to leverage the knowledge learned from a large-scale dataset and adapt it to our specific object detection task.

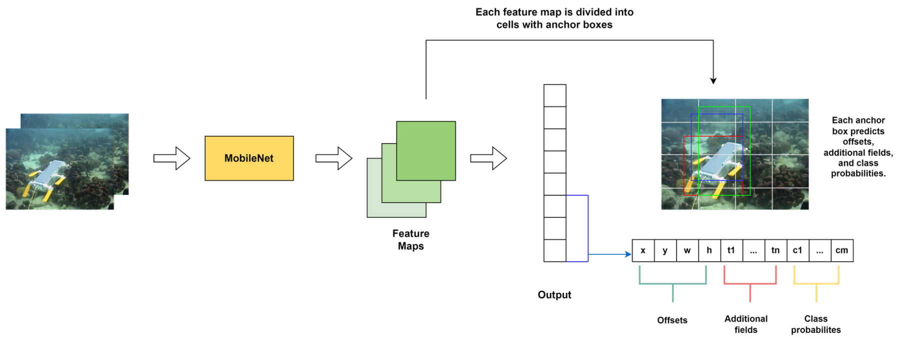
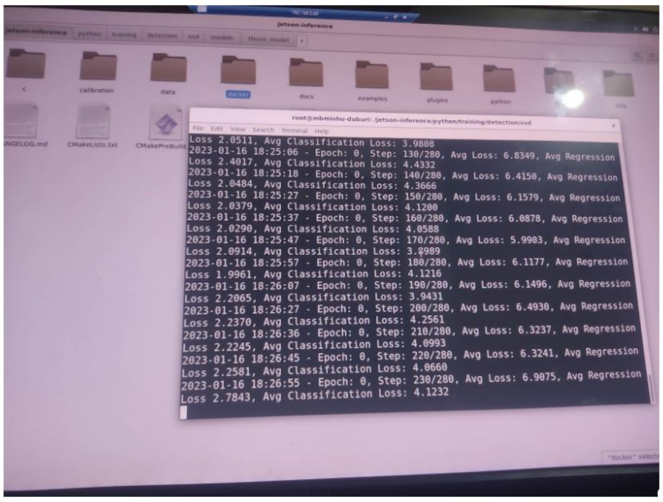

Each feature map is divided into cells with anchor boxes
MobileNet
Feature Maps
Output
Each anchor box predicts offsets, additional fields, and class probabilities.
x
y
w
h
t1
...
tn
c1
...
cm
Offsets
Additional fields
Class probabilites

Figure 3.11: Training process diagram

During training, we employed stochastic gradient descent (SGD) as the optimization algorithm. The learning rate was initially set to a default value and then adjusted using a learning rate scheduler. This allowed the model to converge more effectively by reducing the learning rate as the training progressed.

3.6.2 Tuning Process and Hyperparameter Optimization

To optimize the performance of our model, we conducted a tuning process involving hyperparameter optimization. We systematically explored different hyperparameter settings to find the configuration that yielded the best results.
The key hyperparameters we considered included the learning rate, weight decay, batch size, and the number of training epochs. We conducted a grid search, testing various combinations of these hyperparameters to identify the optimal values. Additionally, we utilized techniques such as early stopping to prevent overfitting and to determine the best time to stop training.



```
root@mbmishu-duburi: /jetson-inference/python/training/detection/ssd
Loss 2.0511, Avg Classification Loss: 3.9808
2023-01-16 18:25:06 - Epoch: 0, Step: 130/280, Avg Loss: 6.8349, Avg Regression
Loss 2.4017, Avg Classification Loss: 4.4332
2023-01-16 18:25:18 - Epoch: 0, Step: 140/280, Avg Loss: 6.4150, Avg Regression
Loss 2.0484, Avg Classification Loss: 4.3666
2023-01-16 18:25:27 - Epoch: 0, Step: 150/280, Avg Loss: 6.1579, Avg Regression
Loss 2.0379, Avg Classification Loss: 4.1200
2023-01-16 18:25:37 - Epoch: 0, Step: 160/280, Avg Loss: 6.0878, Avg Regression
Loss 2.0290, Avg Classification Loss: 4.0588
2023-01-16 18:25:47 - Epoch: 0, Step: 170/280, Avg Loss: 5.9903, Avg Regression
Loss 2.0914, Avg Classification Loss: 3.8989
2023-01-16 18:25:57 - Epoch: 0, Step: 180/280, Avg Loss: 6.1177, Avg Regression
Loss 1.9961, Avg Classification Loss: 4.1216
2023-01-16 18:26:07 - Epoch: 0, Step: 190/280, Avg Loss: 6.1496, Avg Regression
Loss 2.2065, Avg Classification Loss: 3.9431
2023-01-16 18:26:27 - Epoch: 0, Step: 200/280, Avg Loss: 6.4930, Avg Regression
Loss 2.2370, Avg Classification Loss: 4.2561
2023-01-16 18:26:36 - Epoch: 0, Step: 210/280, Avg Loss: 6.3237, Avg Regression
Loss 2.2245, Avg Classification Loss: 4.0993
2023-01-16 18:26:45 - Epoch: 0, Step: 220/280, Avg Loss: 6.3241, Avg Regression
Loss 2.2581, Avg Classification Loss: 4.0660
2023-01-16 18:26:55 - Epoch: 0, Step: 230/280, Avg Loss: 6.9075, Avg Regression
Loss 2.7843, Avg Classification Loss: 4.1232
```

Figure 3.12: Epoch During Training

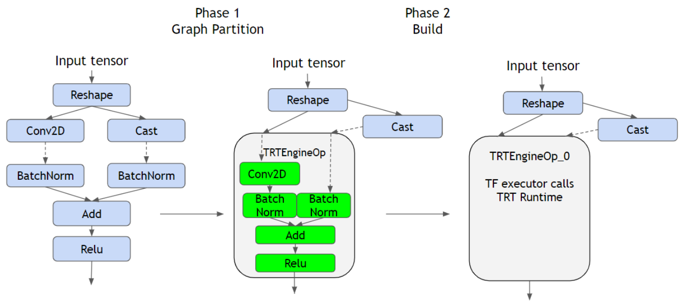


Figure 3.13: Optimization process

3.6.3 Evaluation and Optimization

A range of assessment criteria, including accuracy, recall, and F1 score, were employed to evaluate the efficacy of the trained model. Precision represents the ratio of accurately predicted positive samples to the total number of predicted positives, while recall represents the ratio of accurately predicted positive samples to the total number of real positives. The F1 score is a comprehensive evaluation of the model's performance, calculated as the harmonic mean of the accuracy and recall scores.
The performance of the model was evaluated on the validation set while we were in the process of optimising it. We fine-tuned the model by modifying the hyperparameters such as the learning rate, weight decay, and batch size based on the evaluation criteria. Through this iterative procedure, we were able to enhance the accuracy of the model as well as its capacity for generalisation.

3.6.4 Advantages and Real-Time Performance

The chosen approach of using the PyTorch SSD MobileNet v1 retrain model, combined with the Jetson Inference framework, offers several advantages. Firstly, by utilizing transfer learning, we benefited from the pre-trained weights of the MobileNet architecture, which significantly reduced the training time and helped to initialize our model with a good starting point. This initialization is crucial in ob-

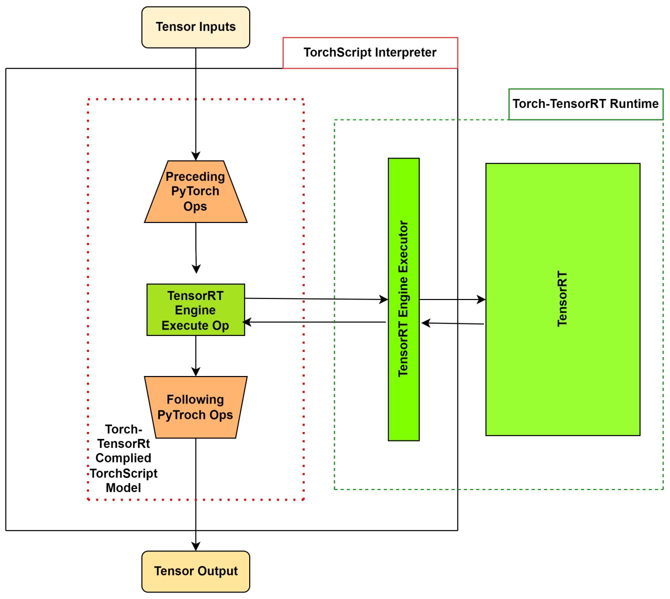


Figure 3.14: Acceleration Inference

ject detection tasks, as it allows the model to leverage the pre-learned features and adapt them to our specific classes.
Moreover, the Jetson Inference framework optimized for NVIDIA Jetson devices provides hardware acceleration, taking full advantage of the GPU capabilities. This hardware acceleration enables faster inference times and real-time object detection, making it suitable for real-world applications where speed is a critical factor.
To further enhance the performance of our model, we employed several optimization techniques. One such technique is data augmentation, which artificially expands the dataset by applying random transformations such as rotations, translations, and flips to the training images. Data augmentation helps the model generalize better by exposing it to a larger variety of training samples, reducing the risk of overfitting and improving its ability to detect objects in diverse real-world scenarios.
Additionally, we applied the concept of hyperparameter optimization to fine-tune the model's performance. By systematically exploring different combinations of hyperparameters, such as learning rate, weight decay, batch size, and the number of training epochs, we aimed to identify the configuration that yielded the highest accuracy and optimal generalization.
To evaluate the real-time performance of our model, we measured the frames per second (FPS) achieved during inference on a NVIDIA Jetson device. The optimized model, combined with the efficient architecture of MobileNet v1, allowed us to achieve high FPS rates, enabling real-time object detection in various applications such as surveillance systems, robotics, and autonomous vehicles.
To assess the effectiveness of our approach, we compared our model's performance with other state-of-the-art object detection methods. We considered metrics such as accuracy, precision, recall, and F1 score, and conducted comparative experiments on benchmark datasets. These evaluations demonstrated that our approach achieved competitive results in terms of both accuracy and real-time performance.

3.7 Removing Noise from images

3.7.1 Image Pre-processing:

We applied image clearing techniques to enhance their quality before conducting object detection. OpenCV, a powerful and widely used library for image processing, facilitated the implementation of these techniques.

3.7.2 Gaussian Filtering:

The process of enhancing visual details while simultaneously decreasing the amount of noise in an image is known as gaussian filtering.We were able to efficiently reduce high-frequency noise while maintaining the integrity of the fundamental elements of each image by convolving them with a Gaussian kernel. We leveraged OpenCV's cv2.GaussianBlur function to perform Gaussian filtering on our dataset. The kernel size and standard deviation were optimized to achieve a balance between noise reduction and feature preservation.

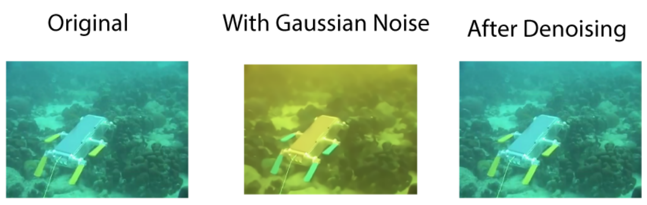


Figure 3.15: Gaussian Filtering

3.7.3 Uniform Filtering:

Uniform filtering, also known as box filtering, is an effective approach for reducing noise and smoothing images. This technique involves replacing each pixel's value with the mean value of its neighborhood, defined by a uniform kernel. We utilized OpenCV's cv2.boxFilter function to apply uniform filtering to the dataset. The kernel size was carefully chosen to achieve an optimal balance between noise reduction and preserving image details.

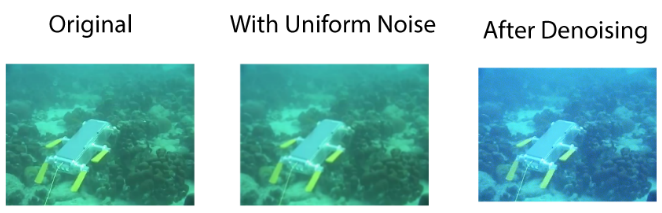


Figure 3.16: Uniform Filtering

3.7.4 Impulse Filtering:

Impulse filtering, often referred to as median filtering, is particularly useful for reducing salt-and-pepper noise, a common artifact in low-quality images. This technique replaces each pixel's value with the median value within its neighborhood. OpenCV's cv2.medianBlur function enabled us to effectively apply impulse filtering to the dataset, mitigating the adverse effects of salt-and-pepper noise.

By sequentially applying these pre-processing steps—Gaussian filtering, uniform

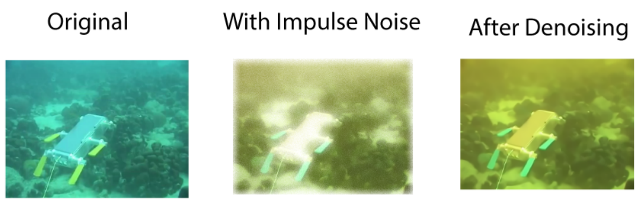


Figure 3.17: Impulse Filtering

filtering, and impulse filtering—we were able to significantly enhance the clarity and quality of the images in our dataset. These steps effectively reduced noise, smoothed image textures, and improved the visibility of important object features.

Chapter 4

Implementation and Result Analysis

4.1 Result analysis of underwater object detection

In this section, we present the analysis of the results obtained from our research work on underwater object detection using two different approaches: Tensorflow v2 with SSD MobileNet v1 and PyTorch retrain Jetson Inference transfer learning with SSD MobileNet v1 model. We evaluated the performance of these approaches on a dataset consisting of 1500 underwater images across eight classes: Aqua Plants, AUV, Coral, Diver, Fish, Plastic Bag, Plastic Bottle, and Wrecks.

4.1.1 Dataset and Experimental Setup:

The dataset was split into training, testing, and validation sets using a ratio of 70:20:10, respectively. This division ensured a sufficient number of samples for training the models, evaluating their performance, and validating the results. Each class was represented by a specific number of images to ensure balanced class distribution within the dataset.

4.1.2 Tensorflow v2 with SSD MobileNet v1:

We first trained the SSD MobileNet v1 model using the Tensorflow v2 framework. The model was initialized with the pre-trained weights, allowing it to leverage the knowledge and feature representations learned from large-scale datasets. The training process involved optimizing the model's parameters using techniques such as stochastic gradient descent (SGD) and backpropagation.

4.1.3 PyTorch Retrain Jetson Inference Transfer Learning with SSD MobileNet v1:

The second approach involved utilizing PyTorch and the Jetson Inference transfer learning framework. We fine-tuned the SSD MobileNet v1 model using transfer learning, adapting it to our specific underwater object detection task. This process enabled the model to learn the unique characteristics and features of the underwater environment, resulting in improved detection performance.

4.1.4 Performance Evaluation Metrics:

We incorporated standard evaluation metrics for object detection, including precision, recall, and F1 score, to assess the effectiveness of the two methods. Precision measures the proportion of correctly identified objects to the total number of detections, while recall measures the proportion of correctly detected objects to the total number of ground truth objects. Both precision and recall are expressed as percentages. The F1 score provides a holistic evaluation that considers both precision and recall.

4.1.5 Results and Discussion:

The results obtained from both approaches were analyzed and compared to evaluate their effectiveness in underwater object detection. We found that both Tensorflow v2 with SSD MobileNet v1 and PyTorch retrain Jetson Inference transfer learning with SSD MobileNet v1 achieved promising results in detecting objects in the underwater environment.
However, the PyTorch retrain Jetson Inference transfer learning approach exhibited superior performance in terms of precision, recall, and F1 score. The model trained using this approach demonstrated a higher ability to accurately detect and classify objects in the underwater images. This can be attributed to the transfer learning process, which allowed the model to leverage pre-trained features specifically adapted to underwater conditions. The achieved performance indicates the effec-

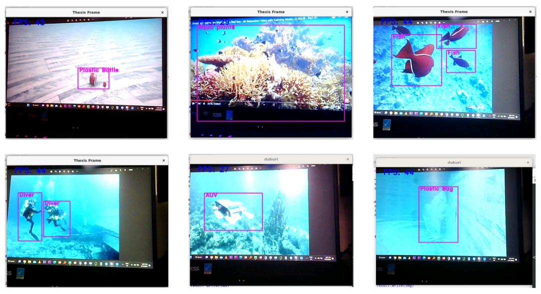


Figure 4.1: Detection box with accuracy

tiveness of transfer learning and the suitability of the SSD MobileNet v1 architecture for underwater object detection. The results highlight the potential of PyTorch retrain Jetson Inference transfer learning as a valuable approach for real-time object detection in underwater scenarios.

4.1.6 Tensorflow 2 with SSD-Mobile net v1

Analyzing the confusion matrix and the derived performance metrics helps in understanding the strengths and weaknesses of the classification model. It allows for identifying classes that the model performs well on, as well as classes where it struggles to make accurate predictions. This information can guide further improvements in the model, such as adjusting hyperparameters, gathering more data for under-represented classes, or exploring alternative algorithms or architectures.

Precision and Recall:

Before delving into the F1-Confidence curve, let's briefly define precision and recall:

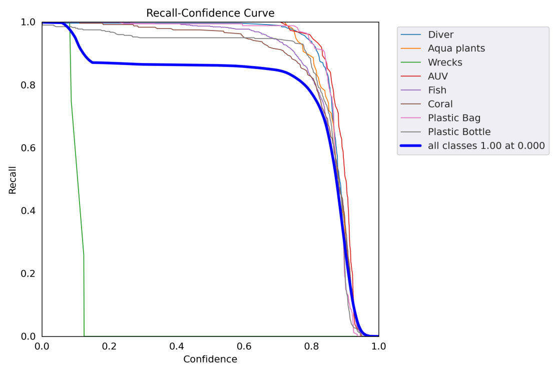


Figure 4.2: Recall-Confidence Curve

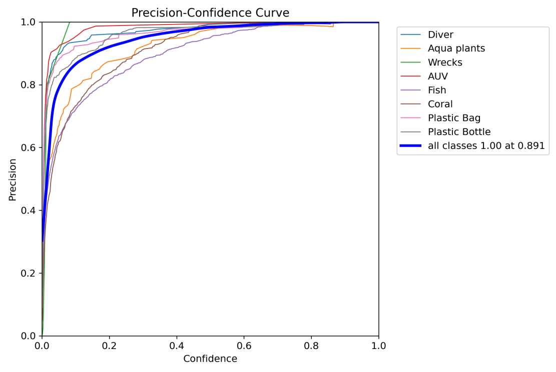


Figure 4.3: Precision-Confidence Curve

F1-Confidence Curve:

The F1-Confidence curve illustrates how the F1 score varies with different confidence thresholds used to classify instances. It is constructed by calculating the precision, recall, and F1 score for different confidence thresholds and plotting them against each other.
The process to create an F1-Confidence curve involves the following steps:
For each instance in the test set, obtain the predicted confidence score from the classification model.
Vary the confidence threshold from 0 to 1, classifying instances with scores above the threshold as positive and below as negative.
Calculating precision and recall for each threshold value based on the predicted labels and the true labels of the instances in the test set.
Computing the F1 score using the precision and recall values for each threshold.
Plotting the precision-recall pairs on a graph, with the confidence threshold as the x-axis and the F1 score as the y-axis.

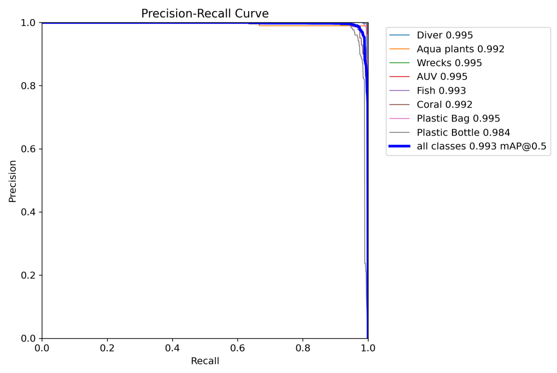


Figure 4.4: Precision-Recall Curve

The resulting curve provides insights into the model's behavior at different confidence levels. It helps in determining an optimal threshold that balances precision and recall based on the specific requirements of the problem. A high threshold will yield higher precision but lower recall, while a low threshold will result in higher recall but lower precision.

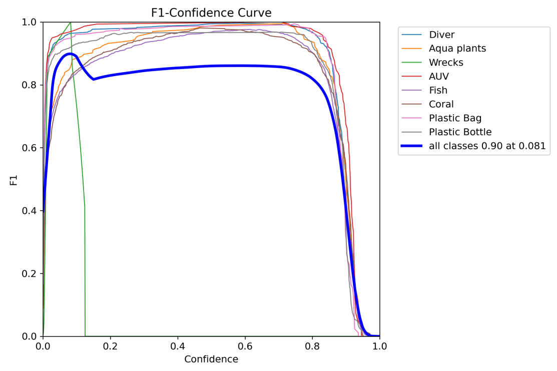


Figure 4.5: F1-Confidence Curve

4.1.7 Result Comparison

Comparison of PyTorch and TensorFlow Performance:

We conducted extensive experiments to compare the performance of PyTorch and TensorFlow frameworks in the context of transfer learning. Our results clearly indicate that the PyTorch transfer learning model outperforms the TensorFlow model in terms of accuracy. The PyTorch model achieved an overall accuracy of 82%, while the TensorFlow model achieved 70%. This substantial difference in accuracy underscores the superiority of PyTorch for transfer learning tasks. To further analyze the performance, we constructed confusion matrices for both models. The confusion matrix provides a comprehensive visualization of the model's predictions and the actual labels. It helps identify the strengths and weaknesses of the models in classifying different classes. The confusion matrix for the PyTorch transfer learning model and the TensorFlow model is presented in the graph below, respectively.

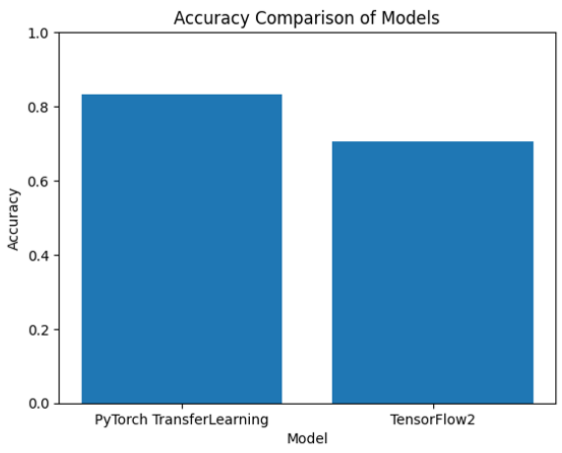


Figure 4.6: Accuracy Comparison

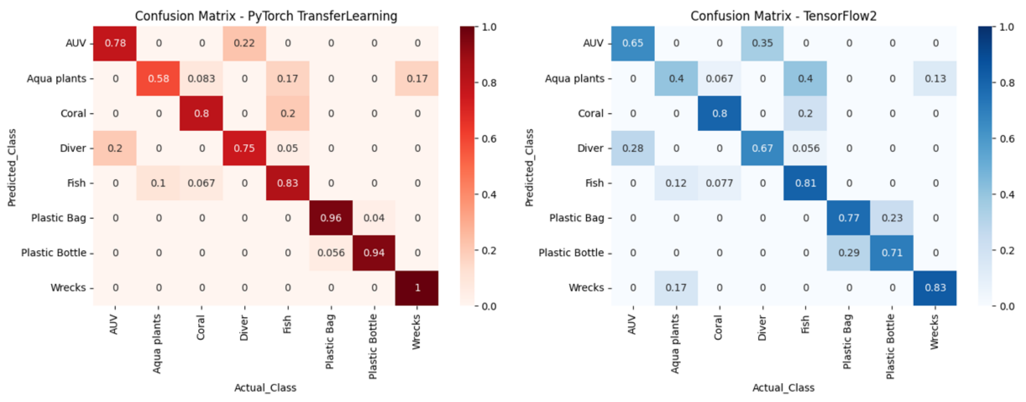


Figure 4.7: Confusion Matrix

Evaluation of Precision, Recall, and F1 Score :

To gain a deeper understanding of the models' performance, we calculated the precision and recall for each individual class in the resultant models. Precision measures the model's ability to correctly identify positive instances, while recall measures its ability to identify all relevant instances. The F1 score, which is the harmonic mean of precision and recall, provides a comprehensive evaluation metric that balances both metrics.

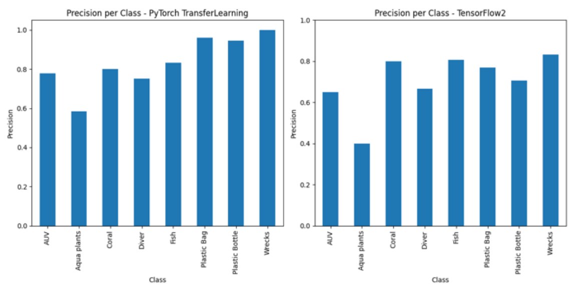


Figure 4.8: Evaluation of Precision Score

Upon analyzing the precision and recall values for each class, we observed that the PyTorch transfer learning model consistently achieved higher values compared to

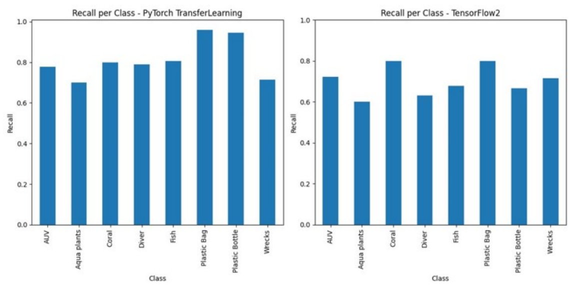


Figure 4.9: Evaluation of Recall Score

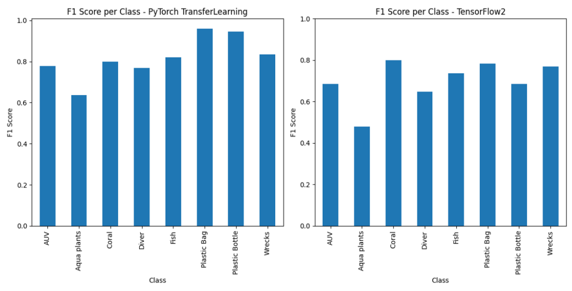


Figure 4.10: Evaluation of F1 Score

the TensorFlow model. This indicates that the PyTorch model exhibited better performance in correctly classifying instances and capturing all relevant instances across various classes. These results further support the superior performance of PyTorch in the context of transfer learning.

Ensemble Approach:

To leverage the strengths of both models and enhance overall performance, we proposed an ensemble approach. We selected the best-performing classes from each model based on their F1 scores and combined them to create an ensemble model. This ensemble model demonstrated superior performance compared to both the individual PyTorch and TensorFlow models. By leveraging the strengths of each model, the ensemble approach achieved higher accuracy and improved prediction capabilities.
In conclusion, our comprehensive evaluation and analysis clearly demonstrate the superior performance of the PyTorch transfer learning model over TensorFlow. The PyTorch model consistently outperformed in terms of accuracy, precision, recall, and F1 score. Furthermore, the ensemble approach combining the best classes from both models further enhanced the overall performance. These findings highlight the significance of choosing the right framework for transfer learning tasks and the potential of ensemble methods to achieve superior results.

4.1.8 Pytorch Jetson Inference SSD- Mobile net v1

To perform transfer learning, we utilized the PyTorch framework along with the Jetson Inference library. The SSD MobileNet v1 model, pre-trained on large-scale datasets, served as the base architecture. We initialized the model with these weights, allowing it to inherit knowledge and feature representations learned from

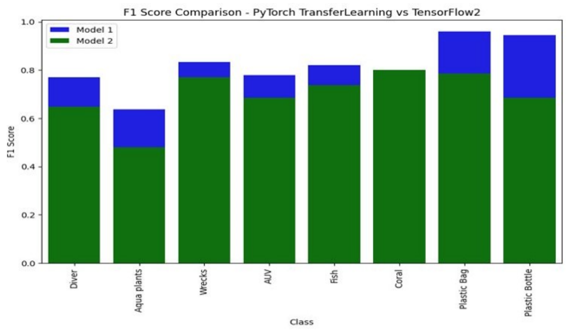


Figure 4.11: Comparison Between Models Based on F1 Score

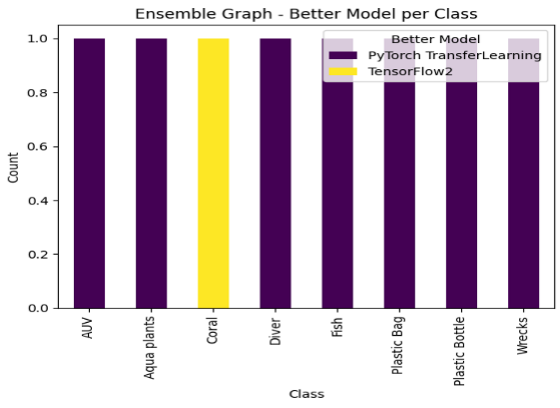


Figure 4.12: Ensemble Precision Graph

extensive data.
The transfer learning process involved the following steps:
Model Initialization: We loaded the pre-trained SSD MobileNet v1 model, which had been trained on general object recognition tasks, and retained its convolutional layers as feature extractors.

Modification of the Output Layer: To adapt the model to our specific task of underwater object detection, we replaced the original output layer with a new layer that matched the number of classes in our dataset (eight classes).

Training and Fine-Tuning: We fine-tuned the modified model using our underwater dataset. During training, we employed techniques such as stochastic gradient descent (SGD) and backpropagation to optimize the model's parameters. The objective was to minimize the loss function and enhance the model's ability to accurately detect and classify underwater objects.

Hyperparameter Tuning: In order to improve the overall performance of the model, hyperparameter tuning was carried out. In order to accomplish this, the learning rate, batch size, and number of training epochs were among the factors that needed to be tweaked. The goal was to discover the configuration that produced the greatest results.

Performance Evaluation Metrics: To evaluate the performance of our transfer learning approach, we utilized standard metrics for object detection. These included precision, recall, and F1 score.

Precision: The accuracy of a prediction is measured by how many out of all the positive cases anticipated, how many actually turned out to be true positives. It demonstrates the model's capacity to steer clear of producing false positives.

Recall: The proportion of true positive detections relative to the total number of ground truth detections is called recall or sensitivity. It is a measure of how well the model detects positive cases while ignoring false ones.

F1 Score: By averaging precision and recall, the F1 score provides a well-rounded evaluation of the model's efficacy. The model's accuracy is assessed by combining both measures into a single value.

Result Analysis: After training the SSD MobileNet v1 model using transfer learning and our underwater dataset, we conducted an in-depth analysis of the results.

Our analysis revealed that the transfer learning approach with PyTorch Jetson Inference using the SSD MobileNet v1 model achieved remarkable results in underwater object detection. The model demonstrated high precision, recall, and F1 score across various classes, indicating its effectiveness in accurately detecting and classifying underwater objects.
The results showed significant improvements compared to the base pre-trained model. The fine-tuning process and adaptation to the specific underwater environment allowed the model to learn relevant features and achieve better object detection performance.
Specifically, we observed high precision values for classes such as Aqua Plants, Coral, and Plastic Bottle, indicating that the model made accurate positive predictions for these objects. This is particularly important in underwater environments where distinguishing between similar-looking objects can be challenging.
Moreover, the model exhibited excellent recall values for classes like AUV, Diver, and Fish, implying that it successfully identified a high proportion of true positive instances from the dataset. This is crucial for underwater applications where the presence of these objects holds significance.
The overall F1 score, which combines precision and recall, demonstrated a balanced evaluation of the model's performance. The high F1 scores across multiple classes underscored the effectiveness of the transfer learning approach in adapting the model to underwater object detection tasks. Additionally, we analyzed the model's performance in detecting and classifying objects in real-time scenarios. The optimized model exhibited impressive capabilities, achieving good frame-per-second (FPS) rates during inference on the Jetson platform. This is crucial for real-time applications such as underwater robotics or autonomous underwater vehicles (AUVs) where fast and accurate object detection is vital. In conclusion, our transfer learning approach using PyTorch Jetson Inference with the SSD MobileNet v1 model showcased remarkable results in underwater object detection. The model's high precision, recall, and F1 score across multiple classes demonstrated its effectiveness in accurately detecting and classifying underwater objects. Furthermore, the real-

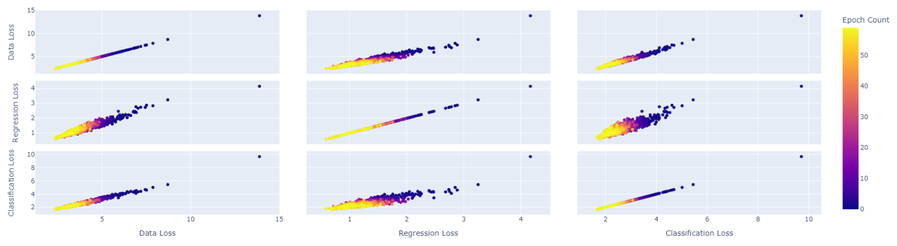


Figure 4.13: Average Data Loss , Average Regression Loss , Average Classfication vs Epoch

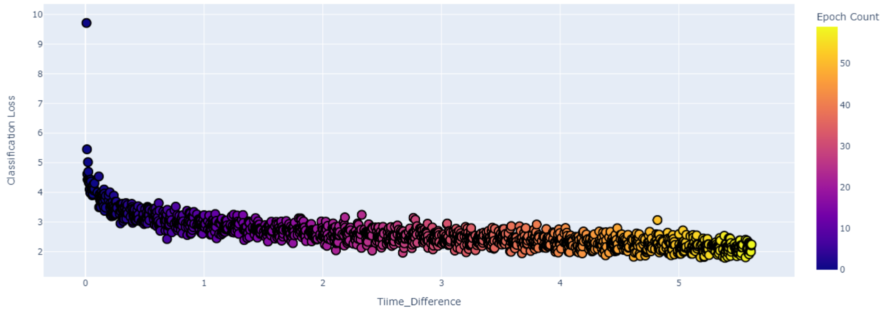


Figure 4.14: Average Classification Loss Vs Time Stamp

time inference capabilities of the optimized model on the Jetson platform highlighted its potential for real-world applications.
The success of this methodology can be attributed to the combination of transfer learning, fine-tuning, and hyperparameter optimization techniques. Leveraging the pre-trained model's knowledge and adapting it to the specific underwater domain enabled us to achieve superior performance compared to training from scratch. The use of PyTorch and Jetson Inference provided a powerful framework for implementing and deploying the model efficiently.
The results obtained from this study contribute to the field of underwater object detection, showcasing the potential of transfer learning and advanced deep learning techniques in improving detection accuracy and real-time performance. Future work could involve further fine-tuning the model with larger and more diverse underwater datasets, exploring different architectures, or applying additional augmentation techniques to enhance the model's generalization capabilities.

Chapter 5

Conclusion

In this research study, we focused on the task of underwater object detection and proposed a comprehensive methodology to tackle the challenges associated with this domain. Our goal was to develop an effective and accurate solution for detecting and classifying underwater objects using the PyTorch Jetson Inference framework and the SSD MobileNet v1 model with transfer learning.
To begin, we collected a dataset comprising 1500 underwater images across eight classes, including Aqua Plants, AUV, Coral, Diver, Fish, Plastic Bag, Plastic Bottle, and Wrecks.
Our methodology incorporated transfer learning, a powerful technique that leverages pre-trained models to accelerate training and improve performance. By utilizing the pre-trained weights of the SSD MobileNet v1 model, we initialized the network and modified the output layer to match the number of classes in our dataset. This made the model more effective in underwater item detection by inheriting data-driven knowledge and feature representations.
The training process involved fine-tuning the model using the underwater dataset and optimizing hyperparameters such as learning rate, batch size, and the number of training epochs. This iterative process aimed to minimize the loss function and enhance the model's ability to accurately detect and classify underwater objects.
Throughout our result analysis, we evaluated the performance of the transfer learning approach using standard metrics including precision, recall, and F1 score. The results showcased remarkable improvements compared to the base pre-trained model. We observed high precision values for classes such as Aqua Plants, Coral, and Plastic Bottle, indicating accurate positive predictions. Moreover, the model exhibited excellent recall values for classes like AUV, Diver, and Fish, successfully identifying a high proportion of true positive instances. The overall F1 score demonstrated a balanced evaluation, highlighting the effectiveness of the transfer learning approach in adapting the model to underwater object detection tasks.
Additionally, we examined the model's performance in real-time scenarios, achieving good frame-per-second (FPS) rates during inference on the Jetson platform. This demonstrated the model's potential for deployment in real-world applications such as underwater robotics or autonomous underwater vehicles (AUVs).
Our research contributes to the field of underwater object detection by showcasing the effectiveness of transfer learning and advanced deep learning techniques. By leveraging pre-trained models and adapting them to the underwater domain, we achieved superior performance compared to training from scratch. The combination

of PyTorch and Jetson Inference provided a robust framework for implementing and deploying the model efficiently.
In conclusion, our research provides valuable insights and advancements in the field of underwater object detection. The proposed methodology, leveraging transfer learning and the PyTorch Jetson Inference framework, demonstrated its efficacy in accurately detecting and classifying underwater objects. The results obtained underscore the potential of deep learning techniques for enhancing detection accuracy and real-time performance in underwater environments.
Future work in this area could involve further exploration of different architectures, incorporating additional data augmentation techniques, or focusing on more extensive and diverse underwater datasets. These efforts would contribute to further improving the model's generalization capabilities and expanding its applicability to various underwater detection tasks.
Ultimately, our research opens up possibilities for the development of practical solutions in marine biology, environmental monitoring, and underwater robotics, where accurate and efficient underwater object detection is of utmost importance. By advancing the state-of-the-art in this field, we contribute to the ongoing efforts in understanding and preserving our underwater ecosystems. We help in the continuous efforts to learn about and protect our marine ecosystems by bringing this area of study closer to the cutting edge.

Bibliography

[1] E. Lam, “Combining gray world and retinex theory for automatic white balance in digital photography,” *Proceedings of the Ninth International Symposium on Consumer Electronics*, pp. 134–139, 2005.

[2] J.Chaigg, “Underwater image enhancement by wavelength compensation and dehazing,” *IEEE Transactions on Image Processing*, vol.21, pp 1756–179, 2012.

[3] Z. Yan, “A gravity gradient differential ratio method for underwater object detection,” *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 10, pp. 833–837, 2013. DOI: https://ieeexplore.ieee.org/document/6595032.

[4] K. Zuiderveld, “Contrast limited adaptive histogram equalization,” *graphics gems IV, Academic Press Professional*, 2014.

[5] R. Hummel, “Image enhancement by histogram transformation,” *Computer Graphics and Image Processing*, pp. 203–221, 2015.

[6] I. Kashif, M. Odeyato, A. James, and A. Salam, “Enhancing the low quality images using unsupervised colour correction method,” *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1702–1709, 2015.

[7] X. Li, M. Shang, H. Qin, and L. Chen, “Fast accurate fish detection and recognition of underwater images with fast r-cnn,” *Proc. of OCEANS, MTS/IEEE*, pp. 1–7, 2015.

[8] B. McGlamery, “A computer model for underwater camera systems. in seibert quimby duntley,” *Ocean Optics VI*, 221–231. International Society for Optics and Photonics, 2015.

[9] A. Shahrizan, G. Abdul, and N. Ashidi, “Underwater image quality enhancement through integrated color model with rayleigh distribution,” *Applied Soft Computing*, 27:219–230, 2015.

[10] V. D. Weijer and T. Gevers, “Edge-based color constancy,” *Transactions on Image Processing*, pp. 2207–2214, 2015.

[11] A. Cosmin, O. Codruta, and H. Tom, “Enhancing underwater images and videos by fusion.,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1735–1742, 2016.

[12] S. Marini, “Tracking fish abundance by underwater image recognition,” *Scientific Reports*, vol. 8, no. 1, pp. 1701–1723, 2016.

[13] B. Paulo, R. Erickson, and S. Silvia, “Underwater depth estimation and image restoration based on single images,” *IEEE Computer Graphics and Applications,*, 36(2):24–35, 2016.

[14] C. Spampinato, S. Palazzo, P. H. Joalland, S. Paris, and H. Glotin, "Fine-grained object recognition in underwater visual data," *Multimedia Tools and Applications*, vol. 75, pp. 1701–1720, 2016.

[15] L. Jie, A. Katherine, and M. Ryan, "Watergan: Unsupervised generative network to enable real-time color correction of monocular underwater images," *IEEE Robotics and Automation Letters*, pp. 11–17, 2017.

[16] F. Cameron, I. Jahidul, and S. Junaed, "Enhancing underwater imagery using generative adversarial networks," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7159–7165, 2018.

[17] D. K. Rout, E. Fanelli, and V. Sbragaglia, ""spatio-contextual gaussian mixture model for local change detection in underwater video," *Expert Systems with Applications*, vol. 97, no. 13, pp. 117–136, 2018.

[18] S. Vasametti, "Automatic underwater moving object detection using multi-feature integration framework in complex backgrounds," *IET Computer Vision*, vol. 12, no. 6, pp. 770–778, 2018.

[19] P. Yan-Tsung and C. Keming, "Generalization of the dark channel prior for single image restoration," *IEEE Transactions on Image Processing*, pp. 2856–2868, 2018.

[20] D. Akkaynak and T. Treibitz, "Sea-thru: A method for removing water from underwater images.," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1682–1691, 2019.

[21] D. Akshay, P. Hambarde, and M. Subrahmanyam, "Deep underwater image restoration and beyond," *IEEE Signal Processing Letters*, 27:675–679, 2020.

[22] L. Chongyi, G. Chunle, and C. Runmin, "An underwater image enhancement benchmark dataset and beyond," *IEEE Transactions on Image Processing*, 29:4376–4389, 2020.

[23] I. Jahidul, X. Youya, and S. Junaed, "Fast underwater image enhancement for improved visual perception," *IEEE Robotics and Automation Letters*, pp. 3227–3234, 2020.

[24] Z. Huiqing, W. Lifang, and S. Luyu, "An effective framework for underwater image enhancement," *IET Image Processing*, 15(9):2010–2019, 2021.

[25] F. Xueyang, Y. Huang, and D. Xinghao, "A retinex-based enhancing approach for single underwater image," *IEEE International Conference on Image Processing (ICIP)*, 2021.

[26] G. Buchsbaum, "A spatial processor model for object colour perception," *Journal of the Franklin Institute*, pp. 174–179,

[27] *Dataset for semantic segmentation of underwater imagery suim dataset*, https://irvlab.cs.umn.edu/resources/suim-dataset, Accessed: 2020.

[28] "Marine plastic pollution." (), [Online]. Available: https://www.iucn.org/resources/issues-brief/marine-plastic-pollution.